

Парадигма развития науки

А. Е. Кононюк

**Основы теории
облачных технологий**

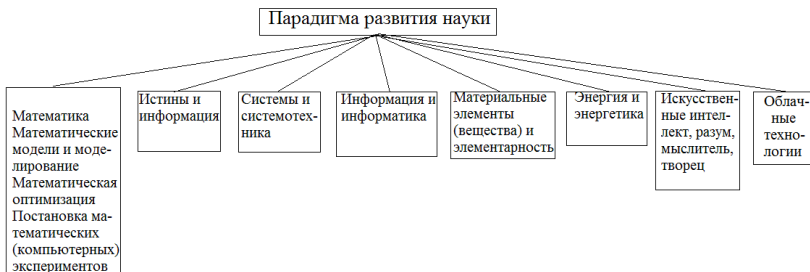
**Киев
«Освіта України»
2018**



Кононюк Анатолий Ефимович



Концептуальная схема парадигмы развития науки



УДК 51 (075.8)

ББК В161.я7

К65

Рецензент:

Н.К.Печурин - д-р техн. наук, проф. (Национальный авиационный университет).

Кононюк А. Е.

К213 Основы теории облачных технологий. — К.: Освіта України. 2018.—710 с.

ISBN 978-966-373-693-8

Работа посвящена систематическому изложению общих формализмов, математических моделей и алгоритмических методов, которые могут быть использованы при моделировании и исследованиях математических моделей объектов облачных технологий.

Развиваются представления и методы решения задач облачных технологий, основанные на теориях эвристического поиска и автоматизированного проектирования облачных управляющих систем, а также процедуральные методы, базирующиеся на классе проблемно-ориентированных языков, сочетающих свойства языков программирования и автоматических решателей задач отображения облачных технологий различными математическими средствами.

В работе излагаются основы теории облачных технологий такими математическими средствами как: множества, отношения, поверхности, пространства, алгебраические системы, матрицы, графы, математическая логика и др.

Для бакалавров, специалистов, магистров, аспирантов, докторантов всех специальностей.

УДК 51 (075.8)

ББК В161.я7

ISBN 978-966-373-693-8

© Кононюк А. Е., 2018

© Освіта України, 2018

Оглавление

Введение.....	9
1. Предмет, задачи и методы теории облачных технологических систем.....	15
1.1. Базовые понятия и определения.....	15
1.2. Предмет теории облачных технологий вычислительных систем.....	34
1.3. Классификация облачных вычислительных систем.....	38
1.4. Классы проблем облачных технологий.....	44
1.5. Классификация услуг, предоставляемых облачными технологиями.....	56
1.6. Типология облачных технологических систем.....	72
1.7. Классификация автоматизированных процессов.....	82
2. Особенности облачных технологий и облачных вычислений.....	98
2.1. Особенности облачных вычислений.....	98
2.2. Облачные технологии.....	104
2.2.1. Некоторые особенности облачных технологий.....	106
2.2.2. Услуги, предоставляемые облачными технологиями	108
2.2.3. Возможности облачных технологий.....	112
2.2.4. Примеры использования облачных технологий.....	112
2.2.5. Работа с документами в облачных технологиях.....	114
2.2.6. Облачные технологии и хранение данных.....	118
2.3. Основные задачи теории облачных технологических систем...	124
2.4. Общая характеристика методов теории облачных технологических систем.....	128
3. Технологические процессы и их отличительные признаки.....	133
3.1. Система преобразований и ее модель.....	133
3.2. Элементы системы преобразований.....	136
3.3. Технологический процесс и его модель.....	139
3.4. Операнды технологического процесса.....	147
3.5. Структура технологического процесса.....	148
3.6. Типы операций.....	150
3.7. Побочные входы и выходы.....	152
3.8. Операторы технологического процесса.....	152
3.9. Характеристики и оценка технологического процесса.....	154
3.10. Классификация технологических процессов.....	157
3.11. Производственная деятельность, связанная с технологическим процессом.....	158
3.12. Представление технологических процессов.....	160
4. Облачные вычислительные процессы и их модели.....	162

4.1. Логическая организация и порядок выполнения программ.....	162
4.2. Сложность задач и потребность в ресурсах памяти.....	166
4.3. Трудоемкость алгоритмов и марковские модели облачных вычислительных процессов.....	168
4.4. Оценка трудоемкости алгоритмов методами теории марковских цепей.....	176
4.5. Сетевой подход к оценке трудоемкости алгоритмов.....	180
4.6. Оценка дисперсии трудоемкости алгоритмов.....	187
5. Облачный управляющий процесс, его математические модели и его моделирование.....	191
5.1. Основные понятия и определения.....	191
5.2. Формальное писание и преобразование процессов.....	198
5.2.1. Исчисление высказываний как язык описания облачных процессов.....	198
5.2.2. Исчисление предикатов как язык описания облачных процессов.....	214
5.2.2.1. Введение в исчисление предикатов.....	214
5.2.2.2. Адаптация в процессе логического вывода.....	223
5.2.2.3. Логические алгоритмы планирования поведения робота.....	229
5.2.2.4. Алгоритмы распознавания ситуаций.....	235
5.2.2.5. Моделирование внешней среды.....	243
5.2.2.6. Алгоритмы построения программных движений.....	245
5.2.2.7. Алгоритмы адаптивного управления движением.....	248
5.2.3. Описания облачных процессов сетями Петри.....	250
5.2.3.1. Сети Петри и их модификация.....	250
5.2.3.2. Облачные управляющие процессы и их формализованное описание.....	259
5.2.3.3. Обобщенная сеть Петри для описания неавтономного облачного управляющего процесса.....	271
5.2.3.4. Получение правильного облачного управляющего процесса.....	276
5.2.3.4.1. Граф достижимых маркировок сети Петри.....	276
5.2.3.4.2. Влияние структуры облачного управляющего процесса на наличие тупиковых состояний.....	278
5.2.3.4.3. Тупиковые состояния, вызываемые разделением функциональных ресурсов.....	284
5.3. Облачные управляющие алгоритмы и языки их описания.....	290
5.3.1. Последовательный облачный управляющий алгоритм и его свойства.....	290
5.3.1.1. Логические схемы алгоритмов.....	290
5.3.1.2. Матричные схемы алгоритмов и их связь с логическими схемами.....	294

5.3.2. Преобразование логических схем алгоритмов.....	298
5.3.2.1. Минимизация числа логических условий.....	298
5.3.2.2. Учет распределения сдвигов при минимизации ЛСА.....	302
5.3.3. Объединение ЛСА.....	307
5.3.4. Описание параллельных облачных управляющих алгоритмов.....	317
5.3.5. Переход от правильного облачного управляющего процесса к облачному управляющему алгоритму.....	321
5.4. Реализация облачного управляющего алгоритма.....	332
5.4.1. Принципы реализации параллельного облачного управляющего алгоритма.....	332
5.4.2. Многопрограммное управление.....	344
5.4.3. Преобразование управляющего алгоритма при его реализации в многопрограммном УС.....	349
5.4.4. Программно-аппаратурная реализация управляющего алгоритма.....	354
6. Технические системы как средства реализации облачных технологий.....	357
6.1. Сущность технической системы.....	357
6.2. Модель технической системы.....	362
6.3. Функциональная структура технической системы.....	367
6.4. Органоструктура технической системы.....	371
6.5. Конструктивная схема технической системы.....	376
6.6. Сравнение структур технических систем и их преобразование.....	382
6.7. Граница технической системы и их окружение.....	385
6.8. Классификация технических систем.....	387
7. Свойства технических систем.....	403
7.1. Общие признаки технических систем.....	405
7.2. Категории свойств технических систем.....	406
7.3. Отношения между свойствами.....	441
7.4. Установление свойств и их отношений.....	445
7.5. Перечень требуемых свойств технической системы.....	449
8. Оценивание и представление технических систем.....	456
8.1. Выбор критериев и проведение оценки.....	456
8.2. Представление технических систем.....	463
8.3. Этапы создания и использования технической системы.....	467
8.4. Развитие технических систем.....	489
8.5. Специальные теории технических систем и требования к ним.....	506
9. Облачные управляющие системы как объекты автоматизированного проектирования.....	509
9.1. Задачи и средства проектирования ОУВС ГАПС.....	511

9.1.1. Распределенные ОУВС и организация программного управления ГАПС.....	511
9.1.2. Задачи системотехнического проектирования ОУВС ГАПС....	529
9.1.3. Распределенная облачная управляющая вычислительная система автоматизированного производства как объект системотехнического проектирования.....	534
9.1.4. Средства автоматизированного системотехнического проектирования ОУВС и их место в САПР ГАПС.....	540
9.2. Сетевые модели проектирования облачных управляющих вычислительных систем.....	549
9.2.1. Изобразительные возможности сетевых моделей.....	549
9.2.2. Нагруженная сеть как графический язык высокого уровня.....	569
9.2.3. Уравнения состояний, динамика сети Петри и анализ качественных характеристик систем.....	573
9.2.4. Операции над сетями и их назначение.....	590
9.2.5. Сетевое представление и автоматизация подготовки алгоритмов распределенной обработки и управления.....	596
9.2.6. Математическое обеспечение для расчета сетей с очередями как моделей ОУВС И ГАПС.....	604
9.3. Методы анализа и оптимизации процессов в облачных управляющих вычислительных системах.....	616
9.3.1. Взаимодействие процессов в мультипрограммных ОУВС и средства-советчики САПР.....	616
9.3.2. Агрегирование структур и слияние потоков — основа для ускорения проектных расчетов.....	625
9.3.3. Балансирование рабочей нагрузки и параметров ОУВС.....	630
9.3.4. Анализ и оптимизация ОУВС с общими шинами, динамической памятью и другими пассивными или одновременно захватываемыми ресурсами.....	638
9.3.5. Оптимизация размещения файлов и вычислительных работ в локальных управляющих сетях.....	648
9.4. Реализация программного обеспечения ОУВС.....	655
9.4.1. Безбумажная Р-технология моделирования и подготовки распределенного управления в диалоговой среде.....	655
9.4.2. Особенности реализации программного обеспечения систем имитации и управления процессами ГАПС.....	663
9.4.3. Средства САПР на базе микропроцессорной техники	673
Приложение.....	676
Литература.....	692

Введение

Современная научно-техническая революция изменила представление о масштабах и темпах развития науки, техники и технологий. Важными ее особенностями являются широкий охват революционными преобразованиями самых разнообразных сфер, и явлений, сокращение цикла реализации научно-технических достижений, интегративный характер развития науки, техники, технологий и производства.

Революционный взрыв, происходящий ныне в сфере науки, техники и технологий, ее промышленном применении, подготовлен многовековой историей их развития, историей, сочетающей эволюционные этапы роста и скачки.

Совершенствование технологического процесса, производительных сил общества непосредственно связано с развитием средств труда, техники и технологий. Развитие технологий и производства вызывает необходимость совершенствования техники. В свою очередь, прогресс техники обуславливает совершенствование всех элементов технологического и производственного процесса.

Технический прогресс, особенно в настоящее время, немислим без прогресса в развитии науки. Наука, как техника и технологии, стала непосредственно производительной силой общества.

Фундаментам научно-технического прогресса являются научные исследования и технологии. Научно-технический прогресс основывается на познании объективных законов развития природы и использовании их для активного на нее воздействия.

Наука как совокупность знаний о природе и обществе, накопленных в ходе познания законов их развития, является важнейшим, первичным источником развития производительных сил. В становлении и создании науки логическое развитие проблем и понятий не является самоцелью. Определяющей в развитии науки является потребность в совершенствовании имеющихся и создании новых видов техники и технологий, потребность развития материального производства, непроизводственной сферы и общества в целом.

Научный прогресс заключается во все более точном и глубоком познании действительности на основе обобщения имеющихся знаний, предвидения развития научных проблем, техники, технологий, общественных явлений.

Элементом научно-технического прогресса является процесс совершенствования техники и технологий. Техника, представляющая собой совокупность средств труда, с помощью которых люди воздействуют на предметы труда для изготовления изделий, удовлетво-

ряющих те или иные потребности человека, является одной из составных частей производительных сил общества, производственного процесса.

Технология, представляющая способ преобразования вещества, энергии, информации в процессе изготовления продукции, обработки и переработки материалов, сборки готовых изделий, контроля качества, управления. Технология воплощает в себе методы, приемы, режим работы, последовательность операций и процедур, она тесно связана с применяемыми средствами, оборудованием, инструментами, используемыми материалами. **Совокупность технологических операций образует технологический процесс**. Современная наука использует термин "технология" и в таких сочетаниях, как "технология обучения, образовательного процесса, лечения, управления"

Понятие «техника» уже понятий «средства труда» и «средства производства». Техника охватывает ведущую, активную, наиболее сложную часть средств труда и, соответственно, средств производства. На заре развития человеческого общества, когда еще не было науки, а люди лишь обладали определенным опытом, некоторыми отрывочными знаниями, техника представляла собой *овеществленный опыт*. В дальнейшем, по мере накопления знаний о законах развития природы, возникновения науки как качественно новой ступени человеческого познания, появляются более совершенные технические и технологические средства, представляющие собой *кристаллизованные научные познания*.

Научно-технический прогресс взаимообусловлен и взаимосвязан с прогрессом производства: его простых элементов (предметов и средств труда, рабочей силы) и форм их соединений (технологических методов и форм организации и управления производством).

Совершенствование предметов и средств труда, а также технологических процессов проявляется прежде всего в следующих направлениях технического прогресса: в механизации, автоматизации, электрификации и химизации производства.

При современном уровне развития общества проблемы научно-технического прогресса нельзя рассматривать только в границах производственной сферы деятельности людей. Часть технических средств находит широкое применение и в отраслях непроизводственной сферы (культурно-бытовое обслуживание населения, государственное управление, оборона и др.).

Исследование развития техники и технологий в производственной и непроизводственной сферах показывает, что ведущее место занимает техника в производственных отраслях, однако неуклонный рост потребностей и их дифференциация обуславливают тенденцию

постоянного увеличения выпуска техники и совершенствования технологий, используемой в непроизводственной сфере. Научно-технический прогресс в непроизводственных отраслях народного хозяйства, в быту, улучшающий условия труда, отдыха и обслуживания трудящихся, косвенно также влияет на рост общественной производительности труда.

Трудно переоценить значение техники и технологий, широко внедряемых в сферу управления производством. Совершенствование управления тесно связано с развитием электронно-вычислительных машин, организационной техники, аппаратуры связи. Современная техника и технология управления дают возможность резко повысить эффективность процесса управления и, как следствие, эффективность процесса производства.

Одним из ценнейших качеств homo sapiens является его способность в процессе познания выделять главное и существенное, оставляя при этом без внимания несущественные или случайные признаки, что необходимо для правильного и глубокого понимания объекта познания. Метод абстрагирования помогал человечеству получать и упорядочивать знания в любой области его деятельности и тем самым способствовал возникновению и развитию отдельных наук.

Тем не менее и до сих пор для решения той или иной проблемы не всегда имеется соответствующая теория. В области техники найдется немало примеров того, как практика опережает теорию, а развитие теории впоследствии позволяет улучшать достигнутые практические результаты. Так обстоит дело и с **теорией облачных технологий**.

Еще сравнительно недавно из практических соображений все вопросы, связанные с планированием, производством и эксплуатацией, решались отдельно в рамках каждой отрасли. Вследствие этого возникли обособленные друг от друга сферы знаний и профессий, в которых всякое профессиональное обучение должно было дополняться многолетним опытом. Причина этого заключалась в отсутствии теории и, как следствие, в отсутствии системы сбора и классификации изобретений и сущности новых технологий.

Приведем изложение концепции нашего понимания теории облачных технологий.

1. Цель теории. Цель теории состоит в том, чтобы привести, имеющиеся знания по объекту теории — облачным технологическим

системам — в единый комплекс понятий, определений и положений, основываясь на сущности и закономерностях структуры, создания и использования облачных технологических систем, а не на отдельных эмпирических данных, относящихся к этим системам. В то же время должна быть установлена целесообразная система понятий, позволяющая, во-первых, понять их смысл без дополнительных пояснений и, во-вторых, выводить из них другие понятия. Поскольку теоретические исследования в области облачных технологических систем находятся на начальной стадии, такая дедуктивная форма теории может быть получена лишь постепенно.

2. Структура теории. Структура теории должна содержать основные положения, определяемые более детально в рамках этой теории, такие, как

- система понятий;
- система преобразований;
- технологический процесс как элемент системы преобразований;
- облачная технологическая система (CTS) как элемент системы преобразований;
- назначение CTS;
- структура CTS ;
- свойства и оценивание CTS;
- возникновение и развитие CTS;
- эволюция CTS;
- систематика — классы, типы и виды CTS;

3. Виды теории. В соответствии с областью применения различают:

- общую теорию облачных технологических систем, которая справедлива для всех локальных облачных технологических систем;
- специальные теории, которые конкретизируют общую теорию для отдельных классов, типов или видов облачных технологических систем.

Структура специальной теории также может быть иерархической (например, теория облачных технологических систем в области техники, теория облачных технологических систем в области транспорта, теория облачных технологических систем в области воздушного транспорта, теория облачных технологических систем в области самолетостроения). Особое положение занимают специальные теории, которые применимы для нескольких отраслей.

г) *Взаимосвязь теории с другими дисциплинами.* Теория облачных технологических систем основывается на целом ряде

научных дисциплин, число которых возрастает по мере включения в нее новых систем и установления требований к ним. Наряду с такими «классическими» науками, как физика (со всеми ее направлениями) и химия, все в большей мере вовлекаются в рассмотрение биология и такие дисциплины, как теория систем, информатика, экономика, эргономика, логика и др.

С другой стороны, теория облачных технологических систем образует некоторые рамки и вводит определенный порядок во многие дисциплины, связанные с проектированием, конструированием, изготовлением, испытаниями, сбытом, хранением, транспортировкой, вводом в действие или ликвидацией систем. В этих дисциплинах положения общей теории облачных технологических систем в том или ином смысле «детализируются».

Рассмотрим некоторые примеры. Наука о сопротивлении материалов исследует связи между прочностью технической системы, с одной стороны, и геометрическими характеристиками, свойствами материала и нагрузками технической системы, с другой; аналогично надежность, срок службы, технология изготовления в соответствующих теориях основываются на некоторых частных положениях общей теории. В теории подъемно-транспортных машин изучаются процессы транспортировки, и, таким образом, с позиции теории технических систем она также является некоторой специальной теорией технологических процессов. В теории механизмов и машин рассматриваются механизмы как часть технической системы; эта теория также является специальной теорией технических систем.

д) *Применение теории облачных технологических систем.* Кроме практического применения, теория облачных технологических систем должна иметь также познавательное значение. Разработка некоторой системы объектно-ориентированных дисциплин (для отдельных областей) позволяет установить ясные взаимосвязи и границы между частными дисциплинами и ввести определенное упорядочение. С учетом этого теория облачных технологических систем важна для создания общей картины в различных областях и будет способствовать совершенствованию дисциплин, где она может и должна служить в качестве обобщающей теории.

В заключение приведем еще несколько соображений относительно целесообразности создания теории облачных технологических систем и покажем некоторые преимущества объединения системных теорий.

1. Теория выявляет закономерности, справедливые для всех рассматриваемых объектов. Она способствует перенесению профессионального опыта из одной области в другие благодаря

возможности переноса системных категорий (использование гомоморфизма рассматриваемых объектов).

2. Объединение всех рассматриваемых объектов в класс «облачные технологические системы» позволяет разработать подход к практической деятельности, не связанный с конкретным объектом и приемлемый во всех специальных областях. В рамках этого подхода можно изучать и разрабатывать методы проектирования облачных технологических систем вообще и систем определенного класса в частности.

3. Работа с абстрактными понятиями заставляет исследователей и разработчиков применять научные методы там, где силы воображения и опыта недостаточно. Тем самым создаются условия для того, чтобы отойти от устаревших традиций и шаблонов.

4. Теория облачных технологических систем позволяет трактовать любую рассматриваемую проблему целостно, с позиций системного подхода. Такой подход является предпосылкой эффективного проектирования и успешного выполнения других работ.

5. Использование кибернетики и ее понятий позволяет улучшить связи инженеров с учеными. Кроме того, при этом облегчается формализация некоторых операций в процессе проектирования, поскольку расширение применения вычислительных устройств требует построения алгоритмов логических операций. Таким образом, теория облачных технологических систем связана с развитием автоматизированного проектирования.

6. Формирование классов облачных технологических систем, основанных на аналогичности отношений, дает специалистам базу для выявления максимального количества способов реализации определенной функции или определенного отношения. Тем самым создаются предпосылки для того, чтобы из множества возможных решений выбрать наилучшие. Практической формой представления такой информации является создание специализированных каталогов объектов.

Теория облачных технологических систем дает в руки специалистам подход, ориентирующий на конечную цель и позволяющий видеть взаимосвязи, понимать и применять целостность как принцип, а также распознавать в различных объектах существенные аналогии и отношения.

1. Предмет, задачи и методы теории облачных технологических систем

1.1. Базовые понятия и определения

Обычно для выражения своих мыслей люди пользуются интуитивно выбираемыми словами и словосочетаниями разговорного языка. Однако интуитивный подход для построения терминологии научной дисциплины неприемлем, так что приходится устанавливать границы применимости и точный смысл каждого слова или выражения в рамках данной научной или специальной области. При этом одни понятия используются только в узкоспециальных областях. Другие понятия, часто выражаемые общеупотребительными словами, применяются в различных смыслах, причем часто их значения близки к обиходным, но иногда могут иметь значение, совершенно отличное от общепринятого (например, такие технические термины, как «журавль», «баба»)

Другая проблема связана с выбором наиболее точных названий для таких понятий, т. е. терминов. Так, даже отыскание общего выражения для понятия «технологически й продукт» является сложной задачей. Здесь всегда нужно прислушиваться к критике, особенно со стороны тех, кто уже рассматривал аналогичные проблемы, тем более если при известных условиях было выбрано другое выражение для обозначения аналогичного содержания. Риск неудачи здесь тем меньше, чем тщательнее и объективнее проведены сопоставление имеющихся данных, их обсуждение и необходимая унификация. Для облачных технологий это справедливо в особенности, поскольку здесь развитие шло и пока идет от практики к теории. В соответствии с установившейся традицией термины в технике чаще всего принимались интуитивно, без их точного определения. Например, термин «технология», являющийся основой целого ряда других понятий и терминов, имеет различное содержание в зависимости от специальной области, времени и места использования.

Терминологические трудности еще более усиливаются в связи с различиями смысла понятий в разных языках. Так, например, немецкий термин *Technik* (техника) не совпадает с английским *technique* (методика, технический прием, оборудование), а немецкое слово *Konstrukteur* (конструктор, строитель, создатель) не адекватно соответствующему английскому *designer* (конструктор, проектировщик, художник).

Даже в некоторых фундаментальных науках пока еще не достигнуто полное единство относительно некоторых терминов. Такое положение

наблюдается в кибернетике и теории систем — науках, которые имеют для нас основополагающее значение. Отсутствие единства по терминологическим вопросам не позволяет сослаться на соответствующую литературу и вынуждает рассматривать некоторые элементарные, но важные понятия.

В основу выбора используемых в данной книге названий для обозначения специальных понятий положены следующие принципы:

- широкое применение терминов в их укоренившемся значении, которое может быть лишь уточнено;
- ориентация в терминологическом плане на фундаментальные науки, такие, как математика, кибернетика и другие, с учетом того, что вводимые термины должны охватывать и область облачных технологий;
- применение, где это возможно, международной терминологии, что облегчает понимание на международном уровне.

При определении понятий мы использовали также многие уже принятые термины.

Кроме того, для различных понятий наряду с их определениями и названиями, будут рекомендованы также буквенные символы для их обозначения. Использование символов, с одной стороны, соответствует целям установления общепринятой терминологии, а с другой — позволяет сократить записи и затраты умственного труда. Однако решение о том, использовать ли такие рекомендации, предоставляется читателю. В тексте данной книги используется небольшое количество символов (например, техническая система — CTS, машинная система — MS); все остальные понятия будут записываться полностью, и никакой необходимости в их расшифровке не возникает.

Определение понятий будет осуществляться в два этапа. Сначала (в следующем разделе) будут даны определения наиболее важных основных понятий. Специальные понятия теории технических систем будут приведены позднее, в порядке обсуждения соответствующих тем. Для облегчения ориентации основные понятия обобщаются в соответствующие группы, например множество, система, тип системы и т. д. Отношения между этими отдельными понятиями в группах определяются последовательно. Например, при определении понятия «система» используется понятие «множество», следовательно, определение системы вытекает из определения множества.

Для определения понятий и установления терминов здесь будут использованы не все возможности. Так, например, мы не воспользуемся возможностями математической логики, исчисления высказываний и предикатов, несмотря на то что для наших целей понятия упомянутых областей знания были бы очень подходящими.

Множество

Множество — это совокупность наблюдаемых или мысленных объектов — *элементов* множества. По количеству элементов различают конечные и бесконечные множества. Если X — элемент множества M , то записывают $X \subset M$. Два множества M и N *эквивалентны*, если каждому элементу множества M точно соответствует элемент множества N и наоборот. Если все элементы множества N содержатся в M , то N — *подмножество* M , $N \subset M$. Совокупность всех не принадлежащих N элементов M называется *дополнением* множества N . *Объединение* $M \cup N$ — это множество, все элементы которого принадлежат либо M , либо N . *Пересечение* $M \cap N$ содержит все элементы, принадлежащие как M , так и N .

Элемент

Элемент — это минимальный неделимый объект. Свойства элемента — это его свойства как целого. Элемент можно использовать только как целое, поэтому недопустимо говорить о половине или четверти элемента. Неделимость элемента — это прежде всего удобное понятие, но не физическое свойство. Опираясь на понятие «элемент», исследователь оставляет за собой право перейти на другой уровень рассмотрения вопросов и говорить о том, из чего состоит элемент, а это свидетельствует о физической разложимости последнего. Таким образом, объекты называются элементами по соглашению, принимаемому с целью дать ответ на определенные вопросы, стоящие перед исследователями. Изменение вопросов может потребовать разложения элементов на составные части или объединения нескольких элементов в один.

Система

Системой мы называем совокупность, образованную (и упорядоченную по определенным правилам) из конечного множества *элементов*. При этом между элементами системы существуют определенные *отношения*. Возможны также системы, включающие изолированные элементы (или группы элементов), которые не имеют отношений с другими элементами системы. Систему определяют и как совокупность связанных элементов, объединенных в одно целое для достижения определенной цели. Здесь

под целью понимается совокупность результатов, определяемых назначением системы. Наличие цели и заставляет связывать элементы в систему. **Целостность** — наиболее важное свойство системы. Элемент принадлежит системе потому, что он связан с другими ее элементами, так что множество элементов, составляющих систему, невозможно разбить на два и более несвязанных подмножества. Удаление из системы элемента или совокупности элементов непременно изменяет ее свойства в направлении, отличном от цели. Искусственные (технологические) системы описывают путем определения их функций и структур.

Элемент и система являются *относительными* понятиями. Элемент может одновременно являться системой меньших элементов, а система в свою очередь может быть элементом некоторой большей системы. Например, некоторая облачная технология — это система, образованная своими элементами, и в то же время эта же облачная технология может быть элементом некоторой вышестоящей облачной технологии. Система может быть разделена на подсистемы различной сложности. Таким образом, система может изучаться с *различных точек зрения*, зависящих не только от специфики системы, но и от возможностей органов чувств или технических средств. Так, например, при проектировании облачных технологических систем не рассматриваются молекулы или атомы; самым нижним уровнем рассмотрения в этом случае являются операции, каждая из которых в свою очередь может рассматриваться как совокупность (система) воздействий более простой формы — прием или переход. Объектом рассмотрения для конструктора являются машины, для проектировщика — даже предприятия. В отличие от этого, например материаловед свои исследования проводит на уровне молекул как элементов системы (материала).

Понятие «система» находится в одном ряду с такими понятиями, как назначение, поведение, структура, окружение, вход, выход, свойство, состояние.

Техника, представляющая собой совокупность средств труда, с помощью которых люди воздействуют на предметы труда для изготовления изделий, удовлетворяющих те или иные потребности человека, является одной из составных частей производительных сил общества, производственного процесса.

Назначение системы

Всякая искусственная система имеет определенное *назначение*, которое может быть описано системой целей. Цель — это некоторое (возможно, воображаемое) положение дел, к осуществлению которого стремятся. Тогда система целей может быть определена как множество целей и отношений между ними. Подцель может конкретизировать цель. Зачастую подцель является средством достижения цели.

Функционирование системы

Поведение может быть определено как множество последовательных во времени *состояний* (см. далее) системы. Поведение биологических систем трактуется как сумма реакций на раздражения. Для некоторых типов систем (таких, как системы понятий и целей) понятие «поведение» не имеет смысла. Целью создания облачных технологических систем является вполне определенное их поведение. Целенаправленное поведение системы часто называют *функцией*. В этом случае под функцией понимают некоторую стабильную способность к определенным действиям, что обеспечивается лишь правильным поведением системы, так как, вообще говоря, система может функционировать неправильно. «Поведение» облачных технологических систем мы будем называть функционированием и применять этот термин в связи с желательным действием. *Функция системы* — это правило получения результатов, предписанных целью (назначением) системы. Определяя функцию системы, ее поведение описывают с использованием некоторой системы понятий — отношений между переменными, векторами, множествами и т. п. Функция устанавливает, что делает система для достижения поставленной цели безотносительно к физическим средствам (элементам, связям), составляющим саму систему, и не определяет, как устроена система. Системы изучают на разных уровнях абстракции, с использованием различных подходов, каждый из которых дает ответ на определенные вопросы. В связи с этим функции системы могут описываться с разной степенью детализации. Для описания функций систем используются теории множеств, алгоритмов, случайных процессов, информации и т. д. Функционировать — значит реализовать функцию, т. е. получать результаты, предписанные назначением системы.

Структура системы

Понятие *структура* (Str) характеризует внутреннюю организацию, порядок и построение системы. Таким образом, структура — это совокупность элементов и отношений между ними.

Если $E = \{e_1, e_2, \dots, e_n\}$ есть множество элементов, а $R = \{r_1, r_2, \dots, r_n\}$ — множество отношений, то структура $Str = \{E, R\}$ представляет собой множество, состоящее из E и R . Один и тот же объект может быть определен несколькими системами и, следовательно, несколькими структурами. Так, например, для человеческого тела можно определить костную систему (структуру костей и связей между ними), систему кровообращения, нервную систему, систему пищеварения и т. д. Структура наряду с функционированием является наиболее важным свойством системы. В нашем понимании термин «структура» отличается от соответствующего термина в философии, где он используется для обозначения только множества отношений в системе. *Структура системы* — это фиксированная совокупность элементов и связей между ними. В общей теории систем под структурой принято понимать только множество связей между элементами, т. е. структура понимается как картина, отображающая только конфигурацию системы безотносительно к составляющим ее элементам. Такое толкование ее понятия удобно при структурном подходе к изучению свойств различных систем — систем с параллельными, последовательными, иерархическими структурами, обратными связями и т. п. На практике в понятие «структура» включают не только множество связей, но и множество элементов, между которыми существуют связи. Этот смысл отражен в данном определении структуры. Наиболее часто структура системы изображается в форме графа: элементы системы представляются вершинами графа, а связи — дугами (ребрами) графа. **Граф — это математическая форма отображения структур.** Инженерной формой изображения структур систем являются схемы. **Схема и граф — понятия адекватные по содержанию, но различные по форме.** В схемах элементы и связи обозначаются любыми фигурами, удобными для применений.

Связь между функционированием и структурой системы.

Функционирование системы задается ее структурой. Относительно замкнутая система с заданной структурой функционирует однозначно, т. е. ее структура полностью определяет способ функционирования. С другой стороны, функционирование не определяет структуру

однозначно. Одна и та же функция может быть реализована различными структурами.

Организация

Организация — это явление аранжировки с целью осуществления определенных функций в системах, состоящих из большого числа элементов. Обычно к одной и той же цели можно прийти различными способами, исходя из различных принципов организации систем.

Каждый принцип организации задает определенный способ построения множества систем, аналогичных по назначению, но различных по функциям и структурам. Конкретная система представляет собой лишь пример реализации некоторого способа организации. Например, подавляющее большинство современных ЭВМ строится на основе одного принципа организации — принципа программного управления реализацией алгоритма на основе команд, имеющих операционно-адресную структуру. Таким образом, **организация — понятие более высокого ранга, чем функция и структура; организация — это модель, на основе которой могут строиться многие конкретные системы.**

Если речь идет о способе порождения функций, достаточных для достижения определенной цели (определенных результатов), то используется термин *функциональная организация*. Если же речь идет о наборе элементов и способе их соединения в структуру, обеспечивающую реализацию функций определенного класса, то используется термин *структурная организация*. Определяя некоторый способ функциональной организации, выявляют класс функций, присущих системам определенного назначения (безотносительно к средствам, необходимым для реализации этих функций), а определяя способ структурной организации, выявляют правило построения структур, реализующих некоторый класс функций, т. е. отвечающих некоторому назначению.

Окружение системы

Окружение (окружающая среда, *Umg*) системы теоретически включает все, что не входит в данную систему. Практически же мы ограничимся окружением, состоящим из систем, включающих хотя бы один элемент, выход которого является в то же время входом (определения этих понятий будут даны позднее) некоторого элемента системы, либо

элемент, вход которого является одновременно выходом некоторого элемента системы. Такое «непосредственное» окружение будет называться *реальным окружением*. Полное окружение системы включает следующие составные части: геосфера, атмосфера, биосфера (включая людей), техносфера и астросфера.

Вход и выход системы

Вход (In) представляет внешнее отношение окружающая среда→система. Входная величина может быть в зависимости от вида системы действием, связью (отношением) или параметром состояния объекта действия (операнда). Совокупность всех входов составляет обобщенный вход (который может быть представлен как вектор отдельных входов).

Выход (Ou) представляет внешнее отношение система → окружающая среда. Выходная величина может быть в зависимости от вида системы действием, связью или параметром состояния операнда. Совокупность всех выходов может быть сведена к обобщенному выходу (вектору выхода). Выход системы есть множество выходов всех элементов, которые не являются входами других элементов системы. Входная и выходная величины являются единственными связями системы с окружающей средой. Входы и выходы системы включают все виды связей с окружающей средой: желательные и нежелательные (помехи), связи материального (S), энергетического (*En*) и информационного (*I*) характера. Необходимо заметить, что на практике отсутствует единство мнений относительно использования понятий входа и выхода. Одни авторы рассматривают вход и выход соответственно как входные (рецепторы) и выходные (эффекторы) элементы системы. В этом случае через них проходят воздействия на систему (влияние окружения) и проявляются ее реакции (влияние системы на окружение). Для других авторов вход и выход — это соответственно то, что поступает в систему и выходит из нее, т. е. то же самое, что пища и отходы для биологических систем. Подобные толкования понятий «вход» и «выход» основаны на рассмотрении целей и представлений, исходя из которых создается система. Далее мы еще раз вернемся к подробному рассмотрению понятий «вход» и «выход».

Свойства системы и их оценки

Каждая система, ее элементы и отношения обладают *свойствами* (*Е*), присущими этой системе и точно ее определяющими, такими как размеры, масса, скорость, форма, стабильность, а также технологичность, транспортабельность и особенно способность что-либо делать, т. е. функционировать.

(Необходимо сделать замечание относительно терминологии в этой области. В исчислении предикатов со свойствами связаны соответствующие однозначные предикаты. В языке предикаты используются для описания свойств. В философии существенные свойства называются атрибутами. Для характеристики существенных свойств технологических объектов используется понятие «параметр», определяющее в некотором отношении функцию соответствующего объекта.)

Свойством, является всякий существенный признак объекта.

Объектов без свойств не существует. Однако степень воплощения этих свойств может быть различной. Например, одним из свойств людей является рост. Если рост определенного человека 170 см, то можно сказать, что мера этого свойства (роста) равна 170 см (при использовании сантиметра как единицы измерения). В этой связи мы сталкиваемся с проблемой количественного определения (измерения, квантификации) свойства .

Для совокупной характеристики объекта, например при его оценке, выбирают существенные свойства этого объекта. В этих случаях говорят о *частной*, *обобщенной* и *совокупной* оценках, обобщенном *качестве* или *ценности* (смысл понятий, применяемых здесь, отличается от их смысла, например, в философии или психологии). Для получения совокупной оценки необходимо измерить отдельные свойства, а частные оценки превратить в обобщенные.

Состояние системы

Совокупность значений свойств системы в определенный момент времени называется *состоянием* системы. Аналогично качеству состояние системы можно определить вектором, имеющим в качестве компонентов отдельные свойства. При определении качества или состояния абстрагируются от большей части несущественных или не представляющих интереса свойств.

Два состояния системы могут быть одинаковыми или различными. Различие между состояниями называется их *разностью*. Разность

возникает при переходе системы из одного состояния в другое. Разность может быть дифференциальной (когда имеет место непрерывный переход к следующему состоянию) либо дискретной.

Анализ

Анализ — это процесс определения свойств, присущих системе. Типичная задача анализа состоит в следующем. Известны функции и характеристики элементов, входящих в состав системы, и определена структура системы. Необходимо определить функции или характеристики, присущие совокупности элементов в целом.

Синтез

Синтез — это процесс порождения функций и структур, необходимых и достаточных для получения определенных результатов. Выявляя функции, реализуемые системой, определяют некоторую абстрактную систему, о которой известно только то, что она будет делать. В связи с этим этап синтеза функций называется *абстрактным синтезом*, а этап порождения структуры, реализующей заданные функции, — *структурным синтезом*.

Эффективность

Эффективность — это степень соответствия системы своему назначению. Из двух систем более эффективной считается та, которая лучше соответствует своему назначению. Оценка эффективности системы — одна из задач анализа систем.

Показатель эффективности

Показатель эффективности (качества)—это мера одного свойства (характеристики) системы. Показатель эффективности всегда имеет количественный смысл, т. е. является измерением некоторого свойства. По этой причине использование некоторого показателя эффективности предполагает наличие способа измерения (оценки) значения этого показателя. Для оценок эффективности систем могут применяться, например, такие показатели, как производительность, стоимость, надежность, габариты и т. п.

Критерий эффективности

Критерий эффективности — это мера эффективности системы. Критерий эффективности имеет количественный смысл и измеряет степень эффективности системы, обобщая все ее свойства в одной оценке — значении критерия эффективности. Эффективность систем, создаваемых для одной цели, оценивается на основе одного критерия, общего для этого класса систем. Различие в назначениях систем предполагает, что для оценки эффективности таких систем используются различные критерии. Если при увеличении эффективности значение критерия возрастает, то критерий называется *прямым*; если значение критерия уменьшается, то *инверсным*. Из двух систем более эффективной считается та, которой соответствует большее значение прямого критерия (меньшее значение инверсного критерия).

Оптимальная система

Оптимальная система — это система, которой соответствует максимальное (минимальное) значение прямого (инверсного) критерия эффективности на множестве мыслимых вариантов построения систем.

Модель системы

Модель, представленная на рис. 1.1, наглядно иллюстрирует приведенные выше определения и их взаимосвязи.

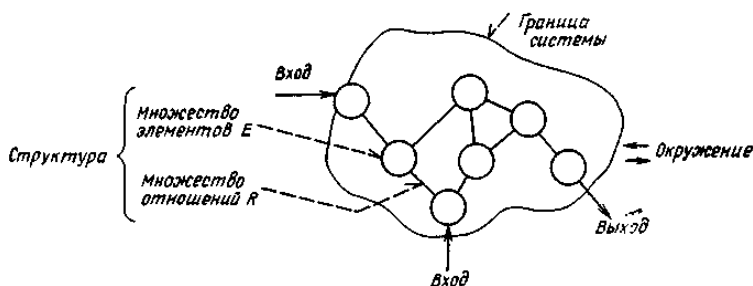


Рис. 1.1. Модель системы.

Модель — это физическая или абстрактная система, адекватно представляющая собой объект исследования.

Физические модели образуются из совокупности материальных объектов. Для их построения используются различные свойства, причем природа применяемых в модели материальных элементов не обязательно та же, что и в исследуемом объекте. Примером физической модели является макет.

Абстрактная модель — это описание объекта исследований на некотором языке. Абстрактность модели проявляется в том, что ее компонентами являются понятия, а не физические элементы [например, словесные описания, чертежи, схемы, графики, таблицы, программы (алгоритмы), математические описания]. Необходимое условие для перехода от исследования объекта к исследованию модели и дальнейшего перенесения его результатов на объект исследования — требование адекватности модели и объекта.

Адекватность предполагает воспроизведение моделью с необходимой полнотой всех свойств объекта, существенных для целей данного исследования. Понятие адекватности — весьма широкое и основывается на строгих в математическом отношении понятиях **изоморфизма** и **гомоморфизма**. Две системы, в данном случае объект исследования и его модель, называются изоморфными, если между ними существует такое **взаимно-однозначное соответствие**, что соответствующие объекты различных систем обладают соответствующими свойствами и находятся в соответствующих отношениях друг с другом. В общем случае обеспечение изоморфизма модели и объекта исследования может быть не только трудновыполнимым, но и излишним, поскольку сложность модели при этом может оказаться настолько значительной, что никакого упрощения исследовательской задачи не произойдет. Гомоморфизм, так же как и изоморфизм, предполагает сохранение в модели всех определенных на объекте исследования свойств и отношений. Однако требование взаимно-однозначного соответствия заменяется требованием однозначного соответствия модели объекту, тогда как обратное соответствие — соответствие объекта модели — не однозначно.

Концептуальная модель

Концептуальная модель — это абстрактная модель, выявляющая причинно-следственные связи, присущие исследуемому объекту и существенные в рамках определенного исследования. Основное назначение концептуальной модели — выявление набора причинно-следственных связей, учет которых необходим для получения

требуемых результатов. Один и тот же объект может представляться различными концептуальными моделями, которые строятся в зависимости от цели исследования. Так, одна концептуальная модель может отображать временные аспекты функционирования системы, другая — влияние отказов на работоспособность системы и т. д.

Математическая модель

Математическая модель — это абстрактная модель, представленная на языке математических отношений. Математическая модель имеет форму функциональных зависимостей между параметрами, учитываемыми соответствующей концептуальной моделью. Эти зависимости конкретизируют причинно-следственные связи, выявленные в концептуальной модели, и характеризуют их количественно.

Моделирование

Моделирование — это процесс представления объекта исследования с помощью ему моделью и проведения экспериментов с моделью с целью получения информации об объекте исследования. При моделировании модель выступает и как средство, и как объект исследования. Моделирование является косвенным методом изучения объекта в том смысле, что исследование производится не над самим объектом, а над представляющей объект системой — моделью. Отличительная особенность моделирования как метода исследования заключается в возможности исследования таких объектов, прямой эксперимент с которыми трудно-выполним, экономически невыгоден или вообще невозможен.

Типы систем

Используя различные критерии, можно установить большое количество типов систем. Системы можно классифицировать следующим образом.

- а) По положению системы в иерархии:
— надсистема, система, подсистема.
- б) По связям с окружением:
— открытые (с определенным окружением, т. е. по крайней мере с одним входом или выходом);

- замкнутые (без связей с окружением).
- в) По изменению состояния:
 - динамические (состояние изменяется во времени);
 - статические (состояние не изменяется во времени).
- г) По характеру функционирования:
 - детерминированные (в зависимости от состояния системы можно однозначно судить о ее функционировании);
 - стохастические (можно только высказать предположение относительно различных возможных вариантов функционирования).
- д) По типу элементов (в смысле их конкретности):
 - конкретные (элементами являются реальные объекты);
 - абстрактные (элементами являются отвлеченные объекты).
- е) По происхождению системы:
 - естественные (созданные природой);
 - искусственные (созданные людьми).
- ж) По характеру зависимости выходов:
 - комбинаторные (выход зависит только от входа);
 - секвентивные (выход зависит от входа и других величин).
- з) По степени сложности структуры:
 - предельно сложные (например, мозг, народное хозяйство);
 - очень сложные (например, полностью автоматизированное предприятие, производственный комплекс);
 - сложные (например, легковой автомобиль, библиотека университета);
 - простые (например, семейная библиотека, болтовое соединение).
- и) По виду элементов:
 - системы типа «объект» (элементами являются предметы, например дом, двигатель, машина);
 - системы типа «процесс» (элементами являются операции, например изготовление, фильтрация, перегонка, приготовление пищи).

Типы задач

В связи с системами рассматриваются три характерных типа задач.

Задача *синтеза* — заданы характер функционирования и другие требования к системе, определить структуру, которая удовлетворяет поставленным требованиям.

Задача *анализа* — задана структура, определить функционирование системы.

Задача «черного ящика» — заданы система, структура которой неизвестна или известна частично, определить ее функционирование и, возможно, структуру.

Символическое представление систем

Символически систему мы будем изображать четырехугольником, кругом или их комбинацией, используя для систем типа «объект» (TS) и систем типа «процесс» (P) различные символы (рис. 2.2).

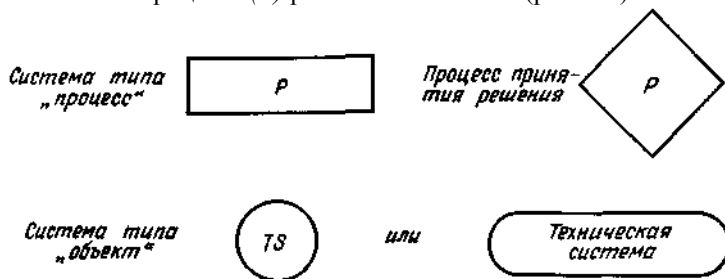


Рис. 1.2 Графическое обозначение двух типов систем

Система типа «объект» не требует особых пояснений, поэтому мы остановимся более подробно на системах типа «процесс».

Система типа «процесс»

Вообще говоря, термин «процесс» означает, что что-то происходит, совершается, т. е. изменяется с течением времени. В природе нескончаемо что-нибудь происходит. Естественным изменениям, т. е. таким процессам, как старение, выветривание, эрозия, подвержены даже такие объекты, которые нам кажутся очень стабильными, неизменными, например скалы и горы. То же самое относится и к процессу существования живого существа.

Наряду с естественными процессами человек организует искусственные процессы с целью осуществления необходимых или желательных для него изменений. Такие изменения служат удовлетворению человеческих потребностей. Хотя человек и подчиняется законам природы, все же он может ускорить, усилить или улучшить некоторые природные процессы или их свойства. Целенаправленное изменение определенных объектов имеет для людей жизненную важность. Искусственные процессы, в которых те или иные свойства объекта действия (операнда) претерпевают

соответствующие изменения при участии людей и технических средств, вследствие чего достигается желаемое состояние операнда, будем называть преобразованиями.

Термин «операнд» (*Od*) здесь выбран в качестве общего названия всех предметов, систем и состояний, подвергаемых целенаправленному преобразованию. Преобразование есть следствие определенных воздействий, основанных на физических, химических или биологических явлениях и описываемых некоторой инструкцией — **рецептом, алгоритмом, технологией**. Науками, исследующими преобразования в какой-либо определенной области, являются, например, термодинамика, технология производства. Воздействия на операнд выполняются *операторами*. Эти воздействия являются *выходами* операторов. На рис. 1.3 представлена общая модель процесса преобразования.

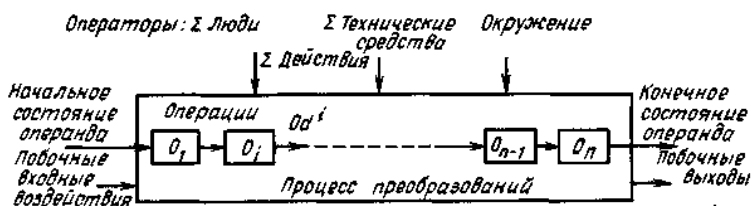


Рис. 1.3. Модель процесса преобразования.

Воздействия операторов осуществляются в виде потоков материи (*S*), энергии (*En*) и информации (*I*). Далее понятие преобразования будет рассмотрено более подробно.

Нужно пояснить еще одно важное понятие — *алгоритм*. Процесс преобразования представляет собой совокупность *операций* (*O*);

алгоритм—это однозначно определенная последовательность операций, которая либо устанавливается один раз заранее и действительна в течение всего процесса преобразований, либо меняется в зависимости от результата выполненной операции.

Таким образом, алгоритм можно определить аналогично **структуре процесса как упорядоченное множество операций, их отношений и условий перехода от одной операции к другой**. Значительное сходство имеется между понятиями алгоритма и технологического процесса, представляющего собой последовательность операций изготовления изделия или предоставления услуги.

Довольно типичными видами процессов в технике являются *управление и регулирование*.

Управление — это процесс в системе, посредством которого одна или несколько входных величин действуют требуемым образом на другие, считающиеся выходными.

Регулирование — это процесс, посредством которого некоторые изменяемые (регулируемые) величины непрерывно сопоставляются с эталонными (управляющими), причем на регулируемые величины оказывается воздействие с целью приведения соответствующих отклонений к нулю.

Технология

Технология - способ преобразования вещества, энергии, информации в процессе изготовления продукции, обработки и переработки материалов, сборки готовых изделий, контроля качества, управления. Технология воплощает в себе методы, приемы, режим работы, последовательность операций и процедур, она тесно связана с применяемыми средствами, оборудованием, инструментами, используемыми материалами. Совокупность технологических операций образует технологический процесс. Современная наука использует термин "технология" и в таких сочетаниях, как "технология обучения, образовательного процесса, лечения, управления"

Система типа «облачная технология»

Облачная технология - это различные аппаратные, программные средства, методологии и инструменты, которые предоставляются пользователю, как интернет-сервисы, для реализации своих целей, задач, проектов.

Типы отношений

Отношением (R) называется взаимозависимость или взаимодействие двух и более объектов либо явлений абстрактного или конкретного типа. При проектировании существенны объективные, определенные отношения, которые поддаются описанию в соответствии с физическими или логическими законами. Отношения связывают отдельные элементы в различных системах. Выражение «объект X находится в отношении R к объекту Y » символически обозначается $R(X, Y)$. Отношение может быть рефлексивным, симметричным или

транзитивным. Эти типы отношений можно охарактеризовать следующим образом:

- а) рефлексивность — каждый объект эквивалентен самому себе;
- б) симметричность — если один объект эквивалентен второму, то второй объект эквивалентен первому;
- в) транзитивность — два объекта эквивалентны между собой, если они по отдельности эквивалентны третьему.

Если выполняются все три условия, то отношение называется *отношением эквивалентности*. Отношение между двумя объектами будет также называться корреляцией. **Корреляция** — это математическая модель отношения в обобщенной форме.

Виды отношений

Подобие. *Подобие* — это отношение сходства между двумя или более системами (объектами, процессами, высказываниями), определяемое некоторыми общими свойствами. Вообще говоря, возможен диапазон степеней подобия от полного равенства (*идентичности*) до частного *сходства*. Можно говорить о функциональном, структурном и других видах подобия. Обычно подобие объектов понимается как одинаковость формы (но, как правило, не равенство по величине). Отношение подобия имеет большое значение при математическом и физическом моделировании. Законы подобия позволяют определить условия, при выполнении которых результаты модельных экспериментов справедливы для реальных условий. Например, течения газа или жидкости подобны при равных числах Рейнольдса. Область подобия может быть определена как пересечение множеств свойств, участвующих в данном отношении.

Аналогия. Соответствие существенных признаков, свойств, структур или функций объектов или явлений будем называть *аналогией*. Этот термин часто употребляется в том же смысле, что и подобие.

Гомоморфизм. Отношение между двумя системами, когда каждую составную часть и каждое отношение одной системы можно отобразить на некоторую составную часть и некоторое отношение второй системы (но не обратно), называется *гомоморфизмом*. В этом случае выполнение соответствующих условий подобия позволяет перенести результаты модельных экспериментов на натуру. Область подобия может быть определена как пересечение множеств свойств.

Изоморфизм. *Изоморфизм* называется отношение между двумя системами, когда каждой составной части одной системы может быть поставлена в соответствие определенная составная часть другой системы и наоборот (симметричность), а также, когда для каждого

отношения между двумя соответствующими составными частями имеется такое же отношение в другой системе и наоборот.

Идентичность. Это отношение между объектами или процессами, характеризующимися одинаковыми свойствами (признаками). При абсолютной идентичности должны быть одинаковыми все свойства, при относительной — только некоторые (в этом случае имеет место подобие).

Эквивалентность. Объекты или процессы называются эквивалентными, если между ними имеется отношение эквивалентности, т.е. **равноценности**. Эквивалентность полнее идентичности, так как для последней характерна только рефлексивность. Применительно к облачной технологии оба понятия будут использоваться как синонимы, т. е. под эквивалентностью будет подразумеваться абсолютная идентичность.

Математические функции. Важный класс отношений выражают *математические функции* как закономерные зависимости от переменной: $y = f(x)$. Такого рода математические функции выражают точно установленное отношение между x и y , т. е. детерминированную связь.

Причинность. Между причиной и вызванным ею действием существует асимметричное отношение. Причина вызывает действие. Существует строгая (детерминированная типа «если..., то») или ослабленная форма причинного отношения. Причинная цепь имеет место, если действие выступает в качестве причины дальнейших действий.

Связь. Если определенные выходы элемента (системы) одновременно являются входами какого-либо элемента (системы), то такого рода отношение называется *связью*.

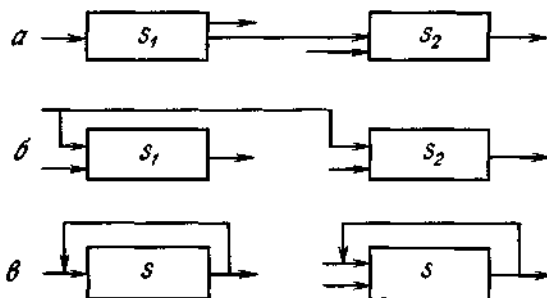


Рис. 1.4 Виды связей между системами
а—последовательная, б — параллельная, в — обратная и комбинированная связи

Связь может быть прямой (последовательной либо параллельной), обратной или комбинированной (рис. 2.4); она может быть материальной, энергетической или информационной.

Отношение цель — средство. Это — двухместное асимметричное отношение между системой целей (назначением, задачей) и средством их реализации.

Пространственное отношение. Отношение такого рода характеризует взаимное положение элементов отношения в пространстве.

Пространственные отношения изучаются в топологии.

Логическое отношение. Логическим отношением (в логике — двух- или многоместным предикатом) называется отношение между объектами типа « l_1 меньше, чем l_2 », или « l_3 находится около l_4 ».

Известными константами (функторами) являются: И; ИЛИ; И-ИЛИ; НЕ-ИЛИ; ТАК, ЧТО; ИЛИ-ИЛИ; ЕСЛИ-ТО; ТОЛЬКО ЕСЛИ-ТО; ТОЛЬКО ТОГДА-КОГДА; РАВНО. Из этого перечисления ясно, что многие описанные выше отношения являются также логическими отношениями. В ЭВМ реализация отношений такого рода осуществляется логическими элементами.

Временное отношение. Отношение такого рода описывает упорядочение процессов и событий во времени.

Выводы

1. Функционирование системы задается ее структурой.
2. Относительно замкнутая система с заданной структурой функционирует однозначно; функционирование полностью определяется структурой.
3. Функционирование не определяет структуру однозначно. Одна и та же функция может быть реализована различными структурами.

1.2. Предмет теории облачных технологий вычислительных систем

Предмет теории облачных технологий рассмотрим на примере облачных вычислительных систем.

Для выполнения облачных вычислений, необходимость в которых возникает в научной, инженерной, производственной деятельности, широко используются *электронные вычислительные машины* (ЭВМ). ЭВМ это комплекс аппаратуры, обеспечивающей выполнение набора операций над информацией, достаточного для реализации некоторого класса алгоритмов. Реализация любого алгоритма на ЭВМ требует выполнения определенной последовательности машинных операций, для порождения которой используется *программа*— алгоритм преобразования информации в терминах команд. Команды инициируют операции, выполняемые аппаратурой ЭВМ. Именно программа «настраивает» ЭВМ на получение требуемых результатов, а сама ЭВМ является только средством выполнения операций, предписанных программой. Таким образом, любое применение ЭВМ предполагает комплектацию машины набором программ, несущим в себе информацию о том, что должна делать машина. Система, состоящая из одной или нескольких ЭВМ и набора программ, обеспечивающего выполнение возложенных на систему функций, будем называть *облачной вычислительной системой* (ОВС).

Облачные вычисления (англ. *cloud computing*) — модель обеспечения удобного сетевого *доступа по требованию* к некоторому общему фонду конфигурируемых вычислительных ресурсов (например, сетям передачи данных, серверам, устройствам хранения данных, приложениям и сервисам — как вместе, так и по отдельности), которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами или обращениями к провайдеру.

Состав ОВС укрупненно представлен на рис. 1.5, а.

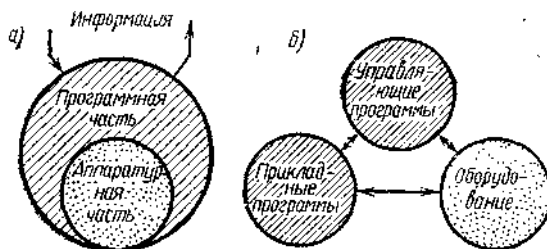


Рис. 1.5. Компоненты ОВС

Аппаратурная часть ОВС включает в себя совокупность устройств, входящих в состав одной ЭВМ или цифрового вычислительного

комплекса, состоящего из нескольких взаимосвязанных ЭВМ. Назначение аппаратурной части ЭВМ — выполнение операций ввода, хранения, преобразования и импорта информации. Программная часть ОВС — это набор программ, задающих порядок реализации возложенных на систему функций в терминах машинных команд, реализуемых аппаратурной частью системы. Программы управляют работой аппаратуры с целью получения результатов, соответствующих назначению системы. ОВС взаимодействует с внешней средой, получая из нее необходимую информацию и выводя информацию, «производимую» программами на основе вводимой информации. Аппаратурную часть ОВС можно рассматривать как пространство, предназначенное для выполнения набора операций над информацией. В этом пространстве развиваются вычислительные процессы, порождаемые программами. Инициирование программ производится либо в определенные моменты времени, либо сигналами извне и командами из других программ, а при возникновении особых ситуаций в самой ОВС - сигналами, формируемыми устройствами ЭВМ. ОВС как конкретный объект характеризуется:

- 1) множеством данных, вводимых в систему, и множеством результатов, формируемых на выходе системы;
- 2) функцией системы, устанавливающей правило отображения множества данных в множество результатов;
- 3) структурой системы, определяющей входы и выходы системы, номенклатуру устройств и связи между ними, номенклатуру алгоритмов и порядок взаимодействия алгоритмов между собой и с аппаратурой системы.

Первые две группы сведений характеризуют назначение ОВС, т. е. то, что должна делать система. Последняя группа сведений определяет способ внутренней организации ОВС, т. е. средства, которыми реализуется функция, возложенная на систему. Устройства, используемые в аппаратурной части системы,— это устройства ЭВМ: процессоры, каналы ввода— вывода, запоминающие и периферийные (внешние) устройства, устройства управления периферийными устройствами (контроллеры) и т. п.

Программное обеспечение включает в себя программы прикладные и управляющие. *Прикладные программы* представляют алгоритмы выполнения функций, реализация которых возлагается на систему, т. е. задают порядок преобразования исходных данных в совокупность результатов, для производства которых предназначается ОВС. Между программами существуют информационные связи, имеющие место, когда результаты, формируемые одной программой, являются данными для другой программы. Кроме того, для выполнения программ

используется общий набор устройств, поочередно распределяемых между программами. Из сказанного вытекает необходимость управления порядком выполнения программ, для чего в составе программного обеспечения выделяется набор управляющих программ. *Управляющие программы* обеспечивают необходимый порядок взаимодействия прикладных программ между собой и с оборудованием ОВС. Иначе говоря, управляющие программы распределяют вычислительные процессы в пространстве (между устройствами ЭВМ) и во времени. Порядок распределения вычислительных процессов в пространстве и во времени определяется *стратегией управления облачными вычислительными процессами* в ОВС. Стратегия управления задает алгоритм планирования работ в системе, который реализуют управляющие программы. Таким образом, внутреннюю организацию ОВС можно рассматривать как совокупность, состоящую из трех основных компонент: прикладных программ, управляющих программ и оборудования (рис. 1.5,б).

При создании ОВС необходимо, исходя из сведений о назначении системы, определить ее структуру и номенклатуру алгоритмов, наилучшим образом отвечающих назначению системы. Чтобы решать такого рода прикладные задачи, необходимо располагать знаниями о том, как влияют различные способы структурной организации ОВС и управления облачными вычислительными процессами на характер протекания последних в ОВС. **Свойства и закономерности, присущие облачным вычислительным процессам, протекающим в системах с различной организацией, составляют предмет теории ОВС.**

Теоретические исследования в отличие от экспериментальных базируются на использовании моделей реальных объектов. Модели, изучаемые в рамках теории ОВС, отображают объект исследований в форме, необходимой и достаточной для получения результатов, составляющих цель исследований. Исследуемые модели являются компромиссом между двумя противоречивыми тенденциями. С одной стороны, желательно, чтобы модель возможно полнее отображала в себе реальные процессы, с другой стороны, модель должна быть достаточно простой, чтобы можно было получить желаемые результаты. В теории ОВС облачные вычислительные процессы изучаются с такой степенью детализации, которую можно получить, если отдельные устройства ЭВМ рассматривать как элементы системы, а алгоритмы — как совокупности операторов, каждый из которых рассматривается как объект, порождающий определенное действие — ввод, передачу, преобразование или вывод информации. Другими словами, в моделях, изучаемых в теории ОВС, алгоритм

рассматривается как совокупность запросов к техническим ресурсам — памяти, средствам обработки, ввода и вывода информации, а оборудование ЭВМ — как совокупность объектов, обслуживающих эти запросы. Такими объектами и являются устройство ЭВМ — процессоры, устройства ввода и вывода, запоминающие устройства. Эта степень детализации систем в моделях достаточна для определения следующих важных характеристик систем: производительности, времени выполнения работ, загрузки отдельных устройств, интенсивности потоков информации и т. п.

Теория ОВС направлена на разработку методов синтеза оптимальных структур ЭВМ и стратегий управления облачными вычислительными процессами, причем структуры выявляются с точностью до устройств. Синтез устройств составляет предмет теории ЭВМ. Разработка программ, реализующих прикладные алгоритмы и стратегии управления облачными вычислительными процессами, проводится с использованием методов теории алгоритмов и аппарата программирования.

1.3. Классификация облачных вычислительных систем

Цель классификации ОВС — разделить множество мыслимых систем на классы подобных систем и исследовать свойства не каждой отдельной системы, а свойства каждого класса систем. Наиболее результативной считается такая классификация, которая приводит к максимальной унификации теоретических исследований и прикладных методов. Классификация проводится на основе совокупности признаков классификации, в качестве которых могут использоваться любые свойства систем.

Можно ожидать, что наиболее существенное влияние на организацию ОВС оказывает ее назначение, определяющее функции, которые должна реализовать система. Системы одного назначения, т. е. с одинаковой сферой применения, в меньшей степени различаются по своей структурной организации и программному обеспечению, чем системы с существенно различным назначением. По этой причине назначение системы можно рассматривать в качестве основного признака классификации ОВС. Классы ОВС, выделяемые на разных уровнях детализации назначения систем, приведены на рис. 1.6.

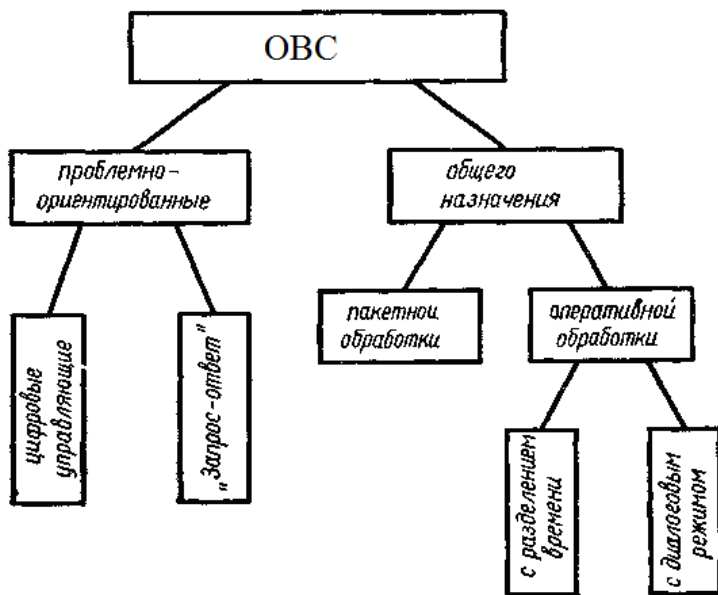


Рис. 1.6. Классификация ОВС по назначению

Все множество ОВС можно разделить на системы проблемно-ориентированные и общего назначения.

Проблемно-ориентированные ОВС. *Проблемно-ориентированной* называется ОВС, предназначенная для выполнения фиксированного набора алгоритмов, т. е. для решения конкретных задач. Примеры проблемно-ориентированных ОВС — системы управления технологическими процессами (например, гальваническими линиями), резервирования и продажи билетов, обработки банковских документов и т. п. Ориентация ОВС предопределяется конкретностью номенклатуры задач, для решения которых создается система. Задачи и алгоритмы их решения известны на момент создания ОВС и не меняются в течение всего периода эксплуатации системы.

Проблемно-ориентированные ОВС разделяют на системы цифровые управляющие и типа «запрос — ответ».

Назначение *цифровой управляющей системы* — управление работой конкретного технического объекта (ядерного реактора, коммутационного оборудования автоматической телефонной станции, прокатного стана и т. п.). Набор программ управления определяется задачей управления и неизменен во время работы системы. К тому же

программы должны выполняться в темпе, определяемом динамикой управляемого объекта. Сверхнормативная задержка в получении результатов, формируемых на выводе системы, может привести к катастрофическим последствиям или к снижению показателей качества работы управляемого объекта.

Системы «запрос — ответ» предназначены для обслуживания не технических объектов, а людей (операторов касс по продаже билетов, операторов банка, работников складов и т. д.). Запрос, формируемый на входе системы, является требованием инициировать определенную программу, формирующую ответ. Система должна обслуживать запросы по возможности быстрее, и это требование относится ко всей совокупности запросов, как к целому.

Следовательно, стратегия управления порядком выполнения программ в управляющих системах и системах «запрос — ответ» должна быть различна: в управляющих системах необходимо выполнять жесткие ограничения на время получения результатов, а в системах «запрос — ответ» необходимо стремиться по возможности уменьшать среднее время обслуживания запросов. К тому же системы «запрос — ответ» работают с большей номенклатурой программ и значительно большим количеством данных, чем управляющие системы. В связи с этим системы «запрос — ответ» содержат в своем составе большее количество устройств, характеризуются более сложной организацией программного обеспечения и требуют специфического подхода к их организации.

Проблемно-ориентированным системам присуще следующее свойство: темп выполнения программ подчиняется темпу процессов, развивающихся вне системы, т. е. подчиняется ходу реального времени. Работа систем этого класса должна быть организована таким образом, чтобы время реакции системы на входные воздействия не превышало нормативных значений, зависящих от динамических свойств объектов, обслуживаемых системой. Режим работы ОВС, при котором порядок функционирования ОВС подчиняется динамике процессов, развивающихся вне системы, называется работой в *реальном масштабе времени* (РМВ). Работа в РМВ — такое же специфичное свойство проблемно-ориентированных систем, как и постоянство набора выполняемых системой программ. В связи с этим проблемно-ориентированные ОВС иначе называют системами, работающими в РМВ.

ОВС общего назначения. ОВС общего назначения предназначены для решения прикладных задач, номенклатура которых не поддается строгому ограничению к моменту создания системы. ОВС общего назначения используются для выполнения расчетных работ в

организациях, занятых научными исследованиями, техническими разработками, планированием, статистическим учетом и т. д. Задачи, возникающие в научной, производственной и оборонной деятельности, называются *прикладными задачами*, а программы их решения — *прикладными программами*.

Решение прикладных задач на ОВС может быть организовано различными способами. Например, можно поставить целью добиться максимальной производительности системы, определяемой числом задач, решаемых системой в единицу времени. Для этого необходимо максимально загрузить оборудование ОВС, т. е. создать условия, гарантирующие наличие работы для каждого из устройств системы в каждый момент времени. Чтобы достичь этого, необходимо предоставить в распоряжение ОВС по возможности больший набор работ, которые система могла бы принимать к исполнению в порядке, гарантирующем высокую загрузку оборудования (отсутствие простоев) и, как следствие этого, высокую производительность. ОВС, выполнение работ в которых организуется путем загрузки в систему пакета (набора) задач, обрабатываемых в порядке, ориентированном на минимизацию времени обработки всего пакета, называются *системами пакетной обработки* (СПО).

В СПО задачи выбираются из пакета и обрабатываются в порядке, минимизирующем время обработки пакета задачи. В результате этого производительность системы увеличивается, но среднее время обработки каждой задачи оказывается далеко не минимальным. Во многих случаях экономия, получаемая за счет лучшего использования оборудования при пакетной обработке задач, оказывается меньше потерь, связанных с простоем пользователей, ожидающих результатов обработки. В этих случаях целесообразно строить системы, в которых облачные вычислительные процессы планируются таким образом, чтобы по возможности уменьшить среднее время решения задач, определяемое промежутком времени от момента поступления задачи на обработку до момента окончания обработки. ОВС общего назначения, используемые для решения задач в режиме, обеспечивающем минимальное среднее время получения результатов по каждой задаче, называется *системами оперативной обработки* (СОО). Чтобы обеспечить минимум среднего времени ответа на требование пользователя, система должна обладать следующими свойствами: 1) принимать задачи на обработку в момент их возникновения — практически немедленно; 2) обслуживать одновременно несколько пользователей, обеспечивая ввод и вывод информации параллельно для нескольких пользователей и, возможно, параллельную обработку нескольких программ; 3) выполнять задачи в

порядке, обеспечивающем минимум среднего времени пребывания задач в системе (среднего времени ответа).

Практика выработала два основных способа организации оперативной обработки: **разделение времени и диалоговый режим.**

Способ разделения времени состоит в выделении поочередно каждому из активных пользователей, создавших задачи, кванта процессорного времени, обеспечивающего выполнение требуемого количества машинных операций. Каждый из пользователей периодически в течение нескольких секунд получает доступ к ЭВМ, в результате чего среднее время ответа пользователям уменьшается. Системы, в которых обработка задач организуется путем последовательного выделения квантов времени всем принятым на обработку задачам, называются *системами с разделением (распределением) времени* (СРВ). В СРВ время квантуется принудительно механизмом, входящим в состав системы.

Аналогичный эффект можно получить, если организовать взаимодействие пользователя и системы таким образом, чтобы процесс решения задачи был расчленен на достаточно небольшие этапы. На каждом этапе пользователь формирует некоторый запрос на обслуживание, обработка которого занимает сравнительно небольшое время. В таком случае одна ЭВМ оказывается способной обслуживать одновременно несколько пользователей. Такой порядок взаимодействия пользователя с системой является диалогом: пользователь формирует задание для машины (посылает соответствующее сообщение), получает ответ, подготовленный машиной, анализирует ответ и посылает очередное задание. ОВС, предназначенные для решения задач, представляемых в виде совокупности независимых шагов, называются *системами с диалоговым режимом работы*. Диалоговый режим удобен для отладки программ, при решении творческих задач, требующих эвристического подхода (аналитические преобразования, доказательство теорем), при конструировании и т. д.

Классификация ОВС по быстродействию. Важная характеристика ОВС — производительность, определяемая количеством «вычислительной работы», выполняемой системой в единицу времени. Использование этой характеристики в качестве признака классификации ОВС очевидно. Однако к настоящему времени не выявлена мера «вычислительной работы», пригодная для практических применений. В связи с этим производительность ОВС характеризуют косвенно — суммарным быстродействием процессоров, входящих в состав ОВС. Быстродействие процессора обычно определяется средним числом операций, выполняемых им за секунду. При этом

учитывается частота (вероятность) использования разнотипных операций в процессе решения задач. Исходя из суммарного быстродействия B процессоров, ОВС общего назначения условно подразделяют на следующие классы: *малые*, *средние*, *большие*. При этом имеется в виду, что состав оборудования и программного обеспечения ОВС определенным образом сбалансирован с быстродействием процессоров.

Назначение и производительность — внешние характеристики ОВС, т. е. они характеризуют систему как целостный объект. При одном назначении и фиксированной производительности системы могут иметь различную внутреннюю организацию. С учетом этого ОВС целесообразно классифицировать и с позиций их внутренней организации.

Структуры ОВС. Наиболее существенный признак внутренней организации ОВС — способ построения аппаратурной части системы. В зависимости от состава аппаратурной части выделяют ОВС одномашинные, мультипроцессорные и многомашинные.

Одномашинная ОВС включает в себя одну ЭВМ, содержащую единственный процессор и необходимый комплект сопутствующих устройств (запоминающих устройств, каналов ввода — вывода и т. д.). При таком составе оборудования в каждый момент времени ОВС способна выполнять счет по одной программе и, возможно, несколько операций ввода — вывода, выполняемых параллельно.

Мультипроцессорная ОВС включает в себя комплекс оборудования, содержащий несколько процессоров, способных параллельно и независимо друг от друга обрабатывать программы. Процессоры связаны с большим комплектом периферийных устройств (внешних запоминающих устройств и устройств ввода — вывода) и, возможно, с общей оперативной памятью. Наличие общего для всех процессоров оборудования позволяет считать, что процессоры и сопутствующие устройства образуют одну мультипроцессорную ОВС. Мультипроцессорная система, состоящая из одинаковых процессоров, называется *однородной*; в противном случае — *неоднородной*.

Многомашинная ОВС состоит из нескольких связанных между собой ЭВМ. Каждая ЭВМ содержит один или несколько процессоров, соответствующий комплект запоминающих устройств и устройств ввода — вывода и работает большую часть времени самостоятельно. В некоторые моменты времени одна ЭВМ может обращаться к другой ЭВМ с целью передачи информации. Многомашинные ОВС подразделяются на *однородные* и *неоднородные* в зависимости от однотипности или разнотипности ЭВМ, входящих в состав системы.

Система из нескольких территориально разобщенных ЭВМ, связанных между собой каналами передачи данных, называется *облачной вычислительной сетью*.

1.4. Классы проблем облачных технологий

Классификацию проблем облачных технологий можно осуществить по разным критериям. Проводить ее жестко невозможно, она зависит от цели и ресурсов. Предложим классификацию проблем облачных технологий по ниже приведенным критериям.

1. По отношению проблем облачных технологий к окружающей среде:
 - открытые (есть обмен ресурсами с окружающей средой);
 - закрытые (нет обмена ресурсами с окружающей средой).
2. По происхождению проблем облачных технологий (элементов, связей, подсистем):
 - искусственные (орудия, механизмы, машины, автоматы, роботы и т.д.);
 - естественные (живые, неживые, экологические, социальные и т.д.);
 - виртуальные (воображаемые и, хотя реально не существующие, но функционирующие так же, как и в случае, если бы они существовали);
 - смешанные (экономические, биотехнические, организационные и т.д.).
3. По описанию переменных проблем облачных технологий:
 - с качественными переменными (имеющие лишь содержательное описание);
 - с количественными переменными (имеющие дискретно или непрерывно описываемые количественным образом переменные);
 - смешанного (количественно-качественное) описания.
4. По типу описания закона (законов) функционирования облачных технологий:
 - типа "Черный ящик" (неизвестен полностью закон функционирования облачных технологий; известны только входные и выходные сообщения);

- не параметризованные (закон не описан; описываем с помощью хотя бы неизвестных параметров; известны лишь некоторые априорные свойства закона);
 - параметризованные (закон известен с точностью до параметров и его возможно отнести к некоторому классу зависимостей);
 - типа "Белый (прозрачный) ящик" (полностью известен закон).
5. По способу управления облачными технологиями:
- управляемые извне (без обратной связи, регулируемые, управляемые структурно, информационно или функционально);
 - управляемые изнутри (самоуправляемые или саморегулируемые - программно управляемые, регулируемые автоматически, адаптируемые - приспособляемые с помощью управляемых изменений состояний, и самоорганизующиеся - изменяющие во времени и в пространстве свою структуру наиболее оптимально, упорядочивающие свою структуру под воздействием внутренних и внешних факторов);
 - с комбинированным управлением (автоматические, полуавтоматические, автоматизированные, организационные).
6. По предметному признаку (проблемы жизнеобеспечения человека (человечества)):
- финансовые;
 - социальные;
 - промышленные;
 - аграрные;
 - транспортные;
 - экономические;
 - здравоохранения;
 - образования;
 - науки;
 - культуры;
 - правоохранительные;
 - оборонные;
 - экологические.

Облачную технологию будем называть большой, если ее исследование или моделирование затруднено из-за большой размерности, т.е. множество состояний облачной технологии S имеет большую размерность. Какую же размерность нужно считать большой? Об этом мы можем судить только для конкретной проблемы, конкретной цели исследуемой проблемы и конкретных средств (ресурсов). Большая облачная технология сводится к облачной технологии меньшей размерности использованием более мощных технологических средств (или ресурсов) либо разбиением технологической задачи на ряд задач меньшей размерности (если это возможно).

Во многих печатных работах можно встретить словосочетания "сложная задача", "сложная технология", "сложная система" и т.п. Интуитивно, как правило, под этими понятиями понимается какое-то особое поведение технологического процесса, делающее невозможным (непреодолимая сложность) или особо трудным (преодолимая сложность) описание, исследование, предсказание или оценку поведения, развития технологического процесса.

Существуют различные определения сложности. Облачную технологию будем называть сложной, если в ней не хватает ресурсов (главным образом, информационных) для эффективного описания (состояний, законов функционирования) и управления облачными технологиями: определения, описания управляющих параметров или для принятия решений в таких облачных технологиях (в таких облачных технологиях всегда должна быть подпроблема принятия решения). Сложной будем считать и такую облачную технологию, для которой по ее трем видам описания нельзя выявить ее траекторию, сущность, и поэтому необходимо еще дополнительное интегральное описание (интегральная модель поведения, или конфигуратор) - морфолого-функционально-инфологическое.

Пример. Сложными технологиями являются, например, химические реакции, если их исследовать на молекулярном уровне; клетка биологического образования, взятая на метаболическом уровне; мозг человека, если его исследовать с точки зрения выполняемых человеком интеллектуальных действий; экономика, рассматриваемая на макроуровне (т.е. макроэкономика); человеческое общество - на политико-религиозно-культурном уровне; ЭВМ, как средство получения знаний; язык - во многих аспектах его рассмотрения. В сложных облачных технологиях результат функционирования не может быть задан заранее, даже с некоторой вероятностной оценкой адекватности. Причины такой неопределенности - как внешние, так и

внутренние, как в структуре, так и в описании функционирования, эволюции. Сложность этих облачных технологий обусловлена их сложным поведением. Сложность облачной технологии зависит от принятого уровня описания или изучения облачных технологий - макроскопического или микроскопического. Сложность облачных технологий может определяться не только большим количеством подпроблем и сложной структурой, но и сложностью поведения. Сложность облачных технологий может быть *внешней и внутренней*. *Внутренняя* сложность определяется сложностью множества внутренних состояний, потенциально оцениваемых по проявлениям проблемы и сложности управления облачными технологиями. *Внешняя* сложность определяется сложностью взаимоотношений с окружающей средой, сложностью управления облачными технологиями, потенциально оцениваемых по обратным связям облачной технологии и среды.

Предложим классификацию сложных облачных технологий по различным типам сложности:

- структурной или организационной (не хватает ресурсов для построения, описания, управления структурой);
- динамической или временной (не хватает ресурсов для описания динамики поведения облачных технологий и управления их траекторией);
- информационной или информационно-логической, инфологической (не хватает ресурсов для информационного, информационно-логического описания облачных технологий);
- вычислительной или реализации результатов исследования (не хватает ресурсов для эффективного прогноза, расчетов параметров облачных технологий, или их проведение затруднено из-за нехватки ресурсов);
- алгоритмической или конструктивной (не хватает ресурсов для описания алгоритма функционирования или управления облачными технологиями, для функционального описания облачных технологий);
- развития или эволюции, самоорганизации (не хватает ресурсов для устойчивого развития, самоорганизации).

Чем сложнее рассматриваемая облачная технология, тем более разнообразные и более сложные внутренние информационные процессы приходится актуализировать для того, чтобы была

достигнута цель, т.е. в результате реализации сформированных рекомендаций по решению задач облачных технологий они (облачные технологии) продолжали бы функционировать или развиваться.

Пример. Поведение ряда различных реальных технологий (например, соединенных между собой проводников с сопротивлениями $x_1, x_2, \dots, x_n$ или химических соединений с концентрациями $x_1, x_2, \dots, x_n$, участвующих в реакции химических реагентов) описывается системой линейных алгебраических уравнений, записываемых в матричном виде:

$$X=AX+B$$

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ x_n \end{pmatrix}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ b_n \end{pmatrix}$$

Заполнение матрицы **A** (ее структура) будет отражать сложность описываемой технологии. Если, например, матрица **A** - верхнетреугольная матрица (элемент, расположенный на пересечении *i*-ой строки и *j*-го столбца всегда равен 0 при *i*>*j*), то независимо от *n* (размерности технологии) она легко исследуется на разрешимость. Для этого достаточно выполнить обратный ход метода Гаусса. Если же матрица **A** - общего вида (не является ни симметричной, ни ленточной, ни разреженной и т.д.), то технологию сложнее исследовать (так как при этом необходимо выполнить более сложную вычислительно и динамически процедуру прямого хода метода Гаусса). Следовательно, технология будет обладать структурной сложностью (которая уже может повлечь за собой и вычислительную сложность, например, при нахождении решения). Если число *n* достаточно велико, то неразрешимость задачи хранения матрицы **A** верхнетреугольного вида в оперативной памяти компьютера может стать причиной вычислительной и динамической сложности исходной задачи. Попытка использовать эти данные путем считывания с диска приведет к многократному увеличению времени счета (увеличит динамическую сложность - добавятся факторы работы с диском).

Пример. Пусть имеется динамическая технологическая проблема, поведение которой описывается задачей Коши вида

$$y'(t) = ky(t), \quad y(0) = a$$

Эта задача имеет решение:

$$y(t) = ae^{-kt}$$

Отсюда видно, что *y(t)* при *k*=10 изменяется на порядок быстрее, чем *y(t)* при *k*=1, и динамику технологии сложнее будет отслеживать: более точное предсказание для *t* → 0 и малых *k* связано с дополнительными затратами на вычисления. Следовательно, алгоритмически, информационно, динамически и структурно "не очень сложная технология" (при *a*, *k* ≠ 0) может стать вычислительно и, возможно, эволюционно сложной (при *t* → 0), а при больших *t* (*t* → ∞) - и непредсказуемой. Например, для больших *t* значения накапливаемых погрешностей вычислений решения могут перекрыть значения самого решения. Если при этом задавать нулевые начальные данные *a* ≠ 0, то технология может перестать быть, например, информационно несложной, особенно, если *a* трудно априорно определить.

Пример. Упрощение технических средств работы в сетях,

например, научные достижения, позволяющие подключать компьютер непосредственно к сети, "к розетке электрической сети", наблюдается наряду с усложнением самих сетей, например, с увеличением количества абонентов и информационных потоков в интернете. Наряду с усложнением самой сети интернета, упрощаются (для пользователя!) средства доступа к ней, увеличиваются ее вычислительные возможности.

Структурная сложность технологии оказывает влияние на динамическую, вычислительную сложность. Изменение динамической сложности может привести к изменениям структурной сложности, хотя это не является обязательным условием. Сложной технологией может быть и технология, не являющаяся большой технологией; существенным при этом может стать связность (сила связности) элементов и подпроцессов технологии (см. вышеприведенный пример с матрицей системы линейных алгебраических уравнений). Сложность технологии определяется целями и ресурсами (набором задач, которые она призвана решать).

Пример. Сложность телекоммуникационной сети определяется:

1. необходимой скоростью передачи данных;
2. протоколами, связями и типами связей (например, для селекторного совещания необходима голосовая телеконференция);
3. необходимостью видеосопровождения.

Само понятие сложности технологии не является чем-то универсальным, неизменным и может меняться динамически, от состояния к состоянию. При этом и слабые связи, взаимоотношения подпроцессов могут повышать сложность технологии.

Пример. Рассмотрим процедуру деления единичного отрезка $[0; 1]$ с последующим выкидыванием среднего из трех отрезков и дотраиванием на выкинутом отрезке равностороннего треугольника (рис. 1.7); эту процедуру будем повторять каждый раз вновь к каждому из остающихся после выкидывания отрезков. Этот процесс является структурно простым, но динамически сложным, более того, образуется динамически интересная и трудно прослеживаемая картина технологии, становящейся "все больше и больше, все сложнее и сложнее". Такого рода структуры называют *фракталами*, или *фрактальными структурами* (фрактал - от fraction - "дробь" и fracture -

"излом", т.е. изломанный объект с дробной размерностью). Его отличительная черта - *самоподобие*, т.е. сколь угодно малая часть фрактала по своей структуре подобна целому, как ветка - дереву.

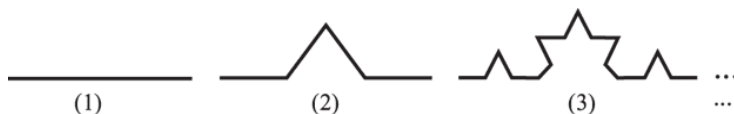


Рис. 1.7. Фрактальный объект (кривая Коха)

Уменьшив сложность технологии, часто можно увеличить ее информативность, исследуемость.

Пример. Выбор рациональной проекции пространственного объекта (т.е. более оптимальная визуализация связей и отношений его частей) делает чертеж более информативным. Используя в качестве устройства эксперимента микроскоп, можно рассмотреть некоторые невидимые невооруженным глазом свойства объекта.

Технологический процесс называется *связным*, если любые два технологических подпроцесса обмениваются ресурсом, т.е. между ними есть некоторые ресурсоориентированные отношения, связи. При определении меры сложности технологии важно выделить инвариантные свойства технологии или информационные инварианты и вводить меру сложности технологии на основе их описаний.

Приведем математический аппарат, позволяющий формализовать понятие сложности, хотя отметим, что понятие сложности - "сложное".

Мерой технологического процесса будем называть некоторую непрерывную действительную неотрицательную функцию, определенную на множестве технологических процессов (событий, систем, множеств) и являющуюся аддитивной, т.е. мера конечного объединения технологических процессов (событий, систем, множеств) равна сумме мер каждого технологического процесса (события).

Как же определять меру сложности для технологического процесса различной структуры? Ответ на этот не менее сложный вопрос не может быть однозначным и даже вполне определённым.

Сложность технологического процесса связывается с мерой $\mu(S)$ - мерой сложности или числовой неотрицательной функцией (критерием, шкалой) заданной (заданным) на некотором множестве элементов и подпроцессов технологического процесса S . Возможны различные способы определения меры сложности технологического процесса. Сложность структуры технологического

процесса можно определять топологической энтропией - сложностью конфигурации структуры (процесса):

$$S = k \ln W,$$

где $k=1,38 \times 10^{-16}$ (эрг / град) - постоянная Больцмана, W - вероятность состояния технологического процесса. В случае разной вероятности состояний эта формула будет иметь вид (мы ниже вернемся к детальному обсуждению этой формулы и ее различных модификаций):

$$S' = -k \sum_{i=1}^n p_i \ln p_i$$

Пример. Определим сложность *иерархического* технологического процесса как число уровней иерархии. Увеличение сложности технологического процесса при этом требует больших ресурсов для достижения цели. Определим сложность линейной структуры как количество подпроцессов технологического процесса. Определим сложность *сетевой* структуры как максимальную из сложностей всех линейных структур, соответствующих различным стратегиям достижения цели (путей, ведущих от начального подпроцесса к конечному). Сложность технологического процесса с *матричной* структурой можно определить количеством подпроцессов технологического процесса. Усложнение некоторого подпроцесса технологического процесса приведет к усложнению всего технологического процесса в случае линейной структуры, и, возможно, в случае иерархической, сетевой и матричной структур.

Пример. Для многоатомных молекул число межъядерных расстояний (оно определяет конфигурацию молекулы) можно считать оценкой сложности топологии (геометрической сложности) молекулы. Из химии и математики известна эта оценка: $3N-6$, где N - число атомов в молекуле. Для твердых растворов можно считать W равной числу перестановок атомов разных сортов в заданных позициях структуры; для чистого кристалла $W=1$, для смешанного - $W>1$. Для чистого кристалла сложность структуры $S=0$, а для смешанного - $S>0$, что и следовало ожидать.

Пример. В эколого-экономических технологических процессах сложность технологического процесса может пониматься как

эволюционируемость, сложность эволюции технологического процесса, в частности, мера сложности - как функция изменений, происходящих в технологическом процессе в результате контакта с окружающей средой, и эта мера может определяться сложностью взаимодействия между технологическим процессом (в организме, в организации) и средой, ее управляемостью. Эволюционную сложность эволюционирующего технологического процесса можно определить как разность между внутренней сложностью и внешней сложностью (сложностью полного управления технологическим процессом). Решения должны приниматься (для устойчивости технологического процесса) таким образом, чтобы *эволюционная сложность равнялась нулю, т.е. чтобы совпадали внутренняя и внешняя сложности. Чем меньше эта разность, тем устойчивее технологический процесс*, например, чем более сбалансированы внутрирыночные отношения и регулирующие их управляющие государственные воздействия - тем устойчивее рынок и рыночные отношения.

Пример. В математических, формальных технологических процессах, сложность технологического процесса может пониматься как алгоритмизируемость, вычислимость оператора системы S , в частности, как число операции и операндов, необходимых для получения корректного результата при любом допустимом входном наборе. Сложность алгоритма может быть определена количеством операций, осуществляемых командами алгоритма для самого "худшего" (самого длительного по пути достижения цели) тестового набора данных.

Пример. Сложность программного комплекса L может быть определена как логическая сложность и измерена в виде $L = L_1 + L_2 + L_3 + L_4 + L_5$, где L_1 - общее число всех логических операторов, L_2 - общее число всех исполняемых операторов, L_3 - показатель сложности всех циклов (определяется с помощью числа циклов и их вложенности), L_4 - показатель сложности циклов (определяется числом условных операторов на каждом уровне вложенности), L_5 - определяется числом ветвлений во всех условных операторах.

Пример. Аналогично примеру, приведенному в книге Дж. Касти, рассмотрим трагедию В. Шекспира "Ромео и Джульетта". Выделим и опишем 3 совокупности: A - пьеса, акты, сцены, мизансцены; B -

действующие лица; C - комментарии, пьеса, сюжет, явление, реплики. Определим иерархические уровни и элементы этих совокупностей.

A:

уровень $N+2$ - Пьеса;

уровень $N+1$ - Акты $\{a_1, a_2, a_3, a_4, a_5\}$;

уровень N - Сцены $\{s_1, s_2, \dots, s_q\}$;

уровень $N-1$ - Мизансцены $\{m_1, m_2, \dots, m_{26}\}$.

B:

уровень N - Действующие лица $\{c_1, c_2, \dots, c_{25}\} = \{\text{Ромео, Джульетта, \dots}\}$.

C:

уровень $N+3$ - Пролог (адресован непосредственно зрителю и лежит вне действий, разворачивающихся в пьесе);

уровень $N+2$ - Пьеса;

уровень $N+1$ - Сюжетные линии $\{p_1, p_2, p_3, p_4\} = \{\text{Вражда семейств Капулетти и Монтекки в Вероне, Любовь Джульетты и Ромео и их венчание, Убийство Тибальда и вражда семейств требует отмщения, Ромео вынужден скрываться, Сватовство Париса к Джульетте, Трагический исход}\}$;

уровень N - Явления $\{u_1, u_2, \dots, u_8\} = \{\text{Любовь Ромео и Джульетты, Взаимоотношения между семейством Капулетти и Монтекки, Венчание Ромео и Джульетты, Схватка Ромео и Тибальда, Ромео вынужден скрываться, Сватовство Париса, Решение Джульетты, Гибель влюблённых}\}$;

уровень $N-1$ - Реплики $\{r_1, r_2, \dots, r_{104}\} = \{104 \text{ реплики в пьесе, которые определяются как слова, обращённые к зрителю, действующему лицу и развивающие неизвестный пока зрителю сюжет}\}$.

Отношения между этими совокупностями на различных уровнях иерархии определяемы из этих совокупностей. Например, если Y - сюжеты, X - действующие лица, то естественно определить связь 1 между X , Y так: действующее лицо из совокупности X уровня $N+1$ участвует в сюжете Y уровня $N+1$. Тогда связность структуры трагедии можно изобразить следующей схемой (рис. 1.8):

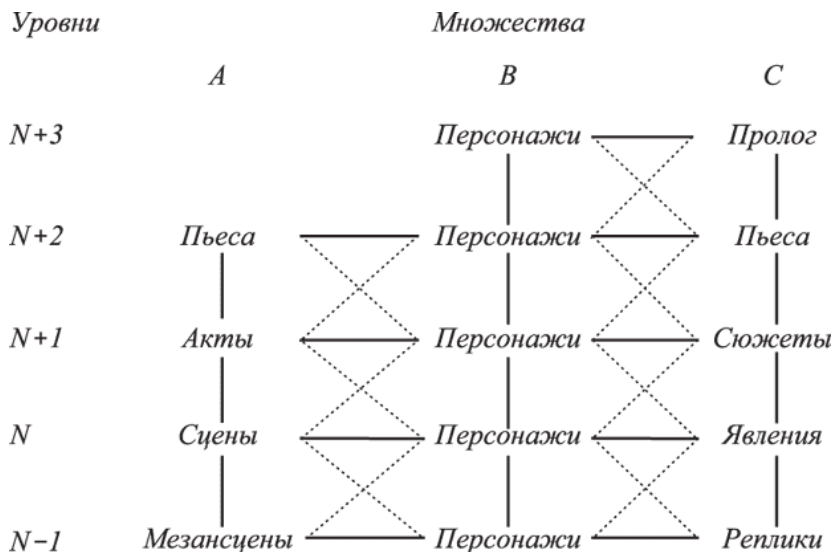


Рис. 1.8. Схема структурных связей пьесы

В этом комплексе $K(Y, X)$ все три сюжета становятся отдельными компонентами только на уровне связности $q=8$. Это означает, что сюжетные линии могут быть различны только для зрителей, следящих за 9 действующими лицами. Аналогично, при $q=6$ имеются всего 2 компоненты $\{p_1, p_2\}$, $\{p_3\}$. Следовательно, если зрители могут отслеживать только 7 персонажей, то они видят пьесу, как бы состоящую из двух сюжетов, где p_1 , p_2 (мир влюбленных и вражда семейств) объединены. В комплексе $K(Y, X)$ при $q=5$ имеются 3 компоненты. Следовательно, зрители, видевшие только 6 сцен, воспринимают 3 сюжета, не связанные друг с другом. Сюжеты p_1 и p_2 объединяются при $q=4$, и поэтому зрители могут видеть эти два сюжета как один, если следят только за 5 сценами. Все 3 сюжета сливаются, когда зрители следят лишь за 3 сценами. В комплексе $K(Y, X)$ явление u_8 доминирует в структуре при $q=35$, u_3 - при $q=26$,

u_6 - при $q=10$. Следовательно, u_8 вероятнее всего поймут те зрители, которые прослушали 36 реплик, хотя для понимания u_3 необходимо 27 реплик, а для понимания u_6 - только 11 реплик.

Таким образом, проведенный анализ дает понимание сложности технологического процесса.

Будем различать так называемые "жесткие" и "мягкие" технологические процессы, в основном, по используемым критериям рассмотрения. Исследование "жестких" технологических процессов обычно опирается на категории: "консультирование", "формирование процессов" "оптимизация", "реализация сформированных процессов", "функция цели процесса" и другие. Для "мягких" процессов используются чаще категории: "возможность", "желательность", "адаптируемость", "здоровый смысл", "рациональность" и другие. Методы также различны: для "жестких" процессов - методы однокритериальной оптимизации, теория вероятностей и математическая статистика, теория игр и другие; для "мягких" процессов - многокритериальная оптимизация и принятие решений (часто в условиях неопределенности), метод Дельфи, теория катастроф, нечеткие множества и нечеткая логика, эвристическое программирование и др.

1.5. Классификация услуг, предоставляемых облачными технологиями

Рынок технологических услуг, предоставляемых облачными технологиями, предлагает клиентам практически весь спектр услуг. Рынок технологических услуг хаотичен. В довольно нестабильной экономике наблюдается определенное переплетение технологических услуг и других видов профессиональных услуг. Автор данной работы представляет это переплетение в виде схемы (рис.1.9), в которой менеджмент-консалтинг выступает в качестве ядра этой схемы.



Рис. 1.9. Связь менеджмент-консалтинга и других видов услуг

Рассмотрим наиболее распространенные виды оказываемых услуг.

1. Управленческий консалтинг.
 1. Оргдиагностика, экспресс-диагностика.
 2. Организационное развитие.
 3. Антикризисное управление.
 4. Стратегическое планирование и управление.
 5. Инновационный консалтинг.
 6. Разработка организационных структур и систем управления.
 7. Управление организационными изменениями.
 8. Реструктуризация.
 9. Управление корпоративными финансами.
 10. Управленческий учет.
 11. Аутсорсинг.

12. “Прокат директоров”.
13. Делопроизводство (разработка должностных инструкций).
2. Инвестиционный консалтинг.
 1. Управление проектами.
 2. Подготовка инвестиционных меморандумов.
 3. Разработка и комплексная экспертиза инвестиционных проектов.
 4. Разработка бизнес-планов, ТЭО.
 5. Управление финансовыми рисками.
 6. Лизинг.
 7. Анализ и мониторинг инвестиционных проектов.
 8. Сопровождение инвестиционных проектов.
 9. Фандрайзинг.
3. Аудитинг.
 1. Общехозяйственный аудит.
 2. Аудит финансовых институтов.
 3. Внутренний аудит.
 4. Системный и финансовый анализ хозяйственной деятельности.
 5. Постановка и ведение бухгалтерского учета.
 6. Налоговое планирование и оптимизация налогообложения.
 7. Взаиморасчеты.
 8. Бюджетирование.
4. Фондовый консалтинг.
 1. Управление финансовыми ресурсами.
 2. Оценка имущественных комплексов.
 3. Оценка и котировка ценных бумаг.
 4. Размещение и покупка ценных бумаг.
 5. Эмиссия ценных бумаг.
 6. Брокерское обслуживание.
 7. Оценка неимущественных прав и интересов.
5. Управление качеством.
 1. Создание системы качества.
 2. Сертификация системы качества.
 3. Проведение аудита системы качества.
 4. Проведение внутренних проверок системы качества.
 5. Анализ документации системы качества.
 6. Написание Руководства по качеству.
 7. Разработка и внедрение элементов TQM.
6. Инжиниринг.

1. Оценка технологических возможностей предприятия.
 2. Подготовка производственного процесса.
 3. Обеспечение процесса производства.
 4. Внедрение новых технологий, новой техники.
 5. Производственный дизайн.
7. Информационно-технологический консалтинг.
 1. Автоматизация офисов и документооборота.
 2. Автоматизация бизнес-процессов (BPA).
 3. Реинжиниринг бизнес-процессов (BRP).
 4. Планирование ресурсов предприятия (ERP).
 5. Разработка и сопровождение специализированного программного обеспечения (СПО).
 6. Создание и сопровождение специализированных баз данных (СБД).
 7. Системное интегрирование.
 8. Создание и сопровождение компьютерных систем.
 9. Проектирование и установки компьютерных сетей.
 10. Предоставление Интернет-ресурсов.
 11. Установка Web-серверов и разработка Web-сайтов в Интернете.
8. Маркетинговый консалтинг.
 1. Маркетинг-планирование.
 2. Комплексный анализ рынка.
 3. Разработка систем стимулирования рынка.
 4. Анализ тенденций сбыта.
 5. Ценообразование и ценовая политика.
 6. Поиск деловых партнеров.
 7. Оптимизация логистики.
9. Реклама и отношения с общественностью (public relations).
 1. Формирование имиджа организации.
 2. Комплексные рекламные компании.
 3. Медиа-планирование.
 4. Подготовка и проведение компаний public relations.
 5. Информационная поддержка.
 6. Предвыборные технологии.
10. Кадровый консалтинг.
 1. Оптимизация организационной структуры.
 2. Оптимизация взаимодействия подразделений (формирование команды).
 3. Разработка стратегии мотивации персонала.
 4. Разрешение внутренних конфликтов.
 5. Медиаторство.

6. Психологическое консультирование.
11. Обучение.
 1. Конференции.
 2. Курсы.
 3. Семинары.
 4. Круглые столы.
 5. Тренинги.
 6. Деловые игры.
 7. Организационно-деловые игры.
 8. Ролевые игры.
 9. Метод мастер-класса.
 10. Кейс-технологии.
 11. Системно-мыследеятельностная (СМД) методология.
 12. Теория решения изобретательских задач (ТРИЗ).
12. Безопасность организации.
 1. Технические системы безопасности.
 2. Охранные услуги.
 3. Проверка благонадежности потенциальных партнеров.
 4. Проведение экспертизы объектов и информации.
 5. Патентная деятельность.
 6. Защита информации.
13. Юридический консалтинг.
 1. Создание, регистрация, аккредитация, инкорпорация предприятий.
 2. Акционирование предприятий.
 3. Экспортно-импортные операции (ВЭД).
 4. Offshore.
 5. Патентование и лицензирование.
 6. Возврат долгов, банкротство, ликвидация.
 7. Реструктуризация задолженности.
 8. Ведение дел в арбитражных и третейских судах.
 9. Юридическая экспертиза документов.
 10. Лоббирование интересов клиента.
 11. Участие в переговорах (медиаторство).

Рассмотрим основные тенденции развития услуг, приведенных в данной классификации.

1. **Управленческий консалтинг**, главная цель которого заключается в разработке комплекса мероприятий, направленных на улучшение качества руководства и повышение эффективности работы как предприятия в целом, так и каждого из сотрудников.

Диагностика состояния предприятия является исходной точкой в процессе управленческого консультирования как при разработке любого инвестиционного проекта, так и любых других изменений, планируемых на предприятии. Данный вид услуг наряду с такими услугами как: стратегическое планирование, реструктуризация, разработка управленческих решений, управление организационными изменениями и инновационный консалтинг имеют относительно постоянный спрос у клиентов. Наиболее перспективными видами консалтинга являются аутсорсинг и “прокат директоров”. Аутсорсинг (outsourcing) основан на полной или частичной передаче рутинных функций предприятия (например, таких, как бухгалтерские расчеты, расчет налогов, управление персоналом и т.п.) консалтинговой фирме с целью сосредоточения собственных усилий на решении ключевых стратегических задач. “Прокат директоров” используется при временном отсутствии руководства или недавно уволенном. Организационное развитие и делопроизводство или администрирование, хотя и являются отдельными видами консалтинга, мы отнесли к управленческому консультированию в силу неразвитости спроса на них в отдельных клиентах. Такие услуги, как управление корпоративными финансами и управленческий учет при переходе на международные стандарты управления финансовыми ресурсами весьма актуальны. Основной целью создания системы управленческой отчетности является обеспечение менеджеров предприятия своевременной и необходимой информацией для принятия эффективных управленческих решений. В основе реализации практически всех услуг лежит анализ существующих и ожидаемых финансовых потоков предприятия. Наиболее эффективной и рекомендуемой методикой является эффективный инструментарий "Business Toolkits", подготовленный группой компаний, среди которых можно отметить Arthur Andersen, Carana Corporation, Deloitte Touche Tohmatsu International, а также специалистами Международного Корпуса Экспертов (International Executive Service Corps).

2. Инвестиционный консалтинг. Сохранение и наращивание производственного, технологического, финансового и экономического потенциала предприятий является наиболее важной задачей современного бизнеса. Деятельность в данном направлении носит название инвестиционной, и служит условием развития предприятия. Пока существуют инвестиционные фонды, банки, программы развития, частные вкладчики и т.п., данный вид консалтинга будет иметь

наибольший спрос, особенно в развивающихся странах. Фандрайзинг (поиск инвестирования) является одним из базовых видов инвестиционного консалтинга. Одним из первых шагов при получении инвестирования является получение и изучение информации о наиболее вероятных источниках финансирования, поскольку условия получения тех или иных видов финансовых ресурсов напрямую определяют конкретные финансовые показатели продвигаемого инвестиционного проекта и саму возможность его реализации. Многие консалтинговые фирмы имеют свои базы данных потенциальных инвесторов или же являются официальными посредниками при получении финансирования.

Лизинг является видом инвестиционной деятельности по приобретению имущества и передаче его на основании договора лизинга физическим или юридическим лицам за определенную плату, на определенный срок и на определенных условиях, обусловленных договором, с правом выкупа имущества лизингополучателем. Лизинговая схема, за счет отнесения лизинговых платежей в себестоимость у лизингополучателя, может иметь существенные преимущества перед обычными схемами финансирования инвестиционных проектов.

Управление проектами мы считаем наиболее перспективной на рынке консалтинговых услуг. Данный вид услуг включает в себя все вышеперечисленные виды инвестиционного консалтинга.

3. Аудитинг. Согласно определению комитета Американской бухгалтерской ассоциации, *"аудит - это системный процесс получения и оценки объективных данных об экономических действиях и событиях, устанавливающий уровень их соответствия определенному критерию и предоставляющий результаты заинтересованному пользователю".*

Основной **целью** аудиторской деятельности является установление достоверной бухгалтерской (финансовой) отчетности экономических субъектов и соответствия совершенных ими финансовых и хозяйственных операций нормативным актам.

Первоочередная цель аудита, таким образом, заключается в защите интересов пользователей экономической информацией, в предоставлении им объективных данных о финансовом состоянии того или иного предприятия. Задача аудитора состоит в оценке достоверности финансовых отчетов и снижении риска появления в них искаженных данных.

Наряду с уже давно распространенными аудиторскими услугами является весьма перспективный вид услуг – *бюджетирование*.

Внедрение системы бюджетирования позволяет руководству предприятия получить значительный положительный эффект за счет возможности согласования деятельности подразделений внутри компании и подчинения ее общей стратегической цели. Беспристрастные бюджеты позволяют менеджерам вначале конкретизировать поставленные задачи, а потом проанализировать состояние дел в компании: как прошел текущий период - принес ли прибыли или убытки, есть ли деньги на счету, что можно ожидать в ближайшем будущем.

4. Фондовый консалтинг является одним из стабильных консалтинговых рынков. Постоянно наблюдается рост спроса на оценочные услуги. Одним из принципиально новых видов фондового консалтинга являются *нематериальные активы*. К нематериальным активам относят объекты, не имеющие физических свойств, но включаемые в активы предприятия и обычно требующие постепенной амортизации в течение срока их использования. Таковыми являются: патенты, авторские права и др. интеллектуальная собственность, права на собственность и пользование землей и полезными ископаемыми, торговые марки, программное обеспечение ЭВМ и т.п.

5. Управление качеством. В последнее десятилетие отмечен повышенный интерес к качеству поставляемой потребителю продукции. Наличие у поставщика системы качества становится важным фактором повышения конкурентоспособности его продукции и необходимым условием для выхода на мировой рынок. Разработка систем качества ведется в соответствии с требованиями стандартов ИСО серии 9000. Стандарты ИСО серии 9000 имеют главные цели: удовлетворение потребителей, вовлечение персонала фирмы в борьбу за качество и предпочтение внимания процессам. Принципиальным отличием создаваемых у поставщика систем качества, основанных на стандартах ИСО серии 9000, является их четкая направленность на удовлетворение конкретных запросов потребителя. Самым новым и, к сожалению, недостаточно востребованным является услуга по разработке и внедрению элементов всеобщего качества (Total Quality Management – TQM). Автор считает данный вид услуг базовым и ставит вопрос об управлении качеством консалтинговых услуг и внедрения системы качества в консалтинговых фирмах.

6. Инжиниринг является одним из родоначальников консалтинга и *представляет собой услуги по инженерной подготовке и обеспечению процесса производства, строительства и эксплуатации различных объектов.*

7. Информационно-технологический консалтинг. Все услуги на данный вид консалтинга являются новыми и очень перспективными вследствие роста спроса на информационную технику и **информационные технологии**. Под информационными технологиями понимаются способы автоматизированной переработки информации. Новыми и весьма перспективными являются услуги по моделированию бизнес-процессов и по их реинжинирингу. Впервые термин **"реинжиниринг бизнес-процессов"** (от англ. business process reengineering, BPR) был введен М.Хаммером и Д.Чампи, которые определяют этот вид деятельности как *"фундаментальное перепроектирование бизнес-процессов компаний для достижения коренных улучшений в основных показателях их деятельности: стоимость, качество, услуги и темпы"*. BPR является направлением, возникшим на стыке двух различных сфер деятельности - управления (менеджмента) и информатизации. Именно поэтому реинжиниринг требует новых специфических средств представления и обработки проблемной информации, понятных как менеджерам, так и разработчикам информационных систем. Подобные средства требуют интеграции ключевых достижений информационных технологий и создания соответствующих инструментальных средств поддержки реинжиниринга. Таким образом, *реинжиниринг бизнес-процессов ориентирован на коренную перестройку всей деятельности предприятия, а не на частичные изменения в той или иной сфере управления.*

8. Маркетинговый консалтинг. Самое распространенное заболевание в бизнесе - "рыночная слепота", которое порождает множество осложнений. По статистике свыше 75 процентов всех банкротств в мире объясняются плохим знанием своего собственного продукта, потребителя, каналов распространения (сбыта) и конкурентов. *Самым новым видом услуг является оптимизация логистики предприятия.*

9. Реклама и отношения с общественностью. Целью этого вида услуг является создание в глазах общественности положительного и вместе с тем особенного, отличающегося от конкурентов, образа

(имиджа) товара или фирмы. Одним из новых и непривычных для нашего общества, но наиболее эффективных видов услуг являются предвыборные технологии. Большинство новоиспеченных, а также и старых политиков часто используют такие услуги.

10. Кадровый консалтинг (рекрутмент) – это услуги по подбору и оценке кадров. Данная область услуг развилась как отдельный вид в связи с постоянно растущей потребностью в руководящих кадрах и высококвалифицированных специалистах. Кадровый консалтинг является одним из наиболее распространенных видов услуг. Консалтинг в области конфликтологии является новым видом услуг в области кадрового консалтинга и требует своего значительного развития.

11. Обучение. Обучающее консультирование является одним из самых старым видом консалтинга. *Отличие различных видов обучения от управленческого консультирования состоит в том, что тренер не рекомендует менеджерам управленческие решения по различным проблемам, а лишь передает им свои знания о том, как их надо принимать, и предоставляет информацию по соответствующей проблематике.*

Автор считает, что обучение в действии, а именно активно-игровые методы - деловые игры, организационно-деятельностные игры, ролевые игры и метод мастер-класса являются наиболее эффективными и перспективными, так как предоставляют не только информацию по интересующей проблеме, но и заставляют менеджера самому принимать решения в экстремальных ситуациях, проигрывая различные варианты исхода проблемы.

12. Безопасность организации – одно из новых направлений консалтинга. **Экономическая безопасность предприятия** – это обеспечение защищенности хозяйственных отношений, развитие экономического потенциала компании, повышение уровня благосостояния всех ее сотрудников и формирующих основы защищенности фирмы от различных опасностей и угроз, источником зарождения и развития которых выступают внутренние и внешние противоречия.

Защита информации является наиболее актуальной на сегодняшний день и наиболее распространенным видом услуг по безопасности. Британский Институт Стандартов (BSI) при поддержке группы

коммерческих организаций, среди которых были Shell, National Westminster Bank, Midland Bank, Unilever, British Telecommunications, Marks & Spencer, Logica и другие, осуществил разработку стандарта информационной безопасности, получившего обозначение BS 7799. Стандарт BS 7799 определяет общую организацию, классификацию данных, системы доступа, направления планирования, ответственность сотрудников, использование оценки риска и т.д. в контексте информационной безопасности.

13. Юридический консалтинг является одним из основных родоначальников консалтинга и вообще профессиональных услуг. *Юристы оказывают услуги как в форме советов, так и путем непосредственного ведения дел своих клиентов. Они также могут давать экспертные заключения о соответствии действия или документа юридическим нормам.*

Приведенный классификатор является предметным, т.е. по видам предоставляемых услуг. Можно также разделить консалтинговую деятельность и по другим критериям и признакам, представленными в таблице 1.3.

Таблица 1.3

Признаки	Типы
• По предмету работы:	• Проектное консультирование • Процессное консультирование
• По типам консультирования	• Экспертное • Процессное • Обучающее
• По целям	• Целевое • Многоцелевое
• По результату на выходе	• Нормативное • Ценностное • Проблемное
• По задачам	• Оперативное • Стратегическое • Организационное

	• Развивающее
• По специализации консультантов	• Специалисты • Джeneralисты
• По месту расположения консультантов	• Внешние • Внутренние
• По длительности проведения работ	• Краткосрочное • Среднесрочное • Долгосрочное • Разовое • Периодическое • Абонементное
• По клиенту	• Индивидуальное • Организационное
• По масштабу клиента	• Государственные предприятия • Крупные приватизированные предприятия • Средние предприятия • Малые предприятия • Частные предприниматели • Некоммерческие предприятия
• По отрасли клиента	• Банковское дело • Энергетика • Транспорт • Здравоохранение и т.д.
• По месторасположению консалтинговой фирмы	• Локальные • Региональные • Государственные • Межгосударственные
• По методу работы	• Диагностика • Расчеты • Разработки • Документирование • Информирование • Обучение • Переговоры

• По функциям консультанта (ролям)	• Диагностик • Эксперт • Аудитор • Разработчик • Идеолог • Преподаватель • Тренер • Медиатор
• По типу консультант-клиентских отношений	• Рекомендации по конкретным изменениям • Новые ценностные ориентации • Решение проблем • Новые методы совместного решения проблем

Альтернативная классификация технологических услуг

Как и любой другой продукт технологические услуги имеют свой жизненный цикл: *стадии проектирования и разработок, апробирования, выпуска на рынок, роста, зрелости, насыщения и упадка*. Этот процесс имеет также и временное измерение, так как некоторые услуги устаревают, и их следует заменять, а некоторые услуги появляются, и их следует внедрять. Поэтому классификация технологических услуг необходима, особенно при идентификации каждого нового вида технологии. Системность характеристик, даваемых каждому новому виду технологии, позволяет сравнить его возможности и оценить его эффективность по отношению к другим. Оценивая соотношение видов технологий в хозяйственной практике, можно наблюдать тенденции этих наблюдений в динамике, что позволяет вносить соответствующие коррективы в развитие технологической деятельности.

Существует множество классификаторов технологической деятельности. Одну классификацию мы рассмотрели выше, ниже приведены некоторые другие. Альтернативная классификация видов технологических услуг приведена ниже.

1. Общее управление.

1. Определение эффективности системы управления.
 2. Оценка бизнеса.
 3. Управление нововведениями.
 4. Определение конкурентоспособности / изучение конъюнктуры рынка.
 5. Диверсификация или становление нового бизнеса.
 6. Международное управление.
 7. Оценка управления.
 8. Слияние и приобретение.
 9. Организационная структура и развитие.
 10. Приватизация.
 11. Управление проектом.
 12. Управление качеством.
 13. Реорганизация инженерных служб.
 14. Исследование и развитие.
 15. Стратегическое планирование.
2. Администрирование.
 1. Анализ работы канцелярии.
 2. Размещение и перемещение отделов.
 3. Управление офисом.
 4. Организация и методы управления.
 5. Регулирование риска.
 6. Гарантии безопасности.
 7. Планирование рабочих помещений и их оснащение.
3. Финансовое управление.
 1. Системы учета.
 2. Оценка капитальных затрат.
 3. Оборот фирмы.
 4. Снижение себестоимости.
 5. Неплатежеспособность (банкротство).
 6. Увеличение прибыли.
 7. Увеличение доходов.
 8. Налогообложение.
 9. Финансовые резервы.
4. Управление кадрами.
 1. Профессиональное движение и сокращение штатов.
 2. Культура корпорации.
 3. Равные возможности.
 4. Поиск кадров.
 5. Отбор кадров.
 6. Здоровье и безопасность.
 7. Программы поощрения.

8. Внутренние связи.
 9. Оценка работ.
 10. Трудовые соглашения и занятость.
 11. Обучение менеджменту.
 12. Планирование рабочей силы.
 13. Мотивация.
 14. Пенсии.
 15. Анализ функционирования.
 16. Психологическая оценка.
 17. Вознаграждение.
 18. Повышение квалификации работников.
5. Маркетинг.
1. Реклама и содействие сбыту.
 2. Корпоративный образ и отношения с общественностью.
 3. Послепродажное обслуживание заказчиков.
 4. Дизайн.
 5. Прямой маркетинг.
 6. Международный маркетинг.
 7. Исследование рынка.
 8. Стратегия маркетинга.
 9. Разработка новой продукции.
 10. Ценообразование.
 11. Розничная продажа и дилерство.
 12. Управление сбытом.
 13. Обучение сбыту.
 14. Социально-экономические исследования и прогнозирование.
6. Производство.
1. Автоматизация.
 2. Использование оборудования и его техническое обслуживание.
 3. Промышленный инжиниринг.
 4. Переработка материалов.
 5. Регулирование внутреннего распределения материалов.
 6. Упаковка.
 7. Схема организации работ на предприятии.
 8. Конструирование и совершенствование продукции.
 9. Управление производством.
 10. Планирование и контроль за производством.
 11. Повышение производительности.

12. Закупки.
13. Контроль качества.
14. Контроль за поставкой узлов и деталей.
7. Информационная технология.
 1. САПР/АСУ.
 2. Применение компьютеров в аудите и оценке.
 3. Электронная издательская деятельность.
 4. Информационно-поисковые системы.
 5. Административные информационные системы.
 6. Проектирование и разработка систем.
 7. Выбор и установка систем.
8. Специализированные услуги.
 1. Обучающее консультирование.
 2. Консалтинг по управлению электроэнергетикой.
 3. Инженерный консалтинг.
 4. Экологический консалтинг.
 5. Информационный консалтинг.
 6. Юридический консалтинг.
 7. Консалтинг по управлению распределением материалов и материально-техническому снабжению.
 8. Консалтинг в государственном секторе.
 9. Консалтинг по телекоммуникациям.

Приведенная классификация является предметной. *Специалисты вырабатывают рекомендации по выполнению вышеперечисленных функций менеджеров.* Они советуют им как осуществлять общее управление, администрирование, финансовое управление, управление кадрами и т.д. Рассмотрим основные задачи, которые выполняют специалисты по вышеперечисленной классификации.

1. Специалисты по общему управлению решают проблемы, связанные с самим существованием бизнеса и его перспективами.
2. Специалисты по административному управлению решают проблемы, связанные с ведением бизнеса, т.е. помогают оптимизировать управление организацией.
3. Специалисты по финансовому управлению оказывают помощь в решении трех основных задач:
 1. Поиск источников финансирования и эффективного его использования;

2. Анализ финансовой деятельности организации и повышение ее эффективности;
3. Перспективное укрепление финансового положения организации.
4. Специалисты по управлению кадрами содействуют менеджерам в оптимизации привлечения и использования человеческого ресурса.
5. Специалисты по маркетингу содействуют такому функционированию организации, при котором производимая продукция будет куплена потребителем.
6. Специалисты по организации производства решают задачи, связанные с инжинирингом, аудитом и контролем качества и т.д.
7. Специалисты по информационным технологиям решают задачи, связанные с проектированием и внедрением информационных технологий на предприятии.
8. Специалисты по специализированным услугам решают специализированные задачи, не связанные ни с одним из перечисленных видов услуг, и отличающиеся от них по методам, по объектам или по характеру внедряемых знаний.

1.6. Типология облачных технологических систем

Проблема построения классификации облачных технологических систем

Классификация облачных технологических систем представляет собой исключительно сложную проблему, которая еще не разрешена в теории облачных технологий. Причин несколько. Наиболее существенная из них заключается в том, что конкретных разновидностей облачных технологических систем столь много, что создается ощущение их полного совпадения со всеми типами имеющихся объектов. Другая причина состоит в абстрактности понимания самой облачной технологической системы. Сказывается также и то обстоятельство, что до сих пор не выработаны общие параметры, характеризующие облачную технологическую систему. Заметим, что с развитием различных наук и практики обостряется необходимость разработки *сущностной классификации облачных технологических систем*. Важнейшее требование к научной классификации облачных технологических систем — обоснованность ее оснований, которые должны получить концептуальное обоснование. Сама классификация, как некоторая умозрительная система, должна

удовлетворять требованиям достаточности оснований и охвата совокупности имеющихся и возможных облачных технологических систем. Таким образом, лучшая классификация, подобно периодической системе элементов Д. И. Менделеева, должна помочь предсказать появление или открытие принципиально новых облачных технологических систем.

Самое важное назначение классификации — *описание свойств ее классов и подклассов, видов и подвидов*, что позволяет использовать ее для идентификации конкретных облачных технологических систем, с которыми сталкиваются люди в тех или иных областях деятельности.

Существуют самые разнообразные подходы к классификации облачных технологических систем. Автор анализирует классификации видов, представленных на рис. 1.11.

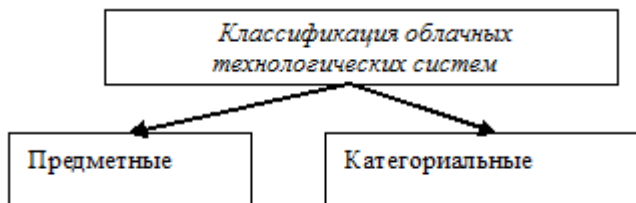


Рис. 1.11. Виды классификаций

Предметная классификация строится на основе выделения всех видов конкретных облачных технологических систем. Такова, например, приведенная в табл. 1.4 классификация, которая представляет собой матрицу.

Таблица 1.4

Облачные технологические системы	Простые	Сложные	Очень сложные
Детерминированные			
Вероятностные			

В клетки данной матрицы заносятся конкретные разновидности существующих облачных технологических систем.

Категориальные классификации выделяют облачные технологические системы по некоторым признакам, общими для всех облачных технологических систем. Такой подход может быть реализован на основе определения понятия облачной технологической системы в триаде "операция" — "свойство" — "отношение" (табл. 1.5)

Таблица 1.5

Классификация облачных технологических систем

Категориальные характеристики	Компоненты облачной технологической системы		
	Свойства	Элементы	Отношения
Моно			
Поли			
Статические			
Динамические (функционирующие)			
Динамические (развивающиеся)			
Детерминистские			
Случайные			
Простые			
Сложные			

Отсюда выделяем такие типы облачных технологических систем:

- монофункциональная детерминистская простая (совокупность процедур);
- монофункциональная случайная простая (тот же пример, только при наличии помех);
- монофункциональная вероятностная сложная (технологии с обилием и сложностью случайных факторов);
- моноразвивающаяся детерминированная простая (технология, описывающая производство новой продукции);
- моноразвивающаяся вероятностная простая;
- моноразвивающаяся вероятностная сложная;
- полифункциональная детерминированная простая;
- полифункциональная детерминированная сложная;
- полифункциональная вероятностная простая;
- полифункциональная вероятностная сложная;
- полиразвивающаяся детерминированная простая;
- полиразвивающаяся детерминированная сложная;
- полиразвивающаяся вероятностная простая;
- полиразвивающаяся вероятностная сложная.

Сущностная классификация облачных технологических систем

В основу любой классификации должна быть положена концепция, объясняющая классифицируемые явления. Как мы уже говорили, *классификация представляет собой многоступенчатое, разветвленное деление логического объема понятия*. В результате создается система соподчиненных понятий: *делимое понятие — род, новые понятия — виды, виды видов (подвиды) и т.д.* Концептуальный подход к классификации делает ее *сущностной* (рис. 1.12).

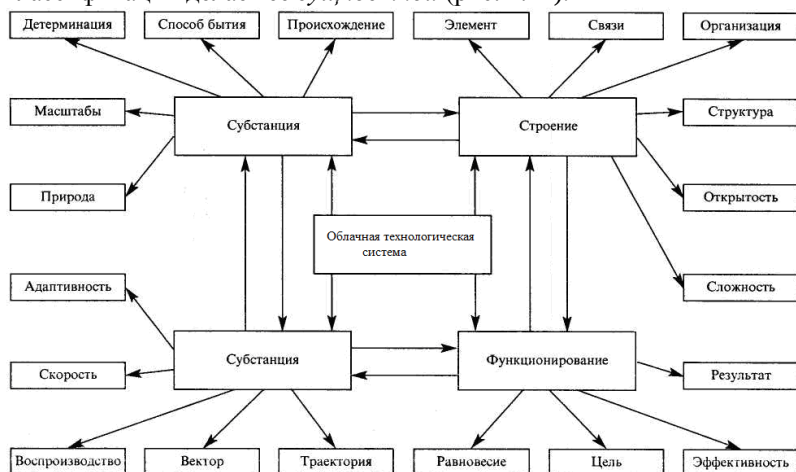


Рис. 1.12. Интерпретация основных составляющих облачной технологической системы

Для построения сущностной классификации облачных технологических систем к ним нужно подходить с системных позиций. По нашему мнению, любая облачная технологическая система характеризуется четырьмя основными параметрами: *субстанцией, строением, функционированием и развитием*.

При этом под *субстанцией* понимается сущностное свойство предмета как целостности, основание и центр всех его изменений, активную их причину и источник функционирования. Под *строением* облачной технологической системы подразумевается наличие в облачной технологической системе элементов, связей и организации.

Функционирование рассматривается как процесс реализации облачной технологической системы своих функций, а *развитие* — как процесс

качественных изменений в облачной технологической системе. Тогда **облачная технологическая система** — это *структурно-функциональная развивающаяся субстанциональная целостность*.

Каждая из четырех составляющих сущностной характеристики облачной технологической системы может быть представлена совокупностями основополагающих параметров, соответствующих их природе. Так, **субстанция** может быть представлена природой облачных технологических систем, их *сложностью, масштабами, детерминацией, происхождением и способом бытия (функционирования)*. Для **строения** свойственны *элементы, связи, организация, структура и сложность*. **Функционирование** выражается *равновесием, целью, результатом и эффективностью*. **Развитие** характеризуется *адаптивностью, скоростью, воспроизводством, вектором и траекторией*.

На основании выделенных параметров можно создать классификацию облачной технологической системы на *субстанциональном уровне, уровнях строения, функционирования и развития* (табл. 1.6).

Таблица 1.6

Классификация облачных технологических систем

Основание классификации	Облачная технологическая система	
	Вид	Характеристика
<i>Субстанциональный уровень облачной технологической системы</i>		
Природа облачной технологической системы	Физическая	Синтезирующая совокупность процедур (операций), которые базируются на физических законах)
	Техническая	Синтезирующая совокупность технологий, которые призваны решать технологические задачи технических систем (станок, конвейер, техническое устройство)
	Кибернетическая	Синтезирующая совокупность технологий, которые призваны решать технологические задачи множества взаимосвязанных объектов — элементов системы, способных воспринимать, запоминать и перерабатывать информацию, а также обмениваться информацией (автопилот, регулятор температуры в холодильнике, ЭВМ, человеческий мозг, живой организм, биологическая популяция, человеческое общество)

	Хими- ческая	Синтезирующая совокупность технологий, которые призваны решать технологические задачи множества элементов, взаимосвязанных химическими связями (молекула, химическое соединение)
	Биологи- ческая	Синтезирующая совокупность технологий, которые призваны решать технологические задачи организмов или их сообществ (растение, животное)
	Социаль- ная	Синтезирующая совокупность технологий, которые призваны решать технологические задачи общества или некоторой его составляющей, развивающаяся как целое (государство, экономика, законодательство)
	Ителлек- туальная	Синтезирующая совокупность технологий, которые призваны решать технологические задачи знаний, способы познания и мышления (методы научного познания, математика)
Способ существова- ния облачной технологичес- кой системы	Абстрактная Материаль- ная	Единство некоторых символов или знаков (теория, система исчисления) Совокупность материальных явлений (город, горная система)
Характер детермини- рованности	Стохастиче- ская Детермини- рованная	Поведение носит вероятностный характер (ценообразование, игра) Поведение предопределено (падение предметов)
Происхожде- ние облачной технологичес- кой системы	Естественная Искусствен- ная Естественно- искусствен- ная	Возникает и развивается естественно, без вмешательства человека Возникает и развивается благодаря человеку Возникает и развивается естественно и путем вмешательства человека
Масштабы	Микромасш- табная Макромасш- табная Метапроб-	Относительно небольшое образование (малая или контактная группа, вирусы) Значительное по размеру образование

	лема Мегопроб- лема	Сверхбольшое образование (общество, планета) Бесконечное по размеру образование (Вселенная)
<i>Уровень строения облачной технологической системы</i>		
Количество элементов	Одноклеточ- ная Бинарная Триарная Четырех- элементная Много- элементная	Состоит из одного элемента (операция, прием) Состоит из двух элементов (процедура — операця) Состоит из трех элементов Состоит из четырех элементов Состоит из множества элементов
Степень открытости	Открытая Закрытая	Открыта для воздействия внешней среды (демократическое общество) Закрыта для воздействия внешней среды (тоталитарное общество)
Характер взаимодей- ствия элементов	Координа- ционная Иерархиче- ская Координа- ционно- иерархиче- ская	Элементы отличаются равноправием (дружба, отделы одного уровня в системе управления) Элементы соподчинены (система управления) Объединяет равноправные и неравноправные элементы (общество)
Степень организован- ности	Недостаточ- но организо- ванная проб- лема или хаос-проб- лема Суммативная Организован- ная	Переходная экономика, реорганизуемое предприятие, кризис Неразвитое взаимодействие между элементами (империя Александра Македонского) Выраженные организационными структурами (правительство, предприятие)

	Заорганизованная	Однозначно предопределенное поведение элементов (армия, тюрьма)
Степень сложности	Простая	Состоит из небольшого числа элементов и связей между ними
	Сложная	Включает в себя большое число простых систем
	Сверхсложная	Включает в себя большое число сложных систем
Тип структуры	Линейная	Линейная структура взаимосвязи элементов (цепь, участок метро)
	Сотовая	Разветвленные связи, множество путей прохождения информации (связь)
	Иерархическая	Соподчинение элементов (власть)
	Смешанная	Наличие всех типов структуры (предприятие)
Наличие информации о строении системы	«Черный ящик»	С неизвестным строением
	«Серый ящик»	С наличием некоторой информации о ее строении
	«Белый ящик»	С известным строением
<i>Уровни функционирования облачной технологической системы</i>		
Характер воспроизводства	Воспроизводимая окружающей средой	Последствия любых действий
	Воспроизводящая себе подобной	Животные, растения
Количество функций	Многофункциональная	Реализация одной функции (контроль)
	Полифункциональная	Реализация одновременно нескольких функций (система управления)
Характер размещения	Плоскостная	Размещена в плоскости (земельный участок)
	Пространственная	Городская среда
	Многомерная	Социальная технология
Равновесие	Равновесная	Сохранение равновесия (рынок)

	Неравновесная	Нарушение равновесия (конфликт)
Цель	Одноцелевая Многоцелевая	Ориентирована на достижение одной цели (система обслуживания) Направлена на достижение нескольких целей (многопрофильная консультационная организация)
Эффективность	Неэффективная Средней эффективности Эффективная	Отличается низкой эффективностью (консультируют неподготовленными специалистами) Достаточно выраженная эффективность (консультации проводятся на среднем профессиональном уровне) Со значительной эффективностью (консультации проводятся на высоком профессиональном уровне)
Результат	С нулевым результатом Результативная С высоким результатом	Не имеет результата (отсутствует положительный результат реализации технологии) Отличается результативностью (отличается средней результативностью реализации технологии) Высокий синергетический результат (отличается высокой результативностью реализации технологии)
<i>Уровень развития облачной технологической системы</i>		
Способность приспосабливаться	Адаптивная Неадаптивная	Способность приспосабливаться, не теряя своей идентичности (перенастраиваемая на различные Классы технологий) Не обладает способностью приспосабливаться (неперенастраиваемая на различные классы технологий)
Вектор развития	Восходящего развития	Свойственен рост показателей развития с той или иной скоростью (экономика периода подъема)

	Нисходящая	Присущее падение показателей развития с той или иной скоростью (экономика в период кризиса,
	Стабильная	Свойственно сохранение показателей (технологии устойчивого развития)
Способность самовоспроизводства	Неорганическая	Неспособность к самовоспроизводству (технические, технологические системы)
	Органическая	Способность к самовоспроизводству (организмы)
Этап развития	Зародыш	Находится на стадии возникновения (ТЗ на разработку технологии)
	Детская	На стадии становления (проектируемая технологическая система)
	Молодая	В процессе достижения зрелости (спроектированная технологическая система)
	Зрелая	Соответствует всем качествам зрелости (реализованная технологическая система)
	Кризисная	В процессе падения показателей, разрушения и перестройки (устаревшая технологическая система)
	Переходная	Переходит из одного состояния в другое (реконструируемая технологическая система)
	Деградирующая	Доминирование процессов ухудшения показателей и разрушения
Траектория развития	Линейная	Подчиняется линейной функции развития (линейные зависимости)
	Нелинейная	Подчиняется нелинейным функциям развития (линейные зависимости)

Данная классификация может быть углублена по нескольким направлениям. Во-первых, она представляет собой дерево, ветвями которого выступают выделенные по основаниям виды облачных

технологических систем. Отсюда каждую конкретную разновидность облачной технологической системы можно представить посредством фиксации ее характеристик по каждому срезу и основанию.

Приведем предметную классификацию облачных технологических систем (рис.1.13)



Рис. 1.13. Предметная классификация облачных технологических систем

1.7. Классификация автоматизированных процессов

1.7.1. Представление процессов

Одно из принципиальных отличий облачного процесса – использование процессного подхода к менеджменту. Основную идею процессного подхода можно свести к следующим положениям:

1. Функционирование любой организации следует представлять в виде сети взаимодействующих между собой процессов;
2. Менеджмент деятельностью организации должен основываться на менеджменте сетью процессов.

Ниже рассматривается подход к описанию и классификации облачных процессов, основанный на применении методологии функционального моделирования IDEF0. В МС ИСО 9000:2000 используется следующее определение процесса: *«Набор взаимосвязанных и взаимодействующих операций (действий), которые преобразуют входы в выходы.»*

Примечание 1. Входы процессов, как правило, являются выходами других процессов.

Примечание 2. Процессы планируются и исполняются при управляемых условиях для добавления ценности.»

Наиболее наглядным способом описания процессов является их графическое представление. В различных работах, посвященных толкованию процессного подхода, предлагаются различные варианты для графического представления процессов.

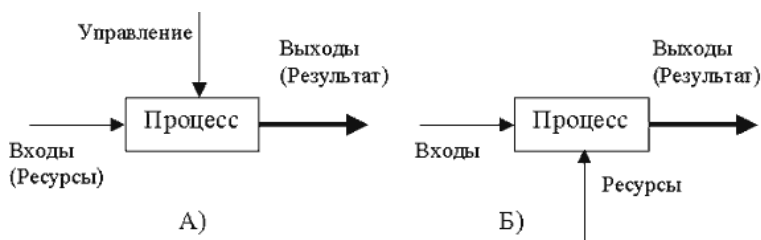


Рис. 1.15. Варианты графического представления процессов.

Описание процессов должно отражать не только отдельные процессы, но также взаимосвязи и взаимодействия между процессами. Процессы вместе с взаимосвязями и взаимодействиями представляют *сеть процессов*.

Описание сети процессов, – это сложная организационно-техническая задача, для решения которой требуются специальные средства описания и анализа.

Впервые это обстоятельство было осознано в середине 70-х годов при реализации комплексных проектов по заказам ВВС США. В то же время была предложена и реализована программа комплексной компьютерной поддержки производства (ICAM – Integrated Computer-Aided Manufacturing), в рамках которой, в частности, применялась методология структурного анализа систем. Позже на базе этого подхода была разработана методология функционального моделирования IDEF0, которая в 1993 году была принята в качестве федерального стандарта в США. В методологии функционального моделирования IDEF0 для графического представления процесса будем использовать следующую нотацию (рис. 1.16).



Рис. 1.16. Графическое представление процесса в IDEF0

В соответствии с методологией IDEF0 процесс представим в виде функционального блока, который преобразует входы в выходы при наличии необходимых ресурсов (механизмов) в управляемых условиях. Взаимосвязи и взаимодействия процессов по рекомендации IDEF0 представляются дугами, соединяющими выходы одних функциональных блоков со входами других.

Модель сети процессов

Модель сети процессов должна отвечать на следующие вопросы:

- Какие процессы в деятельности организации относятся к технологическим ?
- Какова структура этих процессов, включая выходы и потребителей процессов, входы и поставщиков и т.д.?
- Как процессы взаимодействуют друг с другом?
- Как в рамках процессов выполняются требования, определенные в МС ИСО семейства 9000 версии 2000 года?

Требования к функциональной модели

Для того, чтобы функциональная модель сети процессов отвечала на эти и другие вопросы в рамках СК, она должна строиться в соответствии с дополнительными требованиями (помимо тех, которые сформулированы в методологии IDEF0). Ниже приводится не полный

перечень требований, которым должна отвечать функциональная модель процессов:

Функциональная модель строится с точки зрения руководства организации. При таком подходе модель включает все процессы и элементы, влияющие на качество сформированных конечных рекомендаций и процессов.

1. Функциональная модель должна содержать процессы, определенные как обязательные в рамках требований МС ИСО семейства 9000 версии 2000 года. Перечень этих процессов приведен в МС ИСО 9001:2000.
2. Функциональная модель должна содержать элементы (объекты), регламентируемые в МС ИСО семейства 9000 версии 2000 года. Перечень таких элементов приведен в МС ИСО 9001:2000.
3. Функциональная модель должна охватывать все стадии жизненного цикла, относящиеся к сфере функционирования проблемы.

Особенности функциональной модели

Деловой процесс

Для того, чтобы функциональная модель удовлетворяла перечисленным требованиям, она должна строиться как модель делового процесса.

Деловой процесс – это совокупность процессов (операций, действий) и взаимодействий между ними, результатом (выходом) которой является сформированные рекомендации и/или услуги, предоставляемые клиентам, а входами – материальные, информационные и трудовые ресурсы, предоставляемые внешними поставщиками. Таким образом, функциональная модель делового процесса будет охватывать процессы жизненного цикла, а также связанные с ними вспомогательные процессы и процессы менеджмента, входящие в состав деятельности организации. Это полностью согласуется с требованиями МС ИСО семейства 9000 версии 2000 года.

Например, швейное ателье производит (шьет) женские пальто, заключая договора с потребителями. Потребителями продукции являются магазины женской одежды и торгово-посреднические компании. Ателье закупает сырье на комбинатах, а также

у торгово-посреднических компаний. Деловым процессом в швейном ателье является процесс «Производить женские пальто».

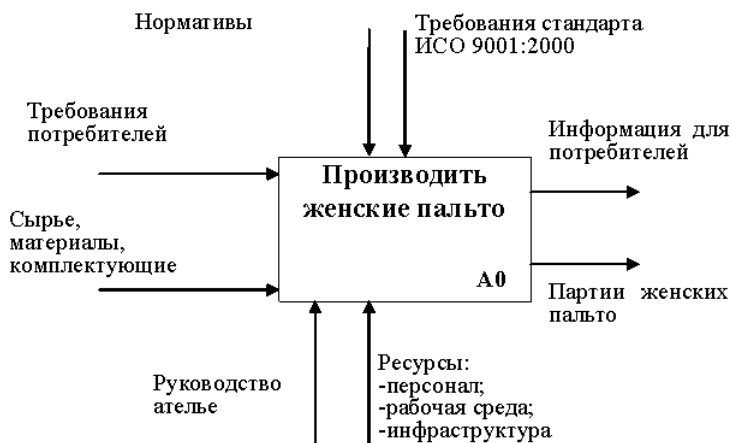


Рис. 1.17. Деловой процесс в швейном ателье

Обязательные процессы и элементы

Согласно требованиям МС ИСО 9001:2000 к обязательным консультационным процессам будем относить:

- Реализация ответственности высшего руководства в рамках системы;
- Менеджмент ресурсами (вспомогательными процессами);
- Менеджмент основными процессами (процессами жизненного цикла);
- Процессы измерения, контроля и улучшения продукции .

К обязательным элементам относятся, в том числе,

- Документы, содержащие политику, цели организации в сфере менеджмента;
- Документы, содержащие ответственность сотрудников организации (должностные инструкции);

- Записи, и т.д.

Соответственно, функциональная модель должна содержать все обязательные процессы и элементы в соответствии с требованиями МС ИСО семейства 9000 версии 2000 года.

В нашем примере деловой процесс в швейном ателье будет иметь следующую структуру (Рис. 1.18).

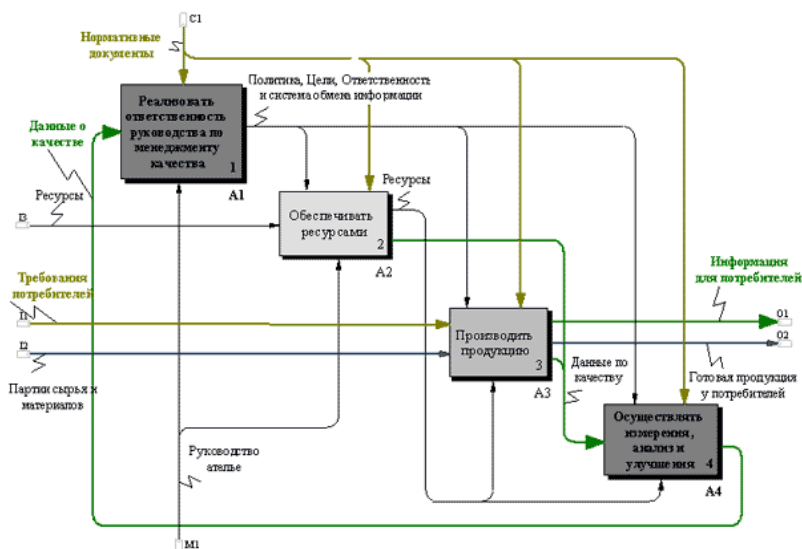


Рис. 1.18. Детализация делового процесса

На диаграмме (рис. 1.18) процесс представлен в виде 4-х взаимодействующих между собой процессов. Каждый из 4 процессов является обязательным с точки зрения выполнения требований МС ИСО 9001:2000.

Стрелки, связывающие функциональные блоки, представляют элементы (объекты), которые передаются с выходов одних процессов на входы других. В том числе, они представляют обязательные с точки зрения МС ИСО 9000:2000 элементы процессов, такие как, например, «Записи сформированных рекомендаций» или «Политика организации в области менеджмента».

Классификация процессов

Классификация процессов является важным этапом анализа деятельности организации. Одна из целей классификации – определение процессов, относящихся к системе производства.

Функциональная модель состоит из двух типов элементов – функциональных блоков и дуг. Соответственно, классификация процессов, представленных на функциональной модели, сводится к классификации собственно функциональных блоков и дуг.

Классификация дуг

В рамках IDEF0-модели дуги в зависимости от их положения на диаграмме уже подразделены на 4 категории: *входные, выходные, управления и механизма*.

Дополнительно дуги могут быть классифицированы в зависимости от категории объектов, которые они представляют на диаграмме. К числу таких категорий могут относиться:

- Материалы, сырье, продукция, ресурсы;
- Информация, данные; записи сформированных рекомендаций; документы;
- Распоряжения руководства, планы, графики; распорядительные документы;
- Стандарты, нормативная документация;
- Ответственные исполнители, сотрудники организации и т.д.

Для того, чтобы выделить в IDEF0-модели элементы определенного типа, при моделировании используются заранее оговоренные соглашения о графическом стиле представления объектов различных категорий. Поскольку дуги на IDEF0-модели представляются прямыми и ломанными линиями, графический стиль для дуг включает соглашение о цвете линии, толщине линии, типе линии (сплошная, пунктирная, штрих - пунктирная, и т.д.), а также о типе стрелки на конце дуги.

Например, дуги, отражающие в рамках функциональной модели «Записи сформированных рекомендаций», изображаются зеленой сплошной линией (Рис. 1.18).

Следует отметить, что установление соответствия между категорией объекта (например, записи сформированных рекомендаций) и графическим стилем представления этого объекта (зеленая сплошная линия) в функциональной модели играет не только презентационную роль. С помощью компьютерной программы такого рода отношения можно обрабатывать, создавая специальные отчеты, содержащие, например, информацию о всех процессах, которые на выходе порождают записи сформированных рекомендаций.

Классификация функциональных блоков

Функциональные блоки в IDEF0-модели могут быть классифицированы в зависимости от категорий процессов, которые они представляют. В рамках функциональных моделей следует использовать категории процессов, которые регламентированы в МС ИСО 9001:2000.

Для того чтобы выделить в IDEF0-модели процессы определенной категории, при моделировании используются заранее оговоренные соглашения о графическом стиле представления соответствующих функциональных блоков в модели по аналогии с соглашениями для дуг.

Процессный подход, составляющий основу версии МС ИСО 9000:2000, требует применения специальных средств для описания и классификации процессов, составляющих деятельность организации. Мы показали, что одним из таких средств может являться методология функционального моделирования IDEF0. В пользу применения методологии IDEF0 для описания и классификации процессов говорит не только возможность методологии решить эту задачу в рамках организации, но также тот факт, что данная методология также является стандартом для функционального моделирования в ряде стран, включая США и Россию. Последнее обстоятельство делает возможным использовать методологию IDEF0 в качестве единого языка для обмена информацией между организациями, аудиторами, экспертами, консультантами.

Методология IDEF0 поддерживается компьютерными программами. Применение компьютерных программ на стадии описания процессов позволит не только повысить эффективность решения задачи, но также использовать эти модели на стадии менеджмента, интегрируя их в корпоративную информационную систему организации.

Классификация архитектуры процессов

Любой процесс достигает своей наивысшей производительности благодаря использованию *унифицированных операций и параллельному выполнению большого числа этих операций*. Именно возможность параллельного функционирования различных операций (работы с перекрытием) является основой ускорения реализации основных операций.

Параллельные процессы будем подразделять на процессы типа SIMD (Single Instruction Multiple Data - *с одним потоком операций (команд) при множественном потоке данных*) и MIMD (Multiple Instruction Multiple Data – *с множественным потоком операций (команд) при множественном потоке данных*). Как и любая другая, приведенная выше классификация не вполне совершенна: существуют процессы прямо в нее не попадающие, имеются также важные признаки, которые в этой классификации не учтены. В частности, к процессам типа SIMD часто относят *векторные средства реализации процессов*, хотя их высокая производительность зависит от другой формы параллелизма - *конвейерной организации средства реализации процессов*. *Многоагрегатные векторные системы* состоят из нескольких векторных средства реализации процессов и поэтому могут быть названы MSIMD (Multiple SIMD).

Рассмотренная выше классификация не делает различия по другим важным характеристикам, например, *по уровню "зернистости" параллельных выполнений операций и методам синхронизации*.

Можно выделить *четыре основных типа архитектуры процессов параллельного функционирования*:

1) *Конвейерная и векторная реализация процессов*.

Основу **конвейерной** реализации процессов составляет раздельное выполнение некоторой операции в *несколько этапов* (за несколько ступеней) с передачей результатов реализации операции в виде данных одного этапа следующему. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько операций.

Конвейеризация эффективна только тогда, когда загрузка средств конвейерного типа близка к полной, а скорость подачи новых операндов соответствует максимальной производительности конвейера. Если происходит задержка, то параллельно будет выполняться меньше операций и суммарная производительность снизится.

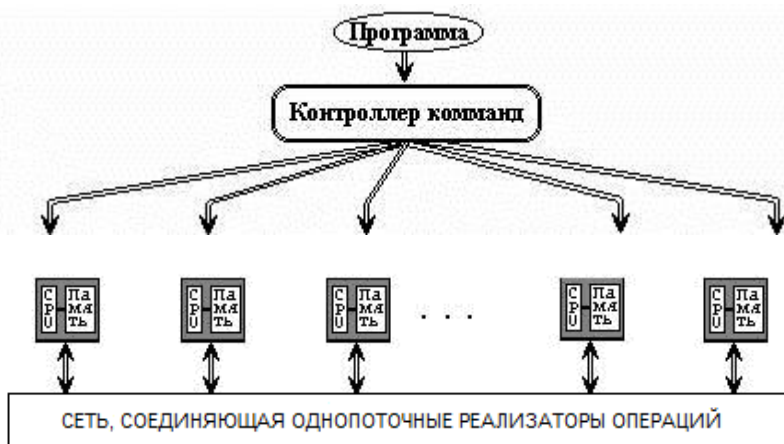
Векторные операции обеспечивают идеальную возможность полной загрузки средств конвейерного типа.

При выполнении векторной команды одна и та же операция применяется ко всем элементам вектора (или чаще всего к соответствующим элементам пары векторов).

Для настройки конвейерного средства на выполнение конкретной операции может потребоваться некоторое установочное время, однако затем операнды могут поступать в конвейер с максимальной скоростью, допускаемой возможностями пропускной способности конвейера. При этом не возникает пауз ни в связи с выборкой новой команды, ни в связи с определением ветви, реализующей операцию, при условном переходе. Таким образом, *главный принцип реализации векторных операций на конвейере состоит в выполнении некоторой элементарной операции или комбинации из нескольких элементарных операций, которые должны повторно применяться к некоторому блоку, представленных в виде данных*. Таким операциям в исходном функциональном блоке соответствуют небольшие компактные циклы.

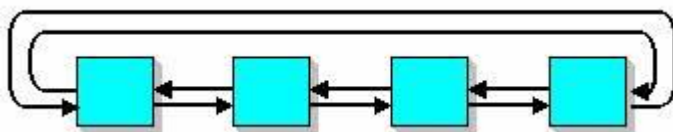
2) Средства типа SIMD.

SIMD средство имеет N идентичных реализаторов операций, N потоков исходных данных и один поток команд. Каждый реализатор операций обладает собственной локальной памятью. Реализаторы операций интерпретируют адреса данных либо как локальные адреса собственной памяти, либо как глобальные адреса, возможно, модифицированные добавлением локального базового адреса. Реализаторы операций получают команды от одного центрального контроллера команд и работают синхронно, то есть на каждом шаге все реализаторы операций выполняют одну и ту же команду над данными из собственной локальной памяти.



Реализаторы операций типа SIMD состоят из большого числа идентичных функциональных элементов, имеющих собственную память. Все функциональные элементы в таком реализаторе операций выполняют один и тот же процесс. Очевидно, что такой реализатор операций, составленный из большого числа функциональных элементов (блоков), может обеспечить очень высокую производительность решения только тех задач, при решении которых все функциональные блоки могут делать одну и ту же работу. Модель реализации процесса для средства SIMD очень похожа на модель реализации процесса для векторного процессора: одиночная операция выполняется над большим блоком данных.

Такая архитектура с распределенными функциями может быть истолкована как *архитектура с параллелизмом данных* (data-parallel), так как параллельность достигается при наличии одиночного потока команд, действующего одновременно на несколько частей данных. Сеть, соединяющая функциональные блоки, обычно имеет регулярную топологию такую как кольцо:



Сеть операций с топологией «кольцо»

В отличие от ограниченного конвейерного функционирования векторного реализатора операций, *матричный реализатор операций* (синоним для большинства SIMD-средств) может быть значительно более гибким. Обработывающие элементы таких реализаторов - это универсальные программируемые микрореализаторы, так что задача, решаемая *параллельно*, может быть достаточно сложной и содержать ветвления. Обычное проявление этой модели в исходной программе примерно такое же, как и в случае векторных операций: циклы на элементах массива, в которых значения, вырабатываемые на одной итерации цикла, не используются на другой итерации цикла.

Модели операций, реализуемые на векторных и матричных реализаторах настолько схожи, что эти реализаторы можно рассматривать как эквивалентные.

3) Средства типа MIMD.

MIMD средство имеет N функциональных реализаторов операций, независимо исполняющих N потоков команд и обрабатывающих N потоков данных. Каждое средство функционирует под управлением собственного потока команд, то есть MIMD средство может параллельно выполнять совершенно разные программы.

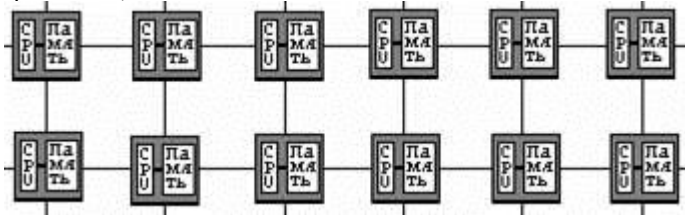


MIMD архитектуры далее классифицируются в зависимости от физической организации памяти, то есть имеет ли средство свою собственную локальную память и обращается к другим блокам памяти, используя коммутирующую сеть, или коммутирующая сеть подсоединяет все средства к общедоступной памяти. Исходя из организации памяти, будем различать следующие типы параллельных архитектур:

- Средства с распределенной памятью (Distributed memory)

Средство может обращаться к локальной памяти, может посылать и получать сообщения, передаваемые по сети, соединяющей средства. Сообщения используются для осуществления связи между средствами

или, что эквивалентно, для чтения и записи удаленных блоков памяти. В идеализированной сети стоимость посылки сообщения между двумя узлами сети не зависит как от расположения обоих узлов, так и от трафика сети, но зависит от длины сообщения.



- Средства с общей (разделяемой) памятью (True shared memory)

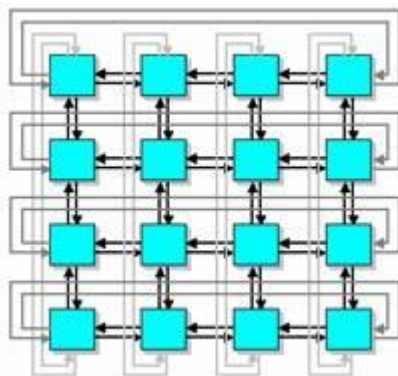
Все средства совместно обращаются к общей памяти, обычно, через шину или иерархию шин. В идеализированной PRAM (Parallel Random Access Machine - параллельное средство с произвольным доступом) модели, часто используемой в исследованиях параллельных процессов, любой реализатор может обращаться к любой ячейке памяти за одно и то же время. На практике масштабируемость этой архитектуры обычно приводит к некоторой форме иерархии памяти. Частота обращений к общей памяти может быть уменьшена за счет сохранения копий часто используемых данных в кэш-памяти, связанной с каждым реализатором. Доступ к этому кэш-памяти намного быстрее, чем непосредственно доступ к общей памяти.

- Реализаторы с виртуальной общей (разделяемой) памятью (Virtual shared memory)

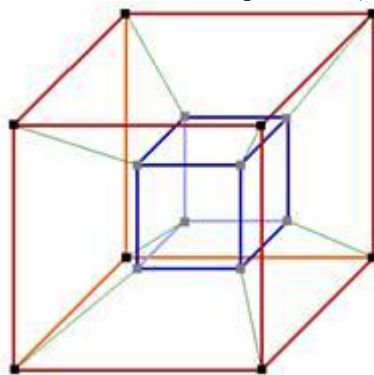
Общая память как таковая отсутствует. Каждый реализатор имеет собственную локальную память и может обращаться к локальной памяти других реализаторов, используя "глобальный адрес". Если "глобальный адрес" указывает не на локальную память, то доступ к памяти реализуется с помощью сообщений, пересылаемых по коммуникационной сети.

MIMD архитектуры с распределенной памятью можно так же **классифицировать по пропускной способности коммутирующей сети**. Например, в архитектуре, в которой пары из реализатора и модуля памяти (реализаторный элемент) соединены сетью с **топологией решетки**, каждый реализатор имеет одно и то же число подключений к сети вне зависимости от числа реализаторов средства. Общая пропускная способность такой сети растет линейно относительно числа реализаторов. В **топологии клика** каждый реализатор должен быть соединен со всеми другими

реализаторами. С другой стороны в архитектуре, имеющей сеть с *топологией гиперкуб*, число соединений реализатора с сетью является логарифмической функцией от числа реализаторов, а пропускная способность сети растет быстрее, чем линейно по отношению к числу реализаторов.



Сеть с топологией 2D-решетки (тор)



Сеть с топологией 2D-гиперкуб (тор)

Термин "мультиреализатор " покрывает большинство средств типа MIMD и (подобно тому, как термин "матричный реализатор" применяется к средствам типа SIMD) часто используется в качестве синонима для средств типа MIMD. В *ультрареализаторной* системе каждый реализаторный элемент (РЭ) выполняет свою операцию достаточно независимо от других реализаторных элементов. Реализаторные элементы, конечно, должны как-то связываться друг с другом, что делает необходимым более подробную классификацию средств типа MIMD. В мультиреализаторах с общей памятью

(*сильносвязанных мультиреализаторов*) имеется память данных и команд, доступная всем РЭ. С общей памятью РЭ связываются с помощью общей шины или сети обмена. В противоположность этому варианту в слабосвязанных *многоореализаторных системах* (средства с локальной памятью) вся память делится между реализаторными элементами и каждый блок памяти доступен только связанному с ним реализатору. Сеть обмена связывает реализаторные элементы друг с другом. Базовой моделью выполнения операций на MIMD-мультиреализаторе является совокупность независимых реализаторов, эпизодически обращающихся к разделяемым данным.

Можно создать большое количество вариантов этой модели. На одном конце спектра - *модель распределенных реализаций операций*, в которой процесс делится на довольно большое число параллельных задач, состоящих из множества подпроцессов. На другом конце спектра - *модель потоковых реализаций операций*, в которых каждая операция в процессе может рассматриваться как отдельный процесс. Такая операция ждет своих входных данных (операндов), которые должны быть переданы ей другими процессами. По их получении операция выполняется, и полученное значение передается тем процессам, которые в нем нуждаются. В потоковых моделях реализаций операций с большим и средним уровнем гранулярности, процессы содержат большое число операций и выполняются в потоковой манере.

4) Многоореализаторные средства с SIMD-реализаторами.

Многие современные средства представляют собой многоореализаторные системы, в которых в качестве реализаторов используются векторные реализаторы или преализаторы типа SIMD. Такие реализаторные средства относятся к реализаторным средствам класса MSIMD.

Языки программирования и соответствующие компиляторы для реализаторов типа MSIMD обычно обеспечивают языковые конструкции, которые позволяют описывать "крупнозернистый" параллелизм. В пределах каждой задачи компилятор автоматически векторизует подходящие циклы. Реализаторы типа MSIMD, как можно себе представить, дают возможность использовать лучший из этих двух принципов декомпозиции: векторные операции ("мелкозернистый" параллелизм) для тех частей процесса, которые подходят для этого, и гибкие возможности MIMD- архитектуры для других частей процесса.

Многоореализаторные системы претерпели ряд этапов своего развития. Первой осваивалась технология SIMD. Однако в настоящее время представляет достаточно большой интерес архитектура

технологии MIMD. Этот интерес главным образом определяется двумя факторами:

1. Архитектура MIMD дает большую гибкость: при наличии адекватной поддержки со стороны аппаратных средств и программного обеспечения MIMD может работать как однопользовательская система, обеспечивая высокопроизводительное формирование рекомендаций для одной задачи, как многореализаторное средство, выполняющее множество задач параллельно, и как некоторая комбинация этих возможностей.

2. Архитектура MIMD может использовать все преимущества облачной технологии на основе строгого учета соотношения стоимость/производительность. В действительности практически все современные многореализаторные средства строятся на тех же микрореализаторах.

Одной из отличительных особенностей многореализаторной системы является сеть обмена, с помощью которой реализаторы соединяются друг с другом или с памятью. Модель обмена настолько важна для многореализаторной системы, что многие характеристики производительности и другие оценки выражаются отношением времени формирования рекомендации к времени обмена, соответствующим решаемым задачам. Предлагается *две основные модели межреализаторного обмена*: одна основана на *передаче сообщений*, другая - на *использовании общей памяти*. В многореализаторной системе с общей памятью один реализатор осуществляет запись в конкретную ячейку, а другой реализатор производит считывание из этой ячейки памяти. Чтобы обеспечить согласованность данных и синхронизацию процессов, обмен может реализовываться по принципу взаимно исключающего доступа к общей памяти методом "почтового ящика".

С ростом числа реализаторов просто невозможно обойти необходимость реализации модели распределенной памяти с высокоскоростной сетью для связи реализаторов.

С быстрым ростом производительности реализаторов и связанным с этим ужесточением требования увеличения полосы пропускания памяти, масштаб систем (т.е. число реализаторов в системе), для которых требуется организация распределенной памяти, уменьшается, также как и уменьшается число реализаторов, которые удастся поддерживать на одной разделяемой шине и общей памяти.

Распределение памяти между отдельными узлами системы имеет два главных преимущества. Во-первых, это эффективный с точки зрения стоимости способ увеличения полосы пропускания памяти, поскольку большинство операций могут выполняться параллельно к локальной

памяти в каждом узле. Во-вторых, это уменьшает задержку выполнения операции (время доступа) к локальной памяти. Эти два преимущества еще больше сокращают количество реализаторов, для которых архитектура с распределенной памятью имеет смысл.

Обычно устройства ввода/вывода, также как и память, распределяются по узлам и в действительности узлы могут состоять из небольшого числа (2-8) реализаторов, соединенных между собой другим способом. Хотя такая кластеризация нескольких реализаторов с памятью и сетевой интерфейс могут быть достаточно полезными с точки зрения эффективности в стоимостном выражении, это не очень существенно для понимания того, как такое средство работает, поэтому мы пока остановимся на системах с одним реализатором на узел. Основная разница в архитектуре, которую следует выделить в средствах с распределенной памятью заключается в том, как осуществляется связь и какова логическая модель памяти.

2. Особенности облачных технологий и облачных вычислений

Облачные технологии - это различные аппаратные, программные средства, методологии и инструменты, которые предоставляются пользователю, как интернет-сервисы, для реализации своих целей, задач, проектов.

Разновидностью облачных технологий являются облачные вычисления.

2.1. Особенности облачных вычислений

Облачные вычисления (англ. *cloud computing*) — модель обеспечения удобного сетевого *доступа по требованию* к некоторому общему фонду конфигурируемых вычислительных ресурсов (например, сетям передачи данных, серверам, устройствам хранения данных, приложениям и сервисам — как вместе, так и по отдельности), которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами или обращениями к провайдеру.

Потребители облачных вычислений могут значительно уменьшить расходы на инфраструктуру информационных технологий (в краткосрочном и среднесрочном планах) и гибко реагировать на изменения вычислительных потребностей, используя свойства вычислительной эластичности (англ. *elastic computing*) облачных услуг.

Ранние концепции использования вычислительных ресурсов по принципу системы коммунального хозяйства относят к 1960-м годам (к Джону Маккарти или Джозефу Ликлайдеру).

Следующими шагами к концептуализации облачных вычислений считаются появление CRM-системы [Salesforce.com](https://www.salesforce.com), предоставляемой по подписке в виде веб-сайта (1999) и начало предоставления услуг по доступу к **вычислительным ресурсам через Интернет** книжным магазином Amazon.com (2002). Развитие сервисов Amazon, фактически превратившейся благодаря этим услугам в **технологическую компанию**, привело к формулировке идеи вычислительной эластичности и запуску в августе 2006 года проекта под названием *Elastic Computing Cloud* (Amazon EC2). Практически одновременно с запуском ЕСС термины *cloud* и *cloud computing* прозвучали в одном из выступлений главы Google Эрика Шмидта, начиная с этого времени встречаются многочисленные упоминания облачных вычислений в СМИ, в публикациях специалистов по информационным технологиям, в научно-исследовательской среде. Отсылка к «**облаку**» использовалась как метафора, основанная на изображении Интернета на диаграмме компьютерной сети, или как **образ сложной инфраструктуры**, за которой скрываются все технические детали.

Запуск в 2009 году приложений Google Apps отмечается как следующий важный шаг к популяризации и осмыслению облачных вычислений. В 2009—2011 годы были сформулированы несколько важных обобщений представлений об облачных вычислениях, в частности, выдвинута **модель частных облачных вычислений**, актуальная для применения внутри организаций, выделены различные **модели обслуживания (SaaS, PaaS, IaaS)**. В 2011 году Национальный институт стандартов и технологий сформировал определение, которое структурировало и зафиксировало все возникшие к этому времени трактовки и вариации относительно облачных вычислений в едином понятии.

2.1.1. Характеристики

Национальным институтом стандартов и технологий США зафиксированы следующие **обязательные характеристики облачных вычислений**:

- **Самообслуживание по требованию** (англ. *self service on demand*) — потребитель самостоятельно определяет и изменяет вычислительные потребности, такие как серверное время, скорости доступа и обработки данных, объём хранимых данных без взаимодействия с представителем поставщика услуг;
- **Универсальный доступ по сети** — услуги доступны потребителям по сети передачи данных вне зависимости от используемого терминального устройства;
- **Объединение ресурсов** (англ. *resource pooling*) — поставщик услуг объединяет ресурсы для обслуживания большого числа потребителей в **единый пул** для динамического перераспределения мощностей между потребителями в условиях постоянного изменения спроса на мощности; при этом потребители контролируют только основные параметры услуги (например, объём данных, скорость доступа), но **фактическое распределение ресурсов, предоставляемых потребителю, осуществляет поставщик** (в некоторых случаях потребители всё-таки могут управлять некоторыми физическими параметрами перераспределения, например, указывать **желаемый центр обработки данных из соображений географической близости**);
- **Эластичность** — услуги могут быть предоставлены, расширены, сужены в любой момент времени, без дополнительных издержек на взаимодействие с поставщиком, как правило, в автоматическом режиме;
- **Учёт потребления** — поставщик услуг автоматически исчисляет потреблённые ресурсы на определённом уровне абстракции (например, объём хранимых данных, пропускная способность, количество пользователей, количество транзакций), и на основе этих данных оценивает объём предоставленных потребителям услуг.

С точки зрения поставщика, благодаря объединению ресурсов и непостоянному характеру потребления со стороны потребителей, облачные вычисления позволяют экономить на масштабах, используя меньшие аппаратные ресурсы, чем требовались бы при выделенных

аппаратных мощностях для каждого потребителя, а за счёт автоматизации процедур модификации выделения ресурсов существенно снижаются затраты на абонентское обслуживание.

С точки зрения потребителя эти характеристики позволяют получить услуги с высоким уровнем доступности (англ. *high availability*) и низкими рисками неработоспособности, обеспечить быстрое масштабирование вычислительной системы благодаря *эластичности* без необходимости создания, обслуживания и модернизации собственной аппаратной инфраструктуры.

Удобство и универсальность доступа обеспечивается широкой доступностью услуг и поддержкой различного класса терминальных устройств (персональных компьютеров, мобильных телефонов, интернет-планшетов).

2.1.2. Модели развёртывания (представления)

Частное облако

Частное облако (англ. *private cloud*) — инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей (например, подразделений одной организации), возможно также клиентами и подрядчиками данной организации. Частное облако может находиться в собственности, управлении и эксплуатации как самой организации, так и третьей стороны (или какой-либо их комбинации), и оно может физически существовать как внутри, так и вне юрисдикции владельца.

Публичное облако

Публичное облако (англ. *public cloud*) — инфраструктура, предназначенная для свободного использования широкой публикой. Публичное облако может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций (или какой-либо их комбинации). Публичное облако физически существует в юрисдикции владельца — поставщика услуг.

Общественное облако

Общественное облако (англ. *community cloud*) — вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи (например, миссии, требований безопасности, политики, и соответствия различным требованиям). Общественное облако может находиться в кооперативной (совместной) собственности, управлении и эксплуатации одной или более из организаций сообщества или третьей стороны (или какой-либо их комбинации), и оно может физически существовать как внутри, так и вне юрисдикции владельца.

Гибридное облако

Гибридное облако (англ. *hybrid cloud*) — это комбинация из двух или более различных облачных инфраструктур (частных, публичных или общественных), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями передачи данных и приложений (например, кратковременное использование ресурсов публичных облаков для балансировки нагрузки между облаками).

2.1.3. Модели обслуживания

Программное обеспечение как услуга

Основная статья: [SaaS](#)

Программное обеспечение как услуга (SaaS, англ. Software-as-a-Service) — модель, в которой потребителю предоставляется возможность использования прикладного программного обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских устройств или посредством тонкого клиента, например, из браузера (например, веб-почта) или посредством интерфейса программы. Контроль и управление основной физической и виртуальной инфраструктурой облака, в том числе сети, серверов, операционных систем, хранения, или даже индивидуальных возможностей приложения (за исключением ограниченного набора пользовательских настроек конфигурации приложения) осуществляется облачным провайдером.

Платформа как услуга

Основная статья: [PaaS](#)

Платформа как услуга (PaaS, англ. Platform-as-a-Service) — модель, когда потребителю предоставляется возможность использования облачной инфраструктуры для **размещения базового программного обеспечения** для последующего размещения на нём новых или существующих приложений (собственных, разработанных на заказ или приобретённых тиражируемых приложений). В состав таких платформ входят инструментальные средства создания, тестирования и выполнения прикладного программного обеспечения — системы управления базами данных, связующее программное обеспечение, среды исполнения языков программирования — предоставляемые облачным провайдером.

Контроль и управление основной физической и виртуальной инфраструктурой облака, в том числе сети, серверов, операционных систем, хранения осуществляется облачным провайдером, за исключением разработанных или установленных приложений, а также, по возможности, параметров конфигурации среды (платформы).

Инфраструктура как услуга

Основная статья: [IaaS](#)

Инфраструктура как услуга (IaaS, англ. Infrastructure-as-a-Service) предоставляется как возможность использования облачной инфраструктуры для **самостоятельного управления ресурсами обработки, хранения, сетями и другими фундаментальными вычислительными ресурсами**, например, потребитель может устанавливать и запускать произвольное программное обеспечение, которое может включать в себя операционные системы, платформенное и прикладное программное обеспечение. Потребитель может контролировать операционные системы, виртуальные системы хранения данных и установленные приложения, а также обладать ограниченным контролем за набором доступных сетевых сервисов (например, межсетевым экраном, DNS). Контроль и управление основной физической и виртуальной инфраструктурой облака, в том числе сети, серверов, типов используемых операционных систем, систем хранения осуществляется облачным провайдером.

2.1.4. Экономические аспекты

При использовании облачных вычислений потребители информационных технологий могут **существенно снизить капитальные расходы — на построение центров обработки данных, закупку серверного и сетевого оборудования, аппаратных и программных решений по обеспечению непрерывности и работоспособности — так как эти расходы поглощаются провайдером облачных услуг.** Кроме того, длительное время построения и ввода в эксплуатацию крупных объектов инфраструктуры информационных технологий и высокая их начальная стоимость ограничивают способность потребителей гибко реагировать на требования рынка, тогда как **облачные технологии обеспечивают возможность практически мгновенно реагировать на увеличение спроса на вычислительные мощности.**

При использовании облачных вычислений затраты потребителя смещаются в сторону операционных — таким образом классифицируются расходы на оплату услуг **облачных провайдеров.**

Для объяснения экономической составляющей облачных подходов к вычислениям часто используется аналогия с услугами водо- или электроснабжения, предоставляемыми в развитых инфраструктурах по соответствующим коммунальным сетям, легкодоступными и оплачиваемыми по мере потребления, в сравнении с разработкой каждым потребителем собственного водозабора или монтажом собственной электроустановки.

2.2. Облачные технологии

За последние годы, эта проблема стала одной из наиболее популярных в ИТ-сфере, о ней написано немало статей, проведено большое количество конференций.

Базой облачных технологий являются облачные вычисления (**cloud computing**) - это технология распределённой обработки данных в которой компьютерные ресурсы и мощности предоставляются пользователю как интернет-сервис в доступной форме – это в некотором смысле рабочая площадка пользователя в интернете, а точнее, на удаленном сервере.

Рассмотрим пример, чтобы убедиться, что практически каждый из нас, так или иначе, уже сталкивался с таким решением. Если вы работаете с электронной почтой (**e-mail**) на каком-то сайте-сервисе (например, gmail), который эту почту позволяет использовать, то это и есть **облачный сервис**, являющийся частью облачной технологии. Или, к примеру, обработка изображений. Если вы уменьшаете размер, переворачиваете свою фотографию в **Photoshop** или другой специальной программе, то к облачной технологии вы не имеете никакого отношения, - всё происходит и обрабатывается локально на вашем компьютере. А вот, если, загрузив изображение, к примеру, через сервис **Picasa**, вы его обрабатываете по ту сторону, т.е. в браузере, то это и есть "облако". Для обеспечения согласованной работы узлов вычислительной сети на стороне облачного провайдера используется специализированное связующее программное обеспечение, обеспечивающее мониторинг состояния оборудования и программ, балансировку нагрузки, обеспечение ресурсов для решения задачи. Облачные сервисы являются комбинацией существующих технологических решений, которые взаимно интегрированы для обеспечения максимального автоматизма и минимизации участия человека в работе комплекса. **Можно выделить основные блоки, которые в первую очередь отличают «облачный» сервис от классического:**

- **Портал самообслуживания** – инструмент, посредством которого пользователь может заказать для себя заранее предопределённый сервис с потенциальным уточнением деталей конфигурации (например, в случае с IaaS, виртуальную машину, уточнив объём требуемой оперативной памяти, число процессорных ядер, размер дискового пространства и т.п.), изменить параметры ранее заказанного сервиса или отказаться от него
- **Каталог сервисов** – список доступных пользователю сервисов и связанные с каждым из сервисов шаблоны их создания, то есть правила, по которым средства автоматизации будут данный сервис конфигурировать на реальном оборудовании и программном обеспечении
- **Оркестратор** – механизм, выполняющий последовательность операций, определённых в шаблоне для каждого сервиса
- **Система тарификации и выставления счетов (биллинга)** – механизм, определяющий объём потреблённых пользователем ресурсов и соотнесение с пользователем соответствующих финансовых затрат.

Одним из основных (но не обязательных) решений для сглаживания неравномерности нагрузки на услуги является размещение **слоя серверной виртуализации между слоем программных услуг и аппаратным обеспечением**. В условиях виртуализации балансировка нагрузки может осуществляться посредством программного распределения виртуальных серверов по реальным, перенос виртуальных серверов происходит посредством живой миграции.

2.2.1. Некоторые особенности облачных технологий

Уточним понятие **облачные технологии** - это взаимосвязанная совокупность операций, технических и программных средств, приспособлений и инструментов, которые предоставляются пользователю, как интернет-сервисы, для реализации своих целей, задач, проектов.

В основном вся отличительная особенность **заключается исключительно в методе формирования, хранения и обработке данных**. Если все операции происходят на вашем компьютере (с использованием его мощностей), то это - не "облако", а если процесс происходит на **сервере в сети**, то этот тренд (услугу) принято называть «облачной технологией».



Как показывает практика, термины «облачные технологии»/«облачный сервис», с их общепринятым графическим представлением, в виде «облаков», только путает пользователей, на самом деле их структуру можно легко понять, если представить ее в виде следующей пирамиды.



Основание пирамиды «**инфраструктура**» – это набор физических устройств (серверы, жесткие диски и т.д.), над ней выстраивается «**платформа**» - набор услуг и верхняя часть пирамиды– **программное обеспечение**, доступное по запросу пользователей.

Также, следует знать, что **облачные вычисления** — это некий базис-вектор, полученный в результате синтеза целого ряда технологий и подходов. Чтобы стало ясно, что имеется ввиду, приведем следующую схему:



2.2.2. Услуги, предоставляемые облачными технологиями

Все, что касается **cloud computing** (далее **CC**), обычно принято называть словом **aaS**. Расшифровывается это так – **"as a Service"**, то есть "как сервис", или "в виде сервиса".

В настоящее время облачные технологии и, собственно, их концепция, предполагает оказание следующих типов услуг своим пользователям:

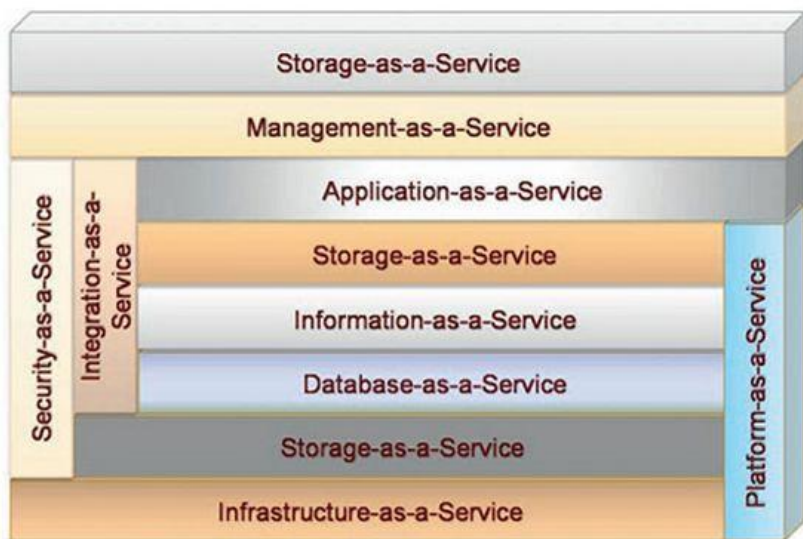
- **Storage-as-a-Service** ("хранение как сервис")
Это, пожалуй, самый простой из **CC**-сервисов, представляющий собой дисковое пространство по требованию. Каждый из нас когда-нибудь сталкивался с ситуацией, когда на мониторе появлялось предупреждение: **"Логический диск заполнен, чтобы освободить место, удалите ненужные программы или данные"**. Услуга **Storage-as-a-Service** дает возможность сохранять данные во внешнем хранилище, в "облаке". Для пользователя, оно будет выглядеть, как дополнительный логический диск или папка. Сервис является базовым для остальных, поскольку входит в состав практически каждого из них. Примером может служить **Google Drive** и прочие схожие сервисы.

- **Database-as-a-Service** ("база данных как сервис")
Database-as-a-Service предоставляет возможность работать с базами данных, как если бы СУБД была установлена на локальном ресурсе. Причем, в этом случае гораздо легче рассредоточивать проекты между разными исполнителями
- **Information-as-a-Service** ("информация как сервис")
Дает возможность удаленно использовать любые виды информации, которая может меняться ежеминутно или даже ежесекундно.
- **Process-as-a-Service** ("управление процессом как сервис")
Представляет собой удаленный ресурс, который может связать воедино несколько ресурсов (таких как услуги или данные, содержащиеся в пределах одного "облака" или других доступных "облаков"), для создания единого бизнес-процесса.
- **Application-as-a-Service** ("приложение как сервис")
Еще, может называется, **Software-as-a-Service** ("ПО как сервис"). Позиционируется как «программное обеспечение по требованию», которое развернуто на удаленных серверах и каждый пользователь может получать к нему доступ посредством Интернета, причем все вопросы обновления и лицензий на данное обеспечение регулируется поставщиком данной услуги. Оплата, в данном случае, производится за фактическое использование последнего. В качестве примера можно привести **Google Docs**, **Google Calendar** и т.п. онлайн-программы.
- **Platform-as-a-Service** ("платформа как сервис")
Пользователю предоставляется компьютерная платформа с установленной операционной системой и некоторым программным обеспечением.
- **Integration-as-a-Service** ("интеграция как сервис")
Это возможность получать из "облака" полный интеграционный пакет, включая программные интерфейсы между приложениями и управление их алгоритмами. Сюда входят известные услуги и функции пакетов централизации, оптимизации и интеграции корпоративных приложений (**EAI**), но предоставляемые как "облачный" сервис.
- **Security-as-a-Service** ("безопасность как сервис")
Данный вид услуги предоставляет возможность пользователям быстро развертывать продукты, позволяющие обеспечить безопасное использование веб-технологий, электронной переписки, локальной сети, что позволяет пользователям

данного сервиса экономить на развертывании и поддержании своей собственной системы безопасности.

- **Management/Governance-as-a-Service** ("администрирование и управление как сервис")
Дает возможность управлять и задавать параметры работы одного или многих "облачных" сервисов. Это в основном такие параметры, как топология, использование ресурсов, виртуализация.
- **Infrastructure-as-a-Service** ("инфраструктура как сервис")
Пользователю предоставляется компьютерная инфраструктура, обычно виртуальные платформы (компьютеры), связанные в сеть, которые он самостоятельно настраивает под собственные цели.
- **Testing-as-a-Service** ("тестирование как сервис")
Дает возможность тестирования локальных или "облачных" систем с использованием тестового ПО из "облака" (при этом никакого оборудования или обеспечения на предприятии, не требуется).

Для наглядности, обобщим все эти сервисы архитектуры «облако», в одну схему за которой и кроются облачные технологии :



Классификация сервисов по типу услуг.

Теперь рассмотрим, какие бывают облачные технологии по форме собственности. Выделяют три категории:

- Публичные
- Частные
- Гибридные.

Кратко о каждой:

- **Публичное облако** — это ИТ-инфраструктура, используемая одновременно множеством компаний и сервисов. Пользователи не имеют возможности управлять и обслуживать данное "облако", а вся ответственность по этим вопросам возложена на владельца ресурса. Абонентом, предлагаемых сервисов может стать любая компания и индивидуальный пользователь. Примерами могут служить онлайн-сервисы: **Amazon EC2, Google Apps/Docs, Microsoft Office Web.**
- **Частное облако** — это безопасная ИТ-инфраструктура контролируемая и эксплуатируемая в интересах одной-единственной организации. Организация может управлять частным "облаком" самостоятельно или поручить эту задачу внешнему подрядчику. Инфраструктура может размещаться либо в помещениях заказчика, либо у внешнего оператора (либо частично у заказчика и частично у оператора).
- **Гибридное облако** — это ИТ-инфраструктура использующая лучшие качества публичного и приватного облака при решении поставленной задачи. Часто такой тип применяется, когда организация имеет сезонные периоды активности, другими словами, как только внутренняя ИТ-инфраструктура не справляется с текущими задачами, часть мощностей перебрасывается на публичное "облако" (например, большие объемы статистической информации), а также для предоставления доступа пользователям к ресурсам предприятия через публичное "облако".

2.2.3. Возможности облачных технологий

Рассмотрим некоторые возможности облачных технологий:

- Доступ к личной информации с любого компьютера, подключённого к Интернету
- Можно работать с информацией с разных устройств (ПК, планшеты, телефоны и т.п.)
- Не важно в какой операционной системе вы предпочитаете работать, - веб-сервисы работают в браузере любых ОС
- Одну и ту же информацию, как вы, так и окружающие, могут просматривать и редактировать одновременно с разных устройств
- Многие программы являются бесплатными (или более дешёвыми) веб-приложениями
- Если что-то случится с вашим устройством (ПК, планшетом, телефоном), то вы не потеряете важную информацию, так как она теперь не хранится в памяти устройств
- Всегда в наличии свежая и обновлённая информация
- Вы всегда пользуетесь самой последней версией программ и при этом не надо следить за выходом обновлений
- Можно свою информацию объединять с другими пользователями
- Легко можно делиться информацией с людьми из любой точки Земли.

2.2.4. Примеры использования облачных технологий

Здесь мы рассмотрим - какие решения, сервисы, программы уже существуют на рынке и на что стоит обратить свое внимание. Начнем с сервисов:

- **iCloud**
Облачный сервис **iCloud** от компании **Apple** (пришедший на смену **MobileMe**), полностью автоматический и бесплатный (хоть и с небольшими функциональными ограничениями). Он сохраняет всевозможный контент (почта, календарь, контакты, документы, музыка, видео и изображения и т.д.) на серверах, а затем доставляет его на все устройства (**iPhone, iPad, iPod touch, Mac и PC**) с помощью беспроводной технологии **Push**.

- **Google Play**

Облачный сервис под названием **Google Play** от «корпорации добра», который предназначен для размещения пользователями кинофильмов, музыки, приложений и книг на специально предназначенных для хранения цифровой информации серверах. Доступ к сервису предоставляется непосредственно из браузера, независимо от ОС, а поэтому может осуществляться как с ПК, так и с мобильных устройств на базе **Android**. У каждого пользователя есть возможность разместить и хранить до **20**-ти тысяч музыкальных записей на бесплатной основе, а также напрямую скачивать на сервер приобретенные в магазинах (**Android Market, Google Music и Google eBookstore**) цифровые товары – кинофильмы, электронные книги, программы, музыкальные треки, как купленные, так и взятые напрокат.

- **OnLive**

Предоставляет возможность играть в современные игры даже на самом простом и слабеньком компьютере. Технически это выглядит следующим образом: сама игра располагается на удаленном сервере и там же производится обработка графики, которая поступает на компьютер к пользователю уже в «готовом» виде. Проще говоря, те вычисления, которые при обычной игре на компьютере выполняют видеокарта, процессор и пр, здесь уже выполнены на сервере, а Ваш компьютер используется лишь как монитор, получающий конечную картинку. Если Вы не поняли, то всё это значит, что автоматически снимаются все проблемы с производительностью компьютера и количеством свободного места на жестком диске, ведь не требуется даже установка. Кроме того, отпадает необходимость платить довольно большие деньги сразу за продукт (игру и т.п.), который Вам не обязательно придется по душе.

- **Xbox Live**

Еще один игровой сервис, который также предоставляет богатую интернет-функциональность и имеет отношение к облачным технологиям. Суть сервиса в том, что обладатели приставок **Xbox 360 и КПК** на базе **Windows Phone 7**, могут играть друг с другом в компьютерные игры и общаться, а также покупать аддоны и различный мультимедийный контент, в онлайн-магазине. Получается, сервис создает некую виртуальную вселенную для геймеров, компоненты которой

расположены не на консолях конечных пользователей, а в облаке.

Таким образом, два последних сервиса предлагают игры как услугу. А теперь представим, что речь идет не об играх, а о программном обеспечении. То есть, Вы платите не за продукт как таковой (за коробку с диском), а за конкретные функции/возможности, которые он Вам предоставляет.

Небольшое примечание

А поскольку нам, как пользователям, больше всего интересно именно программное обеспечение (а не всякие там платформы, как сервис), то сейчас мы и рассмотрим «программный ландшафт» (**SaaS**) облаков. Другими словами, давайте приведем наиболее популярные программные решения, которые, в рамках понятия облачные технологии, собственно, сейчас существуют на рынке.

Собственно, согласно **SaaS**-концепции, как говорилось выше, Вы платите не единовременно, покупая продукт, а как бы берете его в аренду. Причем, используете ровно те функции, которые Вам нужны (и, соответственно, платите за них же). Например, раз в год Вам нужна некая программulina и чаще Вы ее использовать, не собираетесь. Так зачем же покупать продукт, который будет лежать без дела?

И зачем тратить на него место (в квартире, если это коробка с диском, или на винчестере, если это файл)? Правильно, не зачем, ибо есть альтернативный вариант - бесплатный онлайн-сервис (предоставляющий полные функциональные возможности этой программы).

2.2.5. Работа с документами в облачных технологиях

Именно по этому пути и пошли два хедлайнера ИТ-индустрии (а по совместительству еще и конкуренты) — **Google** и **Microsoft**. Обе компании выпустили наборы сервисов, позволяющих работать с документами.

Со стороны **Google** - это их **Google Docs** (ныне **Google Диск**):

Бесплатный онлайн-офис, включающий в себя текстовый, табличный процессор и «приблуду» для создания презентаций, а также интернет-сервис **облачного хранения файлов** с функциями файлообмена.



Это веб-ориентированное программное обеспечение, то есть программа, работающая в рамках веб-браузера без инсталляции на компьютер пользователя, т.е. такая альтернативная версия всяким **Word, Excel** и т.п. без необходимости покупки.

Документы и таблицы, создаваемые пользователем, сохраняются на специальном сервере **Google** или могут быть экспортированы в файл.



Before:



After:



Это одно из ключевых преимуществ программы, так как доступ к введённым данным может осуществляться с любого компьютера подключенного к интернету (при этом доступ защищён паролем).

Со стороны **Microsoft** - это их **Microsoft Office Web Apps**:

Приложения **Microsoft Office Web Apps**, позволяют использовать возможности **Microsoft Office**, через веб-браузер и работать с

документами (причем, не только просматривать их, но и редактировать) непосредственно на веб-сайте, на котором они хранятся.



Таким образом, документы выглядят в браузере точно так же, как в программах **Office**, т.е. полная унификация.

Также стоит отметить, что оба сервиса тесно взаимосвязаны с почтой (**Gmail** в первом случае и **Hotmail** во втором) и файловыми хранилищами, т. е., чтобы воспользоваться **Google Docs**, достаточно завести бесплатный аккаунт гугл и Вы получите набор программ для работы с текстами, электронными таблицами и т.п., прямо в браузере. Для многих, **Google Docs** полностью заменил, как уже и говорилось выше, платный **MS Office**. Если подвести краткий итог (по этим двум сервисам), то можно сказать, что пользователя переводят из привычной ему оффлайн-среды, в онлайн.

2.2.6. Облачные технологии и хранение данных

Не менее популярны и облачные хранилища файлов. Самым известным хранилищем считается..

- **Dropbox.**
У Вас может быть несколько компьютеров, но с помощью этого облачного хранилища можно сделать общую папку с файлами для всех Ваших ПК и даже смартфонов. Тут не придется делать никаких особых действий, ибо операционная система сама будет воспринимать общую папку, как и все остальные папки на винчестере, а дробокс просто займется синхронизацией. Сервис позволяет бесплатно хранить до **2 Гб** данных. Главный акцент в нем делается на синхронизации и обмене информацией. **Dropbox** ведёт историю загрузок, чтобы после удаления файлов с сервера была возможность восстановить данные, плюс ведётся история изменения файлов, которая доступна на период последних **30** дней.
- **Windows Live SkyDrive.**
Сервис **SkyDrive** позволяет сохранять до **7 Гб** (а обмен можно производить файлами до **100 МБ**) информации в упорядоченном с помощью стандартных папок виде. Для изображений предусмотрен режим предпросмотра, а также возможность показать их в виде слайдов. Кроме того, что сервис интегрирован с **Microsoft Office**, он также поддерживает новую операционную систему Windows 8 (а точнее, клиент **SkyDrive** встроен в приложения **Metro** и позволяет загружать в "облако" документы и фотографии за один клик, открывать файлы из удаленного хранилища).
- Ну и конечно **Google Диск**.

К слову, не только всякие офисы и файлохранилища используют облачные технологии. Например, в стане борьбы с цифровой "нечистью" также сделали ставку на облачные вычисления. И вот результат - бесплатный антивирус **Panda Cloud Antivirus**.



Он основан на инновационной технологии "коллективного интеллекта" (которая автоматически выявляет новые угрозы за минимальный промежуток времени) и позволяет свести к минимуму влияние защиты на системные ресурсы компьютера, используя вычислительную мощь облачных технологий для большинства операций: анализ, блокировка и попытки удаления вредоносного ПО.

Сервера антивируса используют информацию, полученную от миллионов пользователей антивирусных продуктов **Panda** по всему миру, для автоматического обнаружения и классификации новых видов вредоносных программ, появляющихся каждый день.

Облачные технологии. Облачно или ясно?

Говоря совсем просто, облако - это возможность всегда иметь гарантированный и безопасный доступ ко всей своей личной информации, а также уход от необходимости держать в своем кармане много лишних вещей (всяких флешек, дисков, проводов и всего такого прочего) или покупать новый компьютер/комплектующие/программы/игры и пр. Несомненно, что на данный момент, облачные технологии являются одной из самых востребованных и интересных тем в ИТ-сфере и всё больше интересных решений, появляющихся в мире, связано именно с ними.

Конечно, обычному пользователю пока сложно в полной мере оценить (и раскрыть) весь их потенциал, но то, что он есть, - видно невооруженным глазом.

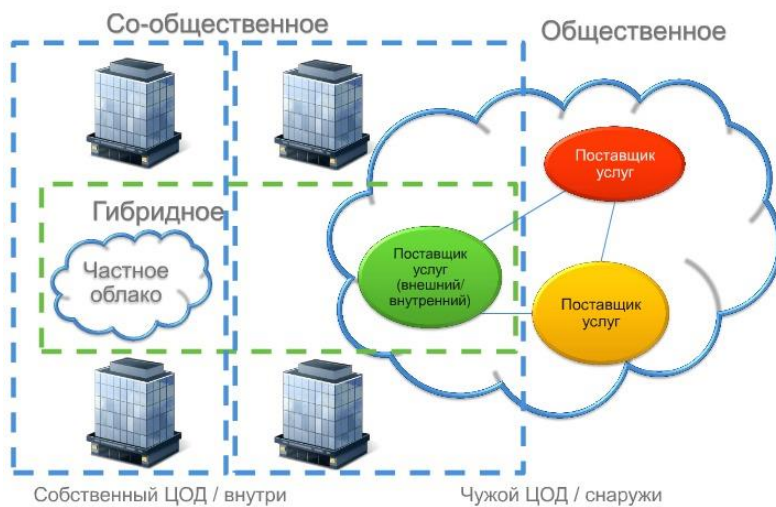
Таким образом, вне всякого сомнения, будущее облачных технологий представляется весьма радужным, ибо такие гиганты (**Microsoft, Apple и Google**) просто так ничего не делают и если они зашли на эту неизведанную территорию, то явно не собираются с неё уходить, ведь сравнительно совсем недавно концепция «**облако**» казалась лишь красивой идеей и смелым экспериментом, а сегодня преимущества облачных технологий могут почувствовать даже те люди, которые не связаны с разработкой программ, веб-технологиями и прочими узкоспециализированными вещами (вышеупомянутые **Xbox Live, Windows Live, OnLive, Google Docs** — яркие тому примеры).

Модели и структуры облачных технологий

4 способа реализации Облака

1. Частное
 - Для себя
2. Со-общественное
 - Для группы
3. Общественное (публичное)
 - Для всех
4. Гибридное
 - Комбинация всех 3-х

Способы реализации Облака



5 характеристик (а), 3 модели (b), 4 способа реализации (с)



3 модели предоставления облачных услуг (сервисов)

1. Cloud Software as a Service (SaaS)
 - Приложения как услуга
2. Cloud Platform as a Service (PaaS)
 - Программная платформа как услуга
3. Cloud Infrastructure as a Service (IaaS)
 - [Информационная] Инфраструктура как услуга



“SaaS”

Аренда



“PaaS”

Разработка

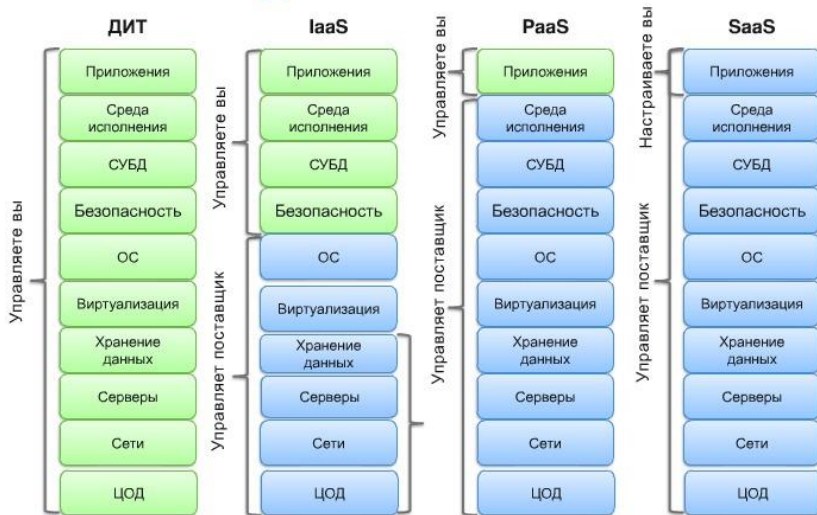


“IaaS”

Миграция

В зависимости от потребностей выбираем необходимую модель облака. Различия моделей, представлены на рисунке ниже

Отличия моделей



ДИТ = ваш департамент информационных технологий

2.3. Основные задачи теории облачных технологических систем

Теория облачных технологических систем (ОТС) — дисциплина, в рамках которой разрабатываются методы решения задач, возникающих при проектировании и эксплуатации ОТС.

Задачи синтеза. Основная задача теории ОТС — задача оптимального синтеза систем, направленная на выбор способа построения системы, наилучшим образом приспособленного для выполнения заданных функций. Исходными в задаче синтеза являются следующие сведения о назначении ОТС: 1) функция системы, обычно представляемая перечнем прикладных задач, решение которых возлагается на систему; 2) перечень ограничений на характеристики системы, например ограничений на время решения задач, производительность системы или стоимость оборудования; 3) критерий эффективности, устанавливающий способ оценки качества системы в целом. Исходя из этих сведений, необходимо определить структуру системы (состав устройств и связей между ними) и стратегию управления технологическими (вычислительными) процессами, которые должны удовлетворять заданным ограничениям на характеристики системы и быть оптимальными в смысле назначенного критерия эффективности. Качество ОТС систем наиболее существенно зависит от затрат оборудования, используемого в аппаратурной части системы, и затрат времени на решение прикладных задач, т. е. затраты оборудования и времени являются основными показателями качества систем. Затраты оборудования определяются в основном структурой системы, т. е. типами и количеством устройств и способом их связи (взаимодействия). Затраты времени на решение задач зависят как от структуры системы, так и от стратегии управления вычислительными процессами. Проектирование ОТС сводится к выбору такой структуры и стратегии управления, которые обеспечивают реализацию заданных функций при затратах оборудования и времени, лимитируемых заданными ограничениями и критерием эффективности.

Чтобы выявить круг вопросов, возникающих при постановке и решении задачи оптимального синтеза ОТС, опишем задачу синтеза с использованием математической символики.

Охарактеризуем объекты (алгоритмы, структуры, системы) совокупностями присущих им свойств, каждое из которых может быть измерено, т. е. оценено количественно. Некоторые свойства объекта будем рассматривать как первичные и характеризовать совокупность таких свойств вектором $V=(v_1, ..., v_M)$, составляющие которого являются оценками соответствующих свойств и называются *парамет-*

рами объекта. Остальные свойства будем относить к вторичным и характеризовать их вектором $Y=(y_1, \dots, y_N)$, составляющие которого называются *характеристиками объекта*. Вторичность свойств, характеризующих величинами $y_1, \dots, y_N$, обусловлена тем, что значения $y_n, n = 1, \dots, N$, могут быть определены как функции параметров объекта, т. е. $y_n = \varphi_n(V)$. Например, количество операторов, выполняемых за одну реализацию алгоритма, определяется операторной схемой алгоритма (составом операторов и связей между ними) и законом распределения исходных величин, обрабатываемых алгоритмом. Следовательно, количество выполняемых операторов является характеристикой алгоритма, значение которой однозначно определяется параметрами, характеризующими операторную схему алгоритма и распределение исходных величин.

Предположим, что к параметрам причислены свойства, учет которых необходим для определения всех интересующих нас характеристик объекта. В таком случае вектор параметров V однозначно характеризует объект, выделяя его из множества объектов такой же природы. Различие объектов A и B эквивалентно неравенству соответствующих им векторов V_A и V_B , и объекты A и B считаются одинаковыми, если $V_A = V_B$.

Многочисленность объектов, с которыми можно иметь дело, отображается следующим образом. Пусть отдельный параметр $V_m, m=1, \dots, M$, принимает различные значения из некоторого множества значений x_m , называемого *областью определения параметра V_m* . В таком случае множество $X=x_1 \times \dots \times x_M$, являющееся декартовым произведением множеств $x_1, \dots, x_M$, содержит в себе всевозможные векторы $V = (V_1, \dots, V_M)$ и представляет всевозможные объекты, с рассматриваемыми свойствами. Множество X называется *областью определения объектов типа V* .

Используя эти понятия, ОТС можно характеризовать следующим набором параметров и характеристик:

$P=(p_1, \dots, p_Q)$ — вектор параметров, представляющий класс прикладных задач, решение которых является функцией системы, с областью определения P ;

$S=(s_1, \dots, s_T)$ — вектор параметров структуры системы, определенный на множестве возможных структур S ;

$C=(c_1, \dots, c_D)$ — вектор параметров стратегии управления технологическими (вычислительными) процессами с областью определения C ;

$Y=(y_1, \dots, y_N)$ — вектор характеристик ОТС, составляющие которого суть функции параметров P, S и C ;

$E = \Psi(Y)$ — функция, определяющая значение критерия эффективности как интегральной характеристики качества системы.

Поскольку $Y = \Phi(P, S, C)$, то критерий эффективности является функцией параметров системы, т. е. $E = \Omega(P, S, C)$.

Используя приведенные обозначения, задачу синтеза ОТС можно описать следующим образом: найти $\max_{S \in S, C \in C} \Omega(P, S, C)$ при $y_i \in y_i^*$;

$i = \alpha, \dots, \omega$; $\alpha, \dots, \omega \in \{1, \dots, N\}$, т. е. определить такие параметры структуры S и стратегии управления C , которым соответствует максимум (минимум) прямого (инверсного) критерия $E = \Omega(P, S, C)$ при выполнении ограничений $y_\alpha \in y_\alpha^*, \dots, y_\omega \in y_\omega^*$ на характеристики системы. Здесь $\alpha, \dots, \omega$ — индексы, выделяющие характеристики, которые должны принадлежать заданным областям $y_\alpha^*, \dots, y_\omega^*$. Решение задачи синтеза сводится к подбору параметров S и C , оптимизирующих значение критерия эффективности системы. При оптимизации выбор направления изменения значений параметров S и C проводится исходя из характера зависимостей $E = \Psi(Y)$ и $Y = \Phi(P, S, C)$, определенных моделью ОТС. Результатом решения задачи синтеза являются значения параметров S и C , характеризующих структуру системы и стратегию управления технологическими (вычислительными) процессами.

Чтобы поставить задачу синтеза конкретно, требуется выявить множество параметров, необходимых и достаточных для определения систем как конкретных объектов, и зависимости между характеристиками системы и параметрами задач, структур и стратегий управления, т. е. необходимо **располагать моделью, отображающей свойства реальных систем в математической форме**. Модель ставит в соответствие каждому набору параметров P, S и C значения характеристик $y_1, \dots, y_N$, соответствующие характеристикам, которые имела бы реальная система с такими же параметрами.

Многообразие сфер применения ОТС, возможных структур и стратегий управления Технологическими (вычислительными) процессами порождает практически бесконечное число вариантов построения ОТС, для описания которых требуется большое число параметров. Многочисленность параметров и сложность зависимостей между ними и характеристиками систем приводят к тому, что задача синтеза в ее общей постановке неразрешима.

Задача синтеза упрощается, если ограничить рамки ее использования определенным классом систем — систем с подобными функциями и требованиями к качеству функционирования. При таком подходе к синтезу уменьшается число структур и стратегий управления, которые следует рассматривать при поиске оптимальных решений.

Правомочность указанного подхода к синтезу систем очевидна. Поэтому задачу синтеза будем понимать как задачу синтеза систем определенного класса — систем пакетной обработки, с разделением времени и т. д.

Задача синтеза максимально упрощается, если, ориентируясь на конкретный класс функций системы и требования к качеству функционирования, выбрать для построения системы один класс структур и определенную стратегию управления технологическими (вычислительными) процессами. Такой выбор возможен, если есть сведения о свойствах структур и стратегий управления в отношении к классам прикладных задач.

Задачи анализа. Важная задача теории ОТС — анализ ОТС с целью качественной и количественной оценки свойств различных классов прикладных задач, структур и стратегий управления технологическими (вычислительными) процессами.

Для задач анализа характерны **три этапа проведения исследований.**

На первом этапе необходимо выявить **причинно-следственные связи**, присущие анализируемому объекту, и построить концептуальную модель объекта, вскрывающую сущность происходящих в нем процессов. При построении концептуальной модели устанавливается факт наличия зависимости между представляющими интерес характеристиками процессов и параметрами объекта. Эти параметры должны непременно присутствовать в модели.

На втором этапе, на базе принятой концептуальной модели, строится математическая модель, выявляющая количественные отношения между характеристиками и параметрами. Эти отношения представляются в форме функциональных зависимостей $Y = \Phi(X)$, где Y — множество характеристик и X — множество параметров, учитываемых концептуальной моделью. Количественные отношения конкретизируют причинно-следственные связи и тем самым полностью определяют модель. Исследование зависимостей $Y = \Phi(X)$ позволяет выявить представляющие интерес свойства объекта: степень влияния параметров на характеристики объекта, предельные и экстремальные значения характеристик, взаимные отношения между характеристиками и т. п. Поскольку построение модели производится неформальными методами, то возникает необходимость проверки достоверности модели и полученных на ее основе оценок, что и осуществляется на **третьем этапе** решения задачи анализа. Проверка достоверности проводится сопоставлением зависимостей, полученных из модели, с экспериментальными данными или данными, получаемыми другими методами анализа.

Результатом анализа, проводимого в рамках теории ОТС, являются модели процессов, происходящих в ОТС, и закономерности, присущие процессам и системам. Модели раскрывают причинно-следственную природу процессов и устанавливают зависимости между их характеристиками и параметрами систем. В этом — познавательная ценность анализа. Прикладная ценность анализа обусловлена использованием результатов анализа для постановки задач синтеза, возникающих при проектировании ОТС.

Таким образом, **теория ОТС начинается с анализа свойств алгоритмов, различных стратегий управления технологическими (вычислительными) процессами, способов организации ОТС в целом.** При этом строятся и исследуются модели вычислительных процессов, которые имеют место в системах, реализующих различные классы прикладных задач на основе различных структур и стратегий управления вычислительными процессами. Результаты анализа способствуют пониманию сущности процессов, происходящих в ОТС, и используются для постановки задач синтеза. В отличие от задач анализа где характеристики процессов определяются как функции параметров системы, при синтезе решается задача выбора параметров системы, при которых удовлетворяются заданные требования к характеристикам **технологических** (вычислительных) процессов. Одно из требований состоит в обеспечении экстремальности критерия эффективности, и, следовательно, решение задачи синтеза сводится к оптимизации системы по заданному критерию эффективности с учетом ограничений, которые могут быть наложены на некоторые ее характеристики и параметры.

2.4. Общая характеристика методов теории облачных технологических систем

Выбор методов, используемых в теории ОТС, предопределяется характером процессов в ОТС и целями, которые ставятся при проведении исследований.

Вероятностный подход к исследованию ОТС. Процесс функционирования ОТС — это процесс выполнения совокупностью устройств некоторого набора операций (алгоритмов). В теории ОТС технологические процессы в системах изучаются в основном во временном аспекте — с целью определения временных характеристик процесса функционирования систем: доли времени, в течение которого устройства простаивают; времени получения каких-то результатов, производимых соответствующей процедурой; произво-

длительности, определяемой числом задач, решаемых в единицу времени, и т. п. Для ОТС типично наличие элемента случайности в порядке инициирования процедур, реализуемых системой. Так, во-первых, запросы на выполнение процедур генерируются пользователями в случайные моменты времени; пакет программ, предназначенных для обработки, состоит из случайным образом сформированной смеси задач и т. д. Во-вторых, хотя процесс выполнения процедур по своей сути детерминированный, однако практическая бесконечность числа различных технологических процессов, порождаемых алгоритмом в зависимости от исходных данных, не позволяет выявить представляющие интерес свойства технологических процессов при исследовании их с позиций детерминизма.

Многочисленность вариантов выполнения процессов является фактором, предопределяющим необходимость исследования технологического процесса как процесса случайного. Такой подход правомочен тогда, когда свойства изучаемого объекта проявляются на множестве процессов, например на множестве реализаций программы или совокупности программ. Таким образом, случайный характер инициирования процессов, практическая непредсказуемость данных, обрабатываемых программой, и связанная с этим случайность моментов времени, в которые начинает выполняться и заканчивается процесс, приводят к необходимости рассматривать процесс функционирования ОТС как случайный процесс и изучать свойства ОТС с позиций теории вероятностей. В связи с этим алгоритмы, несмотря на присущую им определенность, интерпретируются как генераторы случайных процессов, порождающие в случайные моменты времени запросы на передачу и обработку информации, и фактор случайности включается в модели алгоритмов. Итак, функционирование ОТС можно рассматривать как **композицию трех классов** в общем случае недетерминированных процессов:

- 1) на входе ОТС, порождающих заявки на выполнение определенных алгоритмов (процедур);
- 2) выполнения процессов, порождающих заявки на передачу и обработку информации;
- 3) обслуживания заявок устройствами, обеспечивающими передачу и обработку информации.

Изучение процессов, связанных с обслуживанием какого-либо вида запросов, с учетом случайного характера поступления запросов и обслуживания проводится в рамках теории массового обслуживания, являющейся разделом прикладной математики. Аппарат теории

массового обслуживания — основной в теории ОТС, широко используемый для постановки и решения задач анализа ОТС. Большинство моделей теории ОТС строится на основе моделей теории массового обслуживания. В теории массового обслуживания изучаются процессы, являющиеся в основном марковскими либо некоторым образом связанные с марковскими процессами. Поэтому для решения задач теории массового обслуживания используется аппарат теории марковских процессов.

В теории ОТС применяют аналитические, числовые и экспериментальные методы исследования.

Аналитические методы. *Аналитические методы* состоят в преобразовании символьной информации, записанной на языке математического анализа. При использовании аналитических методов строится математическая модель объекта, представляющая физические свойства объекта в виде математических объектов и отношений (математических операций над величинами), например в виде дифференциальных или интегральных уравнений. Математическая модель строится на основе понятий, символики и методов некоторой теории, например теории массового обслуживания, определяющей класс математических аналогий, т. е. основополагающих математических моделей. При аналитическом подходе требуемые зависимости выводятся из математической модели последовательным применением математических правил. Препятствием при этом может быть неразрешимость уравнений в аналитической форме, отсутствие первообразных для подынтегральных функций и т. п., с чем приходится сталкиваться весьма часто. Поэтому лишь при определенных свойствах модели можно получить решение в явной аналитической форме. Несмотря на ограниченные возможности аналитического подхода, решения, полученные в явной аналитической форме, имеют большую познавательную ценность и находят результативное применение при решении широкого класса теоретических и прикладных задач.

Численные методы. *Численные методы* основываются на построении конечной последовательности действий над числами, приводящей к получению требуемых результатов. При наличии математической модели исследуемого объекта применение численных методов сводится к замене математических операций и отношений соответствующими операциями над числами: замене интегралов суммами, производных разностными отношениями, бесконечных сумм конечными и т. д. В результате этого строится алгоритм, позволяющий точно или с допустимой погрешностью определить значения требуемых величин. Алгоритм реализуется вручную или

программируется для ЭВМ. Результат применения численных методов — таблицы (графики) зависимостей, раскрывающих свойства объекта. Численные методы по сравнению с аналитическими позволяют решать значительно более широкий круг задач.

Характер процессов, присущих исследуемому объекту и подлежащих отображению в модели, может быть столь сложным, что построение математической модели превращается в трудную задачу, а анализ модели даже численными методами может оказаться нерезультативным из-за трудоемкости или неустойчивости алгоритмов в отношении погрешностей аппроксимации и округления.

Воспроизведение в модели динамики сложных пространственно-временных отношений между объектами, образующими систему, всего многообразия ее связей со средой, действующих в системе законов управления, адаптивных свойств и черт целенаправленного поведения трудно выполнимо чисто математическими средствами. При исследовании таких систем широкое применение находят модели, представляющие собой содержательное описание объектов исследования в форме алгоритмов. В описаниях отражаются как структура исследуемых систем, что достигается отождествлением элементов систем с соответствующими элементами алгоритмов, так и процессы функционирования систем во времени, представляемые в логико-математической форме. При этом описания объектов исследования имеют алгоритмический характер, а сами модели суть программы для ЭВМ. Модели такого типа называют *имитационными* или *алгоритмическими*.

Главная особенность данного подхода к моделированию заключается в том, что используемые для построения модели алгоритмические языки — гораздо более гибкое и доступное средство описания сложных систем, нежели язык математических функциональных соотношений. Благодаря этому в имитационных моделях сложных систем находят отражение многие детали их структуры и функций, которые вынужденно опускаются или произвольно утрачиваются в математически строгих моделях. Свойственная имитационным моделям реалистичность основывается на использовании для их построения всех имеющихся представлений об объекте исследования как теоретического, так и эвристического характера.

В теории ОТС с присущим ей вероятностным подходом к анализу систем при построении имитационных моделей наиболее широко используется *метод статистических испытаний (метод Монте-Карло)*. Процедура построения и анализа имитационных моделей методом статистических испытаний называется *статистическим моделированием*. Статистическое моделирование представляет собой

процесс получения статистических данных о процессах, происходящих в моделируемой системе. Для получения представляющих интерес результатов статистические данные обрабатываются и классифицируются с использованием методов математической статистики. Позитивное свойство статистического моделирования — универсальность, гарантирующая принципиальную возможность анализа систем любой степени сложности с любой степенью детализации изучаемых процессов. Негативное свойство статистического моделирования — трудоемкость процесса моделирования, т. е. необходимость выполнения очень большого количества операций над числами, и частный характер результатов, не раскрывающий зависимости, а лишь определяющий ее в отдельных априорно назначенных точках.

Экспериментальные методы. *Экспериментальные методы* базируются на измерении характеристик технологических процессов, происходящих в реальных системах, и обработке результатов измерения с целью выявления представляющих интерес зависимостей. Экспериментальные исследования являются источником наиболее достоверной информации, однако получаемые при этом результаты носят частный характер. Эти методы — пока единственный источник сведений о характеристиках задач, решаемых в вычислительных центрах, процессах взаимодействия пользователей с ОТС и т. д. Данные сведения используются в качестве исходных данных при проектировании новых ОТС. Естественно, что возможности экспериментальных методов ограничиваются парком - существующих ОТС.

При проведении исследований даже узкого круга вопросов возникает необходимость в использовании нескольких методов решения одной и той же задачи. Решение задач аналитическими методами всегда предполагает использование допущений, принимаемых либо на этапе построения концептуальной модели, либо на этапе построения математической модели, т. е. вывода результирующих зависимостей. Получаемые при этом зависимости могут быть использованы для решения прикладных задач только после оценки методической погрешности, вносимой принятыми допущениями. Оценить погрешность аналитическими методами — задача обычно более сложная, чем вывод зависимостей. Поэтому для оценки методической погрешности широко используется метод имитационного моделирования, позволяющий получить численное решение с любой заданной точностью. В таком же аспекте используются и экспериментальные методы анализа ОТС.

Методы оптимизации. Задачи синтеза ОТС решаются с использованием численных *методов оптимизации*, иначе называемых *методами математического программирования*. Методология постановки и решения экстремальных задач на основе математических и имитационных моделей систем изучается в общей постановке в рамках *исследования операций*.

3. Технологические процессы и их отличительные признаки

3.1. Система преобразований и ее модель

Когда мы пытаемся сформулировать ответ на вопрос о применении теории облачных технологических систем, то на передний план выступают проблемы, связанные с назначением, целями СТС: для чего предназначаются облачные технологические системы типа «процесс» и технические системы типа «объект».

Человек, как и всякое живое существо, имеет потребности, выполнение которых необходимо для поддержания его жизни. Он нуждается в пище, защите от опасности и непогоды, он должен иметь возможность лечиться в случае болезни. Однако в отличие от других живых существ потребности и желания человека постоянно растут; это связано с развитием цивилизации, прогрессом техники и ростом благосостояния. Повышенные притязания вызывают проблемы их удовлетворения, так как для этого недостаточно физических сил и возможностей органов чувств человека. Поэтому человек ищет решения, позволяющие устранить эти проблемы.

Для удовлетворения своих потребностей человек находит в природе лишь некоторые полноценные средства. В большинстве же случаев он вынужден действовать на эти средства бесчисленными способами: изменять, наращивать, комбинировать. Он должен резать, варить, солить, плавать, разлагать, транспортировать, хранить, преобразовывать и использовать энергию, строить, разрушать, взвешивать, считать и многое другое.

В результате осуществления этих процессов свойствам операнда придается дополнительная ценность, по мере того как операнд трансформируется и достигает состояния, достаточного для удовлетворения потребности.

Назовем процессы такого рода **преобразованиями (отображениями)** и определим некоторые понятия и признаки. Для наглядности и

полноты систематизации на рис. 3.1 представлены все типы элементов и действий, входящих в модель системы преобразований.

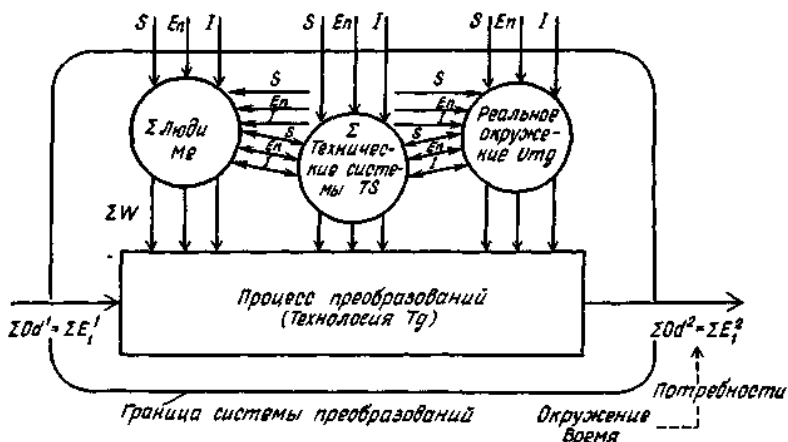


Рис 3.1. Модель системы преобразований.

Примеры таких систем приведены в табл. 3.1.

Таблица 3.1. Примеры систем преобразований

Преобразование	Технология	Действия	Участие операторов в действии		
Стальная деталь Мягкая → твердая	Закалка	Разогрев (в кузнечном горне) Охлаждение (в масляной ванне)	Рабочие (выдержка в горне, операции закалки)	Кузнечный горн, ванна с маслом	Воздух ← Тепло ← Тепло →
Производство зерна Мало → много	Выращивание в поле	Вспашка Высевание семян Боронование Культивация Уничтожение сорняков Внесение удобрений Полив Периодический контроль состояния посевов Жатва Транспортировка Хранение	Тракторист, сельщик, агроном, комбайнер, шофер, другие сельскохозяйственные рабочие	Трактор, плуг, сеялка, борона, культиватор, машины для внесения удобрений и гербицидов, дождевальная установка, комбайн, грузовой автомобиль, элеватор	Земля, солнечное излучение, осадки

Материалы → автомобиль	Конструирование и производство	Эскизное проектирование Расчеты Рабочее проектирование Выбор технологии Комплектация оборудования Материально-техническое снабжение Изготовление деталей Сборка Испытания Регулировка	Конструктор, расчетчик, технолог, снабженец, станочник, сборщик, испытатель, контролер, администратор	Справочники, технические проспекты, чертежные принадлежности, материалы, станки, измерительные инструменты, прессы, сборочный конвейер, система контроля	Цех (освещение, отопление, вентиляция, водопровод, канализация, и т. п.)
Транспортировка людей на дальние расстояния	Полет	Строительство аэропортов Приобретение самолетов Организация рейсов	Проектировщик, инженер, техник, рабочий, администратор, пилот, кассир, диспетчер	Аэровокзал, автобус, самолет	Взлетно-посадочная полоса, метеослужба, воздушный коридор

Основой для построения этой модели являются следующие *предположения*, вытекающие из опыта людей:

- желаемые преобразования операнда (объекта преобразования) достигаются целенаправленными воздействиями материального (S), энергетического (En) или информационного (I) типов;
- эти три типа воздействий при любом преобразовании осуществляются людьми ($\sum Me$), техническими системами ($\sum TS$) и окружением (Umg).

В рамках глобальной системы, включающей Землю, человеческое общество и, в перспективе, элементы Солнечной системы, может быть создано почти бесконечное множество систем преобразований.

Отдельный элемент часто входит в несколько систем преобразований, например человек, который ест в вагоне-ресторане, листает газету и одновременно обсуждает новости с соседями по столу.

Такая интерпретация модели преобразований позволяет сделать следующие выводы.

1. Для удовлетворения потребности людей либо выбирается требуемый объект, либо задается требуемое состояние операнда. Это состояние (Od^2) является целью преобразования.
2. Операндами преобразований могут быть живые существа, в частности люди, а также материальные, энергетические и информационные объекты.
3. Выбирают подходящее начальное состояние, операнда Od^1 в качестве входной величины (или оно задается). Состояние Od^2 может быть достигнуто из нескольких начальных состояний Od^1 .
4. Изменение $Od^1 \rightarrow Od^2$ называется преобразованием (отображением).
5. Преобразование вызывается либо неудовлетворительным состоянием Od^1 , либо потребностью в Od^2 .

6. Преобразование выполняется на основе некоторой **технологии, представляющей собой упорядоченную совокупность целенаправленных частичных изменений (действий)**. Состояние операнда Od^2 может быть достигнуто различными способами (технологиями).

7. Преобразование (как общее, так и частичное) осуществляется путем материального, энергетического или информационного воздействия на операнд.

8. Воздействие осуществляется тремя системами-операторами: людьми, техническими системами и реальным окружением. Все названные операторы имеют связи (материального, энергетического и/или информационного типа) между собой.

Модель охватывает, как это показано в табл. 3.1, все виды преобразований операндов всех типов при любом варианте участия систем-операторов в воздействиях.

При более детальном анализе «внутренних» операторов выявляются элементарные процессы (действия), которые также можно назвать преобразованиями. Так, в технических системах вращательное движение преобразуется в поступательное, наращивается сила и осуществляются другие изменения. Как видно из модели, целью действий являются определенные изменения. Зависимость характера этих изменений от цели станет яснее при рассмотрении *процессов и цепочек действий*.

Между процессом действия и процессом преобразования существует причинная связь, а именно изменения операнда в процессе преобразования (ΣE) вызываются действиями (ΣW) технической системы как причины. С другой стороны, эта причина (действие системы) является следствием цепочки действий в системе, инициируемых входными воздействиями на систему.

3.2. Элементы системы преобразований

Элементы системы преобразований показаны на рис. 3.1. Дадим еще несколько пояснений, относящихся к отдельным элементам.

Операнд — это пассивный элемент рассматриваемой нами системы. Он может относиться к любой из четырех названных выше категорий (люди, материя, энергия, информация) либо их комбинации. Состояние операнда может быть определено как совокупность свойств либо как совокупность операндов — составных частей технической системы. В последнем случае речь идет о преобразованиях типа объединения или разделения.

Так как для нас представляют интерес преобразования, называемые **облачными технологическими процессами**, в которых основная роль

отводится техническим системам, процессы такого рода будут рассмотрены далее более подробно.

Люди как операторы системы преобразований могут быть определены как подмножество людей, включающее только тех, кто выполняет какой-либо вид действий для определенного преобразования.

Технические системы как операторы системы преобразований — это подмножество технических систем, выполняющих какого-либо вида действия для определенного преобразования. Технические системы будут подробно рассмотрены дальше. Необходимо отметить те поразительные масштабы технических систем, которые зачастую необходимы для выполнения преобразований. Например, для такого преобразования, как изменение местонахождения, недостаточно только транспортного средства, а необходимы также система улиц (дорог), мосты, система управления движением, автозаправочные станции, деньги и т. п. **Только при наличии всех этих операторов преобразование может быть осуществлено.**

Оператор *реальное окружение (среда)* охватывает все источники внешних воздействий в ближайшем окружении процесса преобразования, которые в большинстве случаев невозможно указать точно. В реальное окружение мы будем включать только те элементы окружения (среды), которые имеют связи с элементами системы преобразований: геосферу (а именно, сушу и воду), биосферу, техносферу, атмосферу и климат (погоду).

Особенно важным для технических систем (а также для людей) является отношение с системой *геосфера*, без которой они не могут функционировать. Сюда можно отнести воздействующую среду — воду для корабля, воздух для самолета. При перемещениях в пространстве приобретает значение Солнечная система как часть реального окружения технической системы.

Биосфера включает все организмы (людей, животных и растения) и места их обитания. Из гео-, био- и атмосферы могут быть построены различные *экосистемы*. В них в рамках определенных отношений происходят необходимые для жизни процессы преобразования материи, энергии и информации (рис. 3.2).

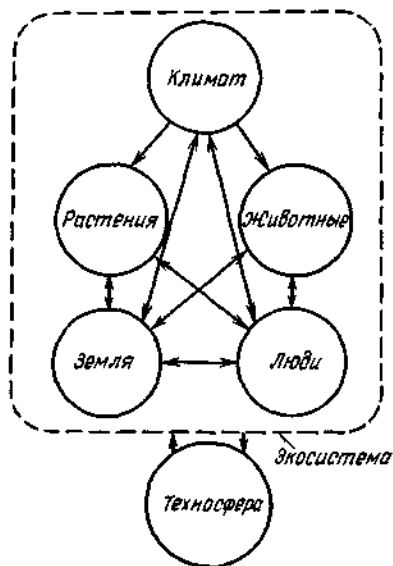


Рис. 3.2. Экосистема и техносфера

При разработке и осуществлении любых технических решений следует не нарушать равновесия этих систем.

Техносфера включает все технологические и технические системы, созданные людьми.

Подытожим полученные выше результаты в виде следующих выводов.

1. Человек и человеческое общество имеют многообразные потребности в зависимости от целей, которые они перед собой ставят. Любая потребность формулируется как совокупность требований к объекту в определенном состоянии. Объект рассматривается как средство достижения цели. Если желаемая цель не может быть достигнута непосредственно, то появляются промежуточная цель (подцель) и потребность изменений, которые позволят в конечном счете нужную цель (состояние) достигнуть.
2. Изменение состояния называется преобразованием, а объект воздействия — операндом.
3. Преобразование — это искусственный процесс, в котором желаемое изменение достигается путем целенаправленного использования природных явлений.

4. Необходимость преобразования вызывается неудовлетворительным исходным состоянием операнда или потребностью в выходном состоянии операнда как желательного средства для достижения цели.
5. Операндами процессов преобразований могут служить биологические системы (живые существа), материя, энергия и (или) информация.
6. Преобразование выполняется на основе технологии, определяющей систему частичных преобразований. Вообще для выполнения некоторого преобразования могут быть разработаны разные технологии, основанные на различных принципах.
7. Преобразования, в том числе частичные, реализуются определенными действиями. Таким образом, действия — это средства осуществления преобразований.
8. Действие осуществляется тремя системами-операторами: людьми, техническими системами и реальным окружением.
9. Система преобразований — это некоторое отношение на множестве всех участвующих в преобразовании элементов. Общая модель такой системы представлена на рис. 3.1.
10. Действия осуществляются операторами в рамках процессов действия (цепочек действия), превращающих вход оператора в выход системы преобразований.
11. Между преобразованием (следствие) и действием (причина) существуют причинные отношения.

3.3. Технологический процесс и его модель

Ранее технологическим процессом были названы такие преобразования, в которых роль операторов выполняют наряду с людьми и технические системы.

Технологические процессы занимают центральное место в таких системах преобразований, поскольку преобразования в них могут быть осуществлены не иначе, как посредством технологических процессов. Кроме того, элементы системы преобразований могут быть объединены лишь с использованием технологических процессов. Современный мир изобилует технологическими процессами, в которых, так или иначе, участвуют люди. Каждый человек обычно участвует в нескольких процессах одновременно, удовлетворяя свои потребности непосредственно либо создавая средства для последующего удовлетворения своих и чужих потребностей. Ключевым словом здесь является *потребность*.

Какова связь потребности с технологическим процессом? Повторим: а) сначала возникает состояние неудовлетворенности либо ситуация,

которую человек хочет или должен изменить; б) это состояние вынуждает сформулировать потребность; в) осознается, что потребность, вообще говоря, может быть удовлетворена различными средствами; г) выясняется, что средства, которые могли бы быть использованы непосредственно, отсутствуют; д) нужно изменить существующее состояние на желаемое путем преобразований; е) преобразования осуществляются в технологических процессах. Мы установили, что технологический процесс (ТР) является элементом системы преобразований (рис. 3.1.). Его модель строится на отношениях в системе преобразований (рис. 3.3, табл.3.2).

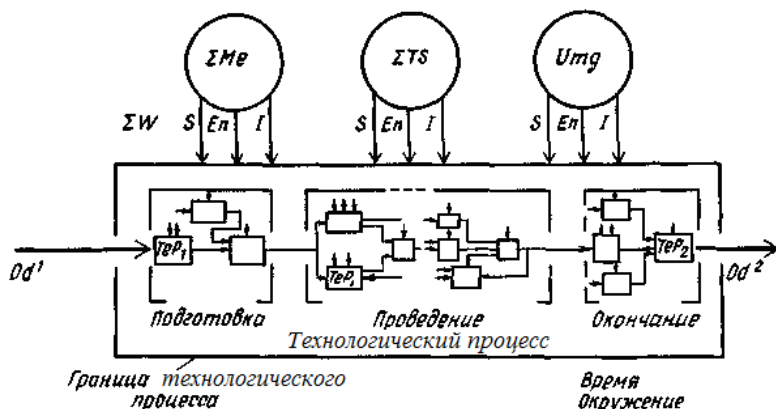


Рис.3.3. Структура модели технологического процесса.

Таблица 3.2. Примеры к модели технологического процесса

При- мер	Операнд			Способ преобразования	Действия		
	Вид	Состояние		Технологический принцип $TgPz$ Подпроцесс TeP	над материалами, энергией и информацией		
		вход- ное (1)	выход- ное (2)		$\{S$	$\{Ea$	$\{I$
1	Сталь Ст50	Заго- товка	Де- таль	$TgPz_1$: обточка TeP_1 : установить и зажать заготовку TeP_2 : включить об- роты TeP_3 : перемещать ре- зец относительно за- готовки со снятием стружки TeP_4 : измерять TeP_5 : освободить, де- таль положить в кон- тейнер готовой про- дукции	TS TS TS TS	Me TS TS Me Me	Me Me Me $Me + TS$ Me
2	Энер- гия	Уголь	Теп- ловая энер- гия	$TgPz_1$: сжигание угля $S + O_2 = CO_2 + Q$ TeP_1 : уголь загрузить в топку и зажечь TeP_2 : сжигать, обес- печивая подвод воз- духа и отвод тепла TeP_3 : шлаки и золу убирать	TS TS	$Me + TS$ TS Me	Me Me Me

Продолжение

При мер	Операнд			Способ преобразования	Действия		
	Вид	Состояние		Технологический принцип $TgPz$ Подпроцесс TeP	над материалами, энергией и информацией		
		вход- ное (1)	выход- ное (2)		$\downarrow S$	$\downarrow En$	$\downarrow I$
3	Человек	Нефункциональная почка	Нормальная работа почек	$TgPz_1$ трансплантация искусственной почки TeP_1 : подготовить пациента и медперсонал к операции TeP_2 : больную почку удалить TeP_3 : трансплантировать искусственную почку TeP_4 : контролировать процесс выздоровления	TS TS TS	$Me + TS$ Me Me Me	$Me + TS$ $Me + TS$ $Me + TS$ $Me + TS$
4	Вода	В источнике	В резервуаре	$TgPz_1$: перекачивание воды насосом TeP_1 : включить насос TeP_2 : контролировать время работы насоса и использование резервуара TeP_3 : выключить насос	TS TS TS	TS TS TS	Me Me Me
5	Человек	Место жительства	Место работы	$TgPz_1$: поездка на работу автобусом TgP_1 : прийти на остановку и сесть в подошедший автобус TeP_2 : ехать до места работы TeP_3 : сойти с автобуса на остановке у места работы и прийти на рабочее место	TS	Me TS Me	Me $Me + TS$ Me

Обобщенный технологический процесс определяет преобразование операнда без точного указания того, «чем», «кто», «когда» и «где» его выполняет.

Описание любого технологического процесса должно содержать ответы на следующие вопросы:

- а) Что является операндом и каковы его состояния (начальное, конечное и промежуточные)?
- б) С помощью каких преобразований (технологии) достигается совокупное преобразование $Od^1 \rightarrow Od^2$ в рамках существующих условий — природных явлений, общественных законов и других ограничений?
- в) Какими действиями (материального, энергетического или информационного типа) могут быть реализованы частичные преобразования (хотя бы в первом приближении без детализации)?
- г) Какими операторами выполняются отдельные действия?

Примеры, приведенные в табл.3.2, содержат ответы на поставленные вопросы. В каждом конкретном случае информация такого рода вводится непосредственно в графическое изображение технологического процесса, и тогда эти связи выступают еще отчетливее.

Предполагая, что желаемое состояние является выходом (результатом) технологического процесса, для этого процесса нужно определить следующие показатели, являющиеся его характерными признаками:

- конечное состояние операнда;
- технологический принцип;
- типы и последовательность операций (действий), соответствующих выбранной технологии;
- результат каждой операции;
- распределение результатов по операциям, соответствующее постановке задачи и требуемому конечному результату.

Указанные признаки можно рассматривать как степени свободы, имеющиеся при выборе, разработке и оптимизации технологического процесса.

Пояснения к модели

Для понимания общей модели ТР и разработки конкретного технологического процесса могут быть полезны следующие пояснения.

1. Принципиальная схема технологического процесса определяется операндами, состояние которых требуется изменить. Последовательные входы и выходы технологического процесса должны быть однородными. Не следует нарушать это правило, пытаясь, например, построить технологический процесс, в котором вход — это горючее, а выход — поездка.

2. Выход может быть сформулирован только из тех изменяемых или не изменяемых в технологическом процессе составных частей, которые поступили в процесс как входы. Кроме того, должен быть определен полностью перечень операндов процесса. Если такой перечень неполон (например, растворимого кофе и воды недостаточно для получения чашки кофе), то разработка технологического процесса окончится скорее всего неудачей. Строго говоря, для стальной заготовки (вход) на выходе необходимо указывать и деталь, и стружку (а не только деталь). В принципе при разработке технологического процесса следует рассматривать балансы материальных, энергетических и информационных потоков.
3. Технологический процесс должен охватывать только операции с операндами. Однако с практической точки зрения целесообразно расписывать по операторам также все вспомогательные операции, особенно в тех случаях, когда они связаны с преобразованиями непосредственно или вызываются этими преобразованиями. При этом возникает более полное представление о связи таких операций с преобразованиями, а технологический процесс получает более солидное обоснование.
4. Модель опирается на следующее важное соображение: каждый моделируемый технологический подпроцесс ТеР или операция О наряду с соответствующим частичным преобразованием включает также сопутствующие преобразованию побочные процессы, процессы материально-технического снабжения, управления, регулирования и т. п.
5. Путем детализации модели технологического процесса, как это показано на рис. 3.4, можно уточнить данные по материальным, энергетическим и информационным входам или потокам, необходимым для реализации преобразований, а также расписать операции по операторам.

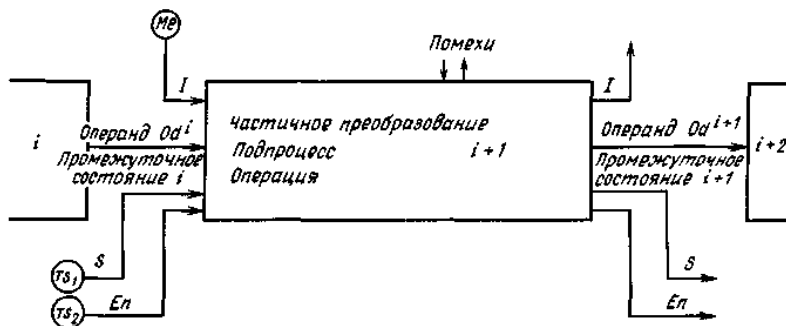


Рис. 3.4. Часть модели технологического процесса

6. Каждый процесс или подпроцесс, каждая операция проходят стадии подготовки, проведения и окончания. Каждая из этих стадий связана с определенным комплексом вопросов.
7. Конкретный технологический процесс (подпроцесс, операция) происходит в определенном месте и в определенное время. Эти условия, которые должны быть заданы для любой модели технологического процесса, обычно включают данные физического (температура, давление, влажность, содержание пыли и т. д.), социального и финансового характера, а также все временные данные.
8. Преобразования в технологическом процессе могут быть описаны в зависимости от цели и стадии его разработки либо укрупненно, либо более подробно. Рекомендуется по возможности все операции прорабатывать с одинаковой степенью детализации.
9. Необходимо тщательно проверять реализуемость каждой операции. В сомнительных случаях следует проводить дополнительные исследования.
10. Разработанный технологический процесс и его запасные варианты можно дополнительно проверить, ставя следующие вопросы:
 - Можно ли пропустить какой-либо подпроцесс или операцию?
 - Нужно ли добавить какую-либо операцию?
 - Не лучше ли заменить одну операцию другой (другими)?
 - Может ли быть изменена последовательность операций?
 - Не целесообразно ли разделить или объединить операции?
 - Возможно ли повторить операцию?
11. Описание подпроцессов и операций осуществляется путем задания объекта преобразования и глагола, например преобразовать энергию, нагреть деталь, послать людей. В зависимости от уровня детализации глаголы описания одной и той же операции могут различаться. Рекомендуется выбирать довольно общие формулировки, соответствующие данному уровню детализации, так как вид описания влияет на выбор средства (например, отпилить шайбу, сделать шайбу).
12. Операции технологического процесса осуществляются во временной последовательности. Если операция выполнена и достигнуто требуемое промежуточное состояние операнда, то после этого должна выполняться следующая за ней операция. При этом условии поток операндов технологического процесса может быть четко зафиксирован на временной оси.

К толкованию модели

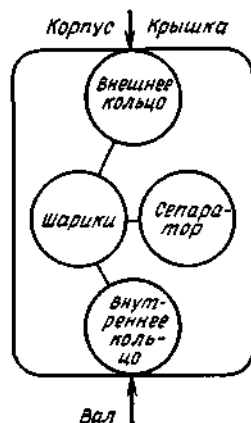
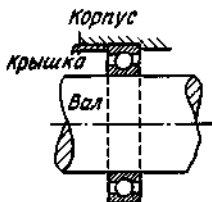
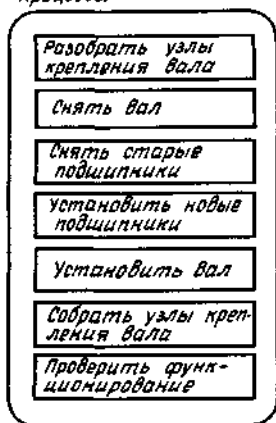
Обсуждаемая модель технологического процесса отражает, конечно, только теоретические возможности. Такие модели могут быть построены различными способами; в заданных рамках ее элементы можно выбирать любым образом. Так, например, в модель можно включить все существенные операции по энергоснабжению и управлению вместо энергетических и информационных потоков. Ограничения по выбору процессов преобразования и отдельных операций, накладываемые на модель, полезны в том смысле, что они позволяют отделить важное от второстепенного и концентрируют внимание на сущности процесса.

Применение модели

Модель технологического процесса могла бы применяться в технике очень широко. Однако ее применение целесообразно только в тех случаях, когда в преобразованиях могут участвовать и участвуют люди, а используемая техническая система имеет характер «машины», т. е. позволяет достичь требуемого результата без участия других технических систем. Такой системой может быть, например, лопата, авторучка или комбайн, но не задняя колесная ось, коробка передач или подшипник. Процессы в технических системах очень специфичны по своему назначению и выполняются обычно одним оператором. Поэтому единственным фактором, определяющим разнообразие, является способ действия. На рис. 3.5 показаны примеры графического представления технологических процессов функционирования и обслуживания технических систем.

а

Процессы



б

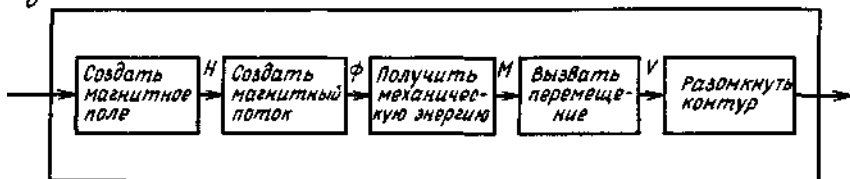


Рис.3.5. Технологические процессы в технических системах

а — замена подшипника, б — работа реле

3.4. Операнды технологического процесса

Операнд технологического процесса является также операндом системы преобразований. Мы будем различать четыре класса операндов:

1. *Живые существа.* Независимо от того, человек это или животное, в технологическом процессе изменяется либо состояние (болен—здоров), либо местоположение операнда; тем не менее было бы неправильным включать живые существа в класс материи ввиду специфичности процессов и ограничений для этого класса операндов.
2. *Материя.* В технологическом процессе изменяются основные свойства (например, химический процесс) или форма, размеры, местоположение и т. д.

3. *Энергия*. В технологическом процессе различные виды энергии (энергоносители) преобразуются в другие виды, а также изменяются их параметры ($p_1 \rightarrow p_2, t_1 \rightarrow t_2$).

4. *Информация*. В технологическом процессе изменяются форма, количество, качество, а также местоположение информации.

Состояние операнда уже определялось как «вектор» его свойств. В математической форме его можно записать следующим образом:

$$Od^1 = \begin{pmatrix} E_1^1 \\ E_2^1 \\ \vdots \\ E_n^1 \end{pmatrix}; \quad Od^2 = \begin{pmatrix} E_1^2 \\ E_2^2 \\ \vdots \\ E_n^2 \end{pmatrix},$$

где подстрочные индексы при E обозначают свойства операнда, а надстрочные — меру этих свойств.

3.5. Структура технологического процесса

Преобразования операндов часто бывают очень сложными. Операнд проходит последовательно многочисленные точно предписанные промежуточные состояния, а его свойства при этом шаг за шагом изменяются в нужном направлении. Изменение свойств может происходить либо непрерывно (например, нагревание), либо скачкообразно (например, разрушение). Объекты переходят в промежуточные состояния посредством подпроцессов TeP и операций O в рамках совокупного технологического процесса.

Операцией мы называем элементарный процесс, соответствующий одному рабочему действию. Операция технологического процесса — это неразрывная во времени часть процесса преобразований, которая выполняется оператором или группой операторов O как независимое задание на одном рабочем месте и касается только одного операнда. Обычно с помощью операции достигается желаемое изменение одного свойства. Так, например, токарной обработкой можно получить тело вращения, фрезерованием — паз, закаливанием — твердую поверхность.

На рис. 3.6 приведен пример технологического процесса из трех операций с изменением двух свойств E_1 и E_2 . Свойство изменяется шагами по ΔE_1 в подоперациях TeO_1 , при этом операнд подвергается воздействию ΔW_1 .

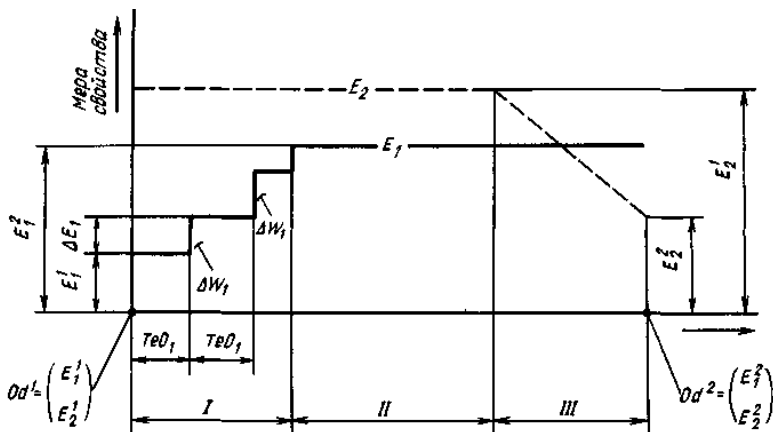


Рис. 3.6. Технологический процесс из трех операций.
 I—операция 1 (обточка), II—операция 2 (транспортировка),
 III—операция 3 (нагрев).

Преобразование $E_1^1 \rightarrow E_1^2$ протекает непрерывно в третьей операции без изменения свойства E_1 . Во второй операции (транспортировка) свойства E_1 и E_2 не меняются.

Согласованная совокупность операций представляет собой технологический процесс. Технологический процесс зависит от требований к операнду, располагаемой информации, квалификации операторов и особенно от технических средств, которыми реализуется процесс. Технология может совершенствоваться во времени в соответствии с прогрессом знаний. Структура технологического процесса TP зависит в первую очередь от технологии и в общем виде изображена на рис.3.7.

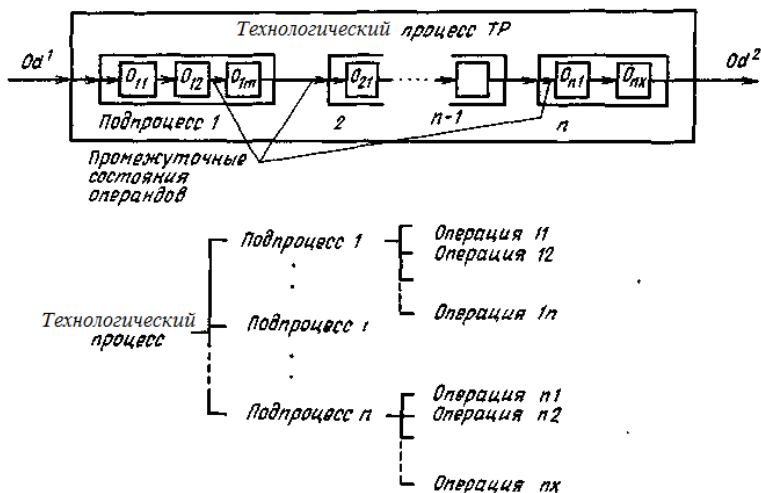


Рис.3.7. Структура технологического процесса.

При описанном расчленении технологического процесса на составные части может возникнуть следующий вопрос: существует ли такой способ расчленения, при котором структура процесса может быть образована из меньшего множества базовых операций, которые могут повторяться? Ответ на него известен, например, в технологии материалов, где такими базовыми операциями являются, например, размельчение, сепарация, агломерация, транспортировка, окисление, восстановление или полимеризация. К другой группе базовых операций относятся изменение физического состояния, формы, местоположения и хранение.

Следующий важный вопрос касается вариантов структуры.

Конкретизируя соответствующее положение изложенное ранее, сформулируем его так: преобразования системы типа «процесс» могут быть реализованы различными структурами (т. е. технологиями).

3.6. Типы операций

Для достижения желаемого изменения свойств нужно надлежащим образом воздействовать на операнд. Например, операнд нужно привезти, нагреть, деформировать и т. д. Такого рода преобразующее воздействие на операнд мы будем называть *рабочей операцией*. При рабочих операциях используются те или иные физические законы и явления. Так для нагревания, т. е. повышения температуры, нужно

знать, как различные виды энергии преобразовать в тепло и как обеспечить теплопередачу.

Совокупное воздействие (W) для осуществления преобразования ($Od^1 \rightarrow Od^2$) состоит из частичных воздействий (TeW), изменяющих определенное свойство, например $E^1_1 \rightarrow E^2_1$. Можно написать $E^1_1 \rightarrow E^2_1 = f(TeW_i)$;

при этом совокупное воздействие является суммой частичных воздействий

$$W = \begin{pmatrix} TeW_1 \\ TeW_2 \\ \vdots \\ TeW_i \\ \vdots \\ TeW_n \end{pmatrix} \text{ или } Od^2 - Od^1 = f(W).$$

Для реализации процесса преобразований (рис. 3.8) одних рабочих операций недостаточно.

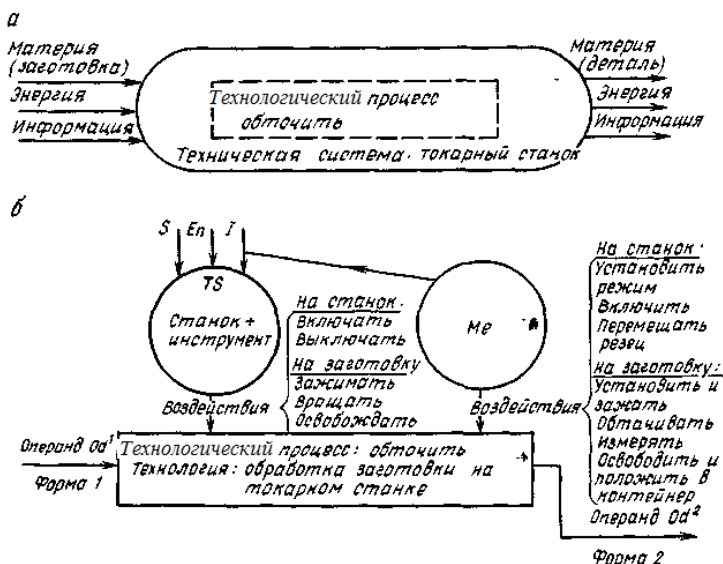


Рис.3.8. Примеры построения модели процесса преобразований (пример — деталь).

а—заготовка является входом технической системы, **б**—заготовка обрабатывается в технологическом процессе за счет воздействий токарного станка и токаря

Обычно для каждого процесса требуются и другие операции, хотя зачастую их не отделяют, по-скольку выполнение таких вспомогательных операций при обслуживании машины человеком необходимо. К операциям такого типа относятся:

- *операции обслуживания*, например смазка, удаление стружки, охлаждение;
- *подготовительные операции*, например закрепление детали, подведение суппорта;
- *операции по управлению и регулированию*, например измерение, наладка машины, изменение рабочего режима;
- *операции согласования*, например сборка, увязка частей проекта.

3.7. Побочные входы и выходы

В процесс входит и из него выходит только операнд *Od*. Для осуществления процесса необходимы также различные другие входные величины, такие, как энергия, смазочные материалы, катализаторы. Неизбежны в технологическом процессе также различные отходы (дым, стружка) и помехи (шум, вибрация). Такие входные и выходные величины будем называть *побочными входами* *NIn* и *побочными выходами* *NOu*.

Учитывать побочные входы и выходы необходимо, так как человечеству уже сегодня угрожают вредные выходы технологических процессов (загрязнение окружающей среды). До настоящего времени вредным побочным выходам уделялось слишком мало внимания. При осуществлении технологического процесса следует учитывать нежелательные, но неизбежные входные помехи. При проектировании технологического процесса нужно проанализировать все возможные помехи, исходящие от окружения, а при функционировании технической системы следует контролировать уровни отдельных входов, если они близки к предельно допустимым. Таковыми являются, например, предельные значения температуры и влажности для работы высокочувствительного оборудования.

3.8. Операторы технологического процесса

Важным вопросом в разработке технологического процесса *TP* является вопрос о том, каким образом осуществить нужное воздействие. Рассмотрим, например, операцию закаливания. Рабочий может щипцами окунуть раскаленную деталь в холодную ванну и затем вынуть ее. По другому варианту технологического процесса

уложенные на ленточный конвейер детали нагреваются до нужной температуры в печи непрерывного действия, затем погружаются в ванну охлаждения и со второго ленточного конвейера поступают в специальный контейнер. В первом случае оператор *Ot* без участия технической системы *TS* (если не учитывать печь и ванну охлаждения) выполняет все вспомогательные и управляющие операции, включая транспортировку из одной технической системы в другую. В случае автоматической линии (второй вариант) работа людей ограничивается наладкой и приведением в действие системы, манипуляциями с материалами, контролем производства и техническим обслуживанием. Таким образом, вся упорядоченная совокупность необходимых действий в технологическом процессе осуществляется системой человек-машина (точнее, человек — техническая система) с распределением функций между обоими компонентами системы. Полностью ради в такую систему следует включить и животных, поскольку иногда они используются в технологических процессах (например, лошадь с плугом или слон при переноске бревен). Для совокупного результата технологического прогресса (рис. 3.9.) имеют значение также *обобщенные операторы*, которые оказывают влияние прежде всего на эффективность протекания процесса.

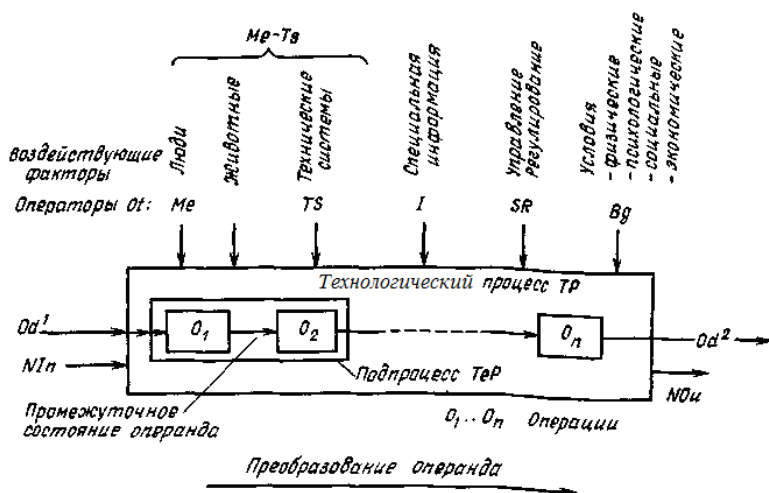


Рис.3.9. Воздействия на операнд в процессе его преобразования.

В некоторых случаях они могут помешать осуществлению преобразований. Поэтому мы должны учитывать возможность влияния этих

операторов на качество продукции. Такими операторами, кроме прочих, являются:

- специальная информация I , определяющая технический уровень данного производства;
- управление процессом SR , в первую очередь оперативное управление, координация производственных действий и система материально-технического снабжения;
- условия окружения Bg : физические (температура, влажность, объем, запыленность); психологические (взаимоотношения людей); социальные (состояние и интересы общества в данный период времени); финансовые (средства для осуществления процесса). Результат технологического процесса, т. е. выходное состояние операнда Od^2 , определяется системой человек — техническая система ($Me—TS$), а также упомянутыми выше обобщенными операторами, а именно техническим уровнем производства, специальной информацией, управлением процессом и условиями окружения. Указанные факторы можно включить в материальные, энергетические и информационные потоки.

3.9. Характеристики и оценка технологического процесса

Обычно технологический процесс характеризуется выходом, т. е. качеством операнда на выходе. Если все требуемые свойства операнда достигнуты, то задача процесса выполнена. С другой стороны, технологический процесс может быть охарактеризован качеством обобщенных операторов процесса, а именно: а) качеством технологии; б) квалификацией специалистов и совершенством технических средств; в) качеством информации, управления и внешних условий. Если технологический процесс не выполняет свою задачу преобразования операнда, то анализ указанных обобщенных операторов позволяет выявить причины этого.

Совокупным показателем, характеризующим технологический процесс и обеспечивающим возможность сравнения с другими процессами, является его эффективность E_f , определяемая как

Полезность процесса/Затраты на преобразование.

Полезность процесса создается совершаемыми воздействиями и определяется изменением свойств операнда. Кроме того, следует учитывать побочные выходы, которые могут увеличить полезность. Затраты охватывают все расходы, связанные с системой преобразований, независимо от того, являются ли они операндами, побочными входами или какими-либо операторами, т. е. это могут

быть расходы по операциям, энергии, основным и вспомогательным материалам, заработной плате, социальным затратам, обслуживанию, накладные расходы и т. п.

Для однозначной оценки или характеристики технологического процесса следует определить показатели ценности его свойств. Речь идет о совокупности показателей, приведенных в табл. 3.3.

Таблица 3.3. Показатели технологических процессов

Показатели	Технические	Экономические	Плановые
Операнд Вход Выход	Материал, размеры, форма состояние поверхности давление, температура и другие параметры	Цена Затраты	Количество Срок Поставщик
Технология	Спецификация операций Последовательность операций	Затраты	Время работы Оператор Рабочее место
Оператор рабочих	Специальные знания Опыт Личные свойства	Заработная плата "	Часы работы Количество рабочих

продолжение

Показатели	Технические	Экономические	Плановые
Техническая система	Функциональные свойства Эксплуатационные свойства Эстетические свойства Эргономические свойства Манипуляционные свойства и др	Цена Эксплуатационные расходы	Срок поставки Количество Поставщик
Специальная информация	Перечень типов информации Источники информации	Затраты	Срок Работник Рабочее место
Условия окружения	Физические расположение потребность в пространстве температура влажность освещение шум	Затраты	Срок Оператор Рабочее место
	Психологические рабочая обстановка	Производительность труда	
	Социальные общественные условия	Экономическая ситуация	

Они разделены на три категории: технические, экономические и плановые показатели. Формальная процедура сравнения и оценки технологического процесса аналогична оценке системы типа «объект» (см. ниже). В качестве примера в табл. 3.4 приведена укрупненная оценка типа «плюс — минус» для процесса передачи энергии. Альтернативой этому была бы оценка по баллам, возможно, с собственными критериями по каждому показателю.

Таблица 3.4. Оценка технологических процессов (на примере передачи энергии)

Передача энергии	Механической	Гидравлической	Электрической
Большое расстояние	—	+	++
Удобство управления процессом	—	+	+
Безопасность	Одинаково		
Экономичность	+	—	—
Доступность компонентов	—	+	++
Приспособляемость	—	+	++
Стоимость	—	—	+

Примечание — невозможно, + возможно, ++ предпочтительно.

Здесь необходимо обратить внимание на социальные и экономические последствия решений, связанных с технологическим процессом. Решения такого рода оказывают влияние на стоимость товаров, занятость людей, экономическую ситуацию и ряд других важных социально-экономических факторов.

3.10. Классификация технологических процессов

Классификацию технологических процессов можно проводить с различных точек зрения. Пример такой классификации приведен в табл.3.5.

Таблица 3.5. Классификация технологических процессов

Показатели	Классы процессов	Показатели	Классы процессов
Операнд	Процессы переработки материи Процессы переработки энергии Процессы переработки информации Процессы, связанные с биологическими объектами	Способ рабочего действия Способ управления и регулирования Сложность процессов	Ручной труд Использование силы животных Механизированный процесс С участием человека Автоматизированный процесс
Явления, на которых основываются рабочие действия	Физические Механические Электрические Тепловые Химические Биологические Комбинированные	Соотношение между входом и выходом	Операции Подпроцессы Сложные процессы Объединения: количество входов больше количества выходов Разделения: количество входов меньше количества выходов
Рабочее действие	Транспортировка Сортировка Измельчение Обработка Сборка	Характер протекания преобразования	Непрерывный процесс Дискретный (прерывающийся) процесс

Объединив типы операндов со способами изменения, преобразования структуры, формы, пространственных (транспортировка) и временных (хранение) координат, получим основные виды технологических процессов, представленные в табл.3.6.

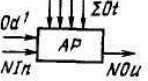
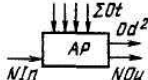
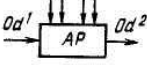
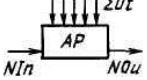
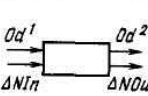
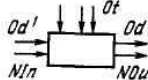
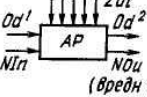
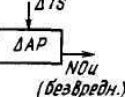
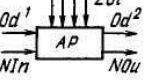
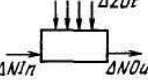
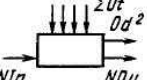
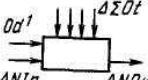
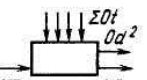
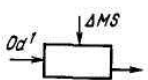
Таблица 3.6. Основные виды технологических процессов

Типы операций		Преобразование			
		структуры	формы	пространственных координат	временных координат
Материя <i>S</i>	Обобщенный процесс	Перерабатывать	Обрабатывать	Транспортировать	Хранить
	Пример	Железная руда → сталь	Заготовка → деталь	Цемент на складе → на стройплощадке	Продукты в холодильнике
Энергия <i>En</i>	Обобщенный процесс	Превращать	Трансформировать	Транспортировать	Накапливать
	Пример	Гидравлическая → электрическая	50000 → 220 В	Электростанция → потребитель	Электрическая энергия в аккумуляторе
Информация <i>I</i>	Обобщенный процесс	Преобразовывать	Изменять форму	Передавать	Накапливать
	Пример	Графическая → цифровая	Текст немецкий → русский	Радиопередача → слушатель	Запись на магнитной ленте
Человек <i>Me</i>	Обобщенный процесс	Изменять существо	Изменять форму	Транспортировать	Содержать
	Пример	Рука → протез	Болен → здоров	Путешествие	Ожидание врача

3.11. Производственная деятельность, связанная с технологическим процессом

Технологический процесс — это сложная система. На практике часто возникают ситуации, когда задан ряд характеристик технологического процесса, а недостающие нужно определить. В соответствии с тем, какие величины являются искомыми, возникают различные задачи (табл.3.7).

Таблица 3.7. Типичные задачи в области технологических процессов

Ситуация	Дано	Найти	Деятельность
1 Для удовлетворения потребностей нужен новый продукт			Процесс проектирования при заданном выходе
2 Для существующего материала (энергии) ищется применение			Процесс проектирования при заданном входе
3 Для заданных операндов, рабочего процесса и всех операторов (существующее производство) ищется TS			Создание (приобретение) технической системы
4 Для существующего производства с комплектом оборудования ищется работа (операнд)			Работа по заказу
5 Для разработанной TS изыскиваются возможности применения			Новая область применения имеющейся технической системы
6 Побочный выход существующего применения вреден			Разработка системы ликвидации вредного побочного выхода (NO u)
7 Должна быть повышена экономическая эффективность существующего производства			Рационализация технического процесса
8 Существующее производство должно быть перестроено на новую программу			Подготовка серийного производства
9 На существующем производстве должен быть изготовлен новый объект (в единственном экземпляре)			Подготовка единичного производства

Из этой таблицы наглядно видно, что путем систематического упорядочения можно довольно точно определить виды необходимой производственной деятельности.

3.12. Представление технологических процессов

Технологические процессы могут быть представлены различными способами в зависимости от вида процесса, цели представления, действующих инструкций или традиции. Могут быть использованы следующие способы представления:

- а) *Представление в форме блок-схемы.* В этом случае процесс изображается в виде прямоугольника с текстом или рисунком, характеризующим данный процесс.
- б) *Представление в форме графа.* Ребра графа обозначают процессы, а узлы — состояния операндов. При этом входы и выходы процессов можно изображать точками и стрелками (рис.3.10,а).
- в) *Временная диаграмма.* Такая диаграмма позволяет наглядно представить последовательность и привязку операций во времени, что может быть полезным, например, для целей управления (рис.3.10,б).

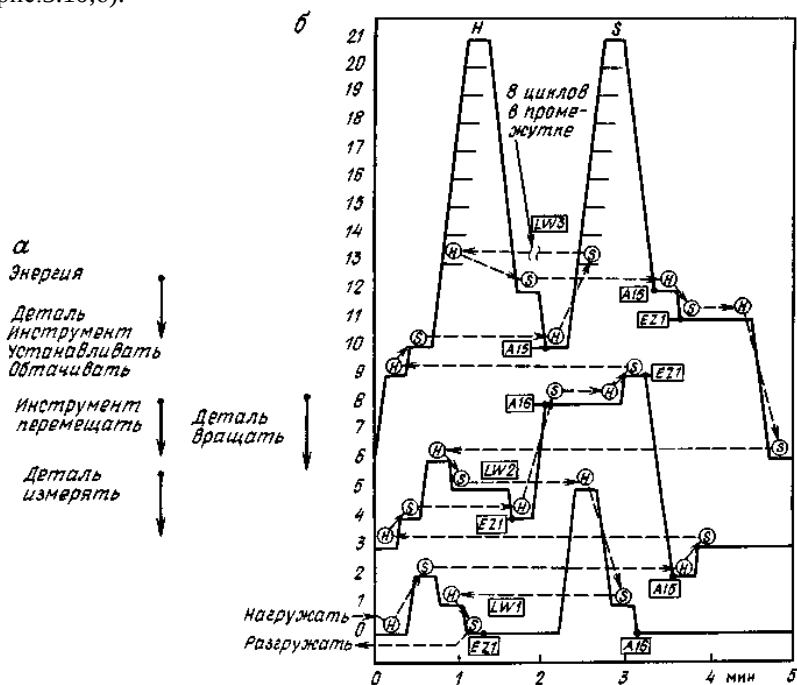


Рис.3.10. Примеры изображения технологических процессов в форме графа (а) и в виде временной диаграммы (б).

г) *Математическое описание*. Уравнения в большинстве случаев наиболее точно описывают протекание процессов и их зависимости.
д) *Словесное описание*. Словесное описание проще других, однако оно либо не обеспечивает однозначности, либо оказывается слишком длинным.

е) *Специальное представление*. Примером может служить описание на языке Ляпунова (Операторный язык программирования (см. Ляпунов А. А. О логических схемах программ — *Проблемы кибернетики*, 1958, вып. 1). При символическом описании такого рода технологические операции обозначаются прописными латинскими буквами, а условия — строчными. Таким образом, запись $q(\varnothing 90H7)$ означает, что задается условие получить диаметр 90 с допуском H7. Согласно «языку Ляпунова», каждой логической операции соответствуют вертикальные стрелки, причем $\uparrow$ обозначает начало и $\downarrow$ — конец логической операции. Так, выражение $\uparrow \downarrow Aq \uparrow \uparrow B$ означает следующее: провести операцию A (например, обточка), затем проверить, выполняется ли условие q (например, $\varnothing 90H7$). Если оно выполняется, то осуществляется операция B (например, фрезерование паза). Если оно не выполняется, то следует повторить операцию A , что обозначается стрелкой $\uparrow$ после q .

Наши рассуждения до сих пор были сосредоточены на общей модели, применимой для всех видов технологических процессов.

Обсуждение технологических процессов, проведенное в этом разделе, можно обобщить в следующих выводах.

1. Технологические процессы — это особый вид процессов преобразований. В них технические системы выступают как используемое человеком орудие труда, поэтому для технологических процессов справедливы все выводы по процессам преобразований.
2. Технологический процесс состоит из множества подпроцессов и операций и множества промежуточных состояний операндов.
3. Для стадий подготовки, проведения и окончания процесса можно построить подструктуры.
4. Процессы, в которых не участвуют ни преобразуемый операнд, ни выполняющая его роль техническая система, допускаются только как исключение (из прагматических соображений).
5. Каждая рабочая операция технологического процесса вызывает ряд вспомогательных операций (подготовки, обслуживания, управления, регулирования и согласования) и побочных про-

цессов. При изображении технологического процесса все они условно включаются в процесс преобразования.

6. При разработке и осуществлении технологического процесса всегда нужно учитывать окружение (среду) и временные рамки.

7. Технологический процесс как система всегда имеет побочные входы и выходы (включая помехи); их необходимо учитывать и анализировать.

8. Хотя в технологическом процессе, рассматриваемом как система, наиболее важными операторами являются технические системы, остальные операторы также должны быть приняты во внимание.

4. Облачные вычислительные процессы и их модели

4.1. Логическая организация и порядок выполнения программ

Задача, сформулированная для решения облачными вычислениями, представляется множеством исходных данных, множеством искомых результатов и программой, описывающей алгоритм решения задачи. *Алгоритм* — это правило, определяющее последовательность действий над исходными данными, приводящую к получению искомых результатов. Алгоритмы могут представляться в словесной форме с использованием необходимой математической символики, средствами некоторой алгоритмической системы — в виде операторной схемы алгоритма (логической схемы или граф-схемы), на алгоритмическом языке и т. д.

Форма представления алгоритма, ориентированная на машинную реализацию, называется *программой*. Программа записывается на некотором языке программирования и строится с учетом присущих машине ограничений.

Информация, относящаяся к задаче, разделяется на программу и данные. *Данные* — совокупность символов, представляющих исходные данные, промежуточные величины и конечные результаты. На организацию машинных программ, представленных в терминах машинных команд, наиболее существенное влияние оказывает специфика организации машинной памяти. Память типичной ЭВМ разделяется на две функционально различные части: оперативную и внешнюю.

Оперативная память предназначена для хранения информации, участвующей в обработке, т. е. для хранения совокупности команд, порождающей необходимую последовательность операций, и совокупности данных, участвующих в качестве операндов в этих операциях. Емкость оперативной памяти незначительна и может оказаться недостаточной для размещения всей информации, относящейся к решаемой задаче.

Внешняя память ЭВМ имеет значительную емкость, в 10^5 — 10^{10} раз превосходящую емкость оперативной памяти, но хранимая в ней информация может подвергаться обработке только после передачи этой информации в оперативную память.

Оперативная и внешняя память имеет существенно различные скоростные характеристики. Затраты времени на чтение или запись информации в оперативную память пропорциональны количеству читаемой или записываемой информации. Затраты времени на передачу информации из внешней памяти пропорциональны в основном количеству обращений к памяти и мало зависят от количества передаваемой информации. Поэтому для уменьшения затрат времени на передачу информации из внешней памяти за каждое обращение стремятся передавать достаточно большое количество информации.

Таким образом, по отношению к процессору, выполняющему обработку информации, память ЭВМ *двухуровневая* (рис. 4.1), где *Пр* — процессор, *ОП* и *ВП* — соответственно оперативная и внешняя память.

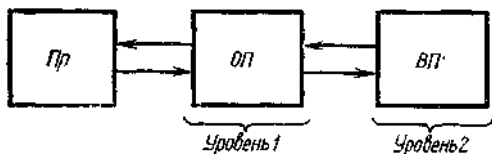


Рис. 4.1. Двухуровневая структура памяти ЭВМ

Наличие двух функционально различных уровней памяти приводит к необходимости при составлении машинных программ распределять информацию по уровням памяти и включать в программы команды, обеспечивающие передачу информации между ними.

Оперативная и внешняя память может быть объединена в память одного уровня на основе принципа виртуальной памяти со страничной организацией. С этой целью в аппаратуру и программное обеспечение ЭВМ включаются средства, обеспечивающие автоматическую, заранее не программируемую передачу информации между уровнями памяти.

Применительно к двухуровневой памяти задачи программируются в следующем виде. Информация, которая должна храниться во внешней памяти, разделяется на *файлы* (наборы данных)— упорядоченные совокупности данных, обрабатываемых одним общим для этих данных способом. Данные, составляющие файл, разделяются на *записи* — совокупности единиц информации (байтов), обрабатываемых совместно как одно целое. Информация, составляющая запись, обычно характеризует один объект — документ, строку матрицы, библиографическую справку о книге и т. п. Запись может быть загружена во внешнюю память и прочитана только как целое — за одно обращение к внешней памяти. Файлы и записи — логические единицы информации, выделяемые с учетом смысла информации и логики ее обработки.

С использованием понятий файл и запись логическая организация программы может быть представлена так, как показано на рис. 4.2, где ОП— оперативная память; Д — данные.

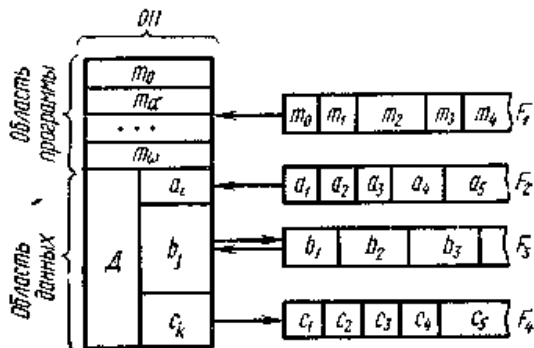


Рис. 4.2. Логическая организация программы

Если программа не может быть целиком размещена в оперативной памяти из-за ограниченной емкости последней, то она разделяется на части $m_0, \dots, m_n$, называемые *загрузочными модулями*. Загрузочные модули хранятся во внешней памяти как записи $m_0, m_1, \dots$, составляющие файл F_1 , называемый *главным файлом*. Данные, связанные с задачей, разделяются на файлы $F_2, F_3, \dots$, а файлы и свою очередь разделяются на записи $a_1, a_2, \dots; b_1, b_2, \dots$.

В зависимости от способа использования данных и процессе решения задачи различают входные, выходные и входные — выходные файлы. *Входной файл* содержит исходные данные, которые только читаются из

файла в процессе решения задачи. Входные файлы существуют до начала решения задачи. *Выходной файл* содержит результаты решения задачи, записываемые в файл в порядке их получения. *Входной* — *выходной файл* используется как для чтения записей, так и для загрузки их в процессе решения задачи. Такая ситуация типична при корректировке записей и файла в целом (файл может пополняться новыми записями и из него могут исключаться отдельные записи). Входные— выходные файлы подразделяются на основные и рабочие. *Основной файл* существует до начала и (или) по окончании решения задачи. *Рабочий файл* создается только на период решения задачи и содержит в себе промежуточные данные, необходимые лишь для одного прогона программы.

Для решения задачи в оперативной памяти выделяется раздел, в котором условно можно указать две части: область программы и область данных (рис. 4.2). *Область программы* имеет емкость, достаточную для размещения подмножества загрузочных модулей, которые должны одновременно присутствовать в оперативной памяти в процессе решения задачи. Задача начинается выполняться с момента загрузки в оперативную память корневого (ведущего) модуля m_0 и передачи ему управления. В некоторый момент времени модуль m_0 может вызвать в оперативную память модуль m_α , который в свою очередь может обратиться к модулю m_β и т. д. В результате это в области программы могут одновременно размещаться несколько модулей $m_n, m_\alpha, \dots, m_\omega$. После завершения работы модуль m_ω передает управление вызвавшему его модулю и область памяти, занимаемая модулем m_ω , считается свободной и в нее может быть вызван очередной модуль. Таким образом, в каждый момент времени в оперативной памяти присутствует лишь часть модулей, входящих в состав программы. В *области данных* выделяется подобласть D , в которой размещаются данные, в основном постоянно используемые в процессе решения задачи. Доступ к данным, размещенным в файлах F_2, F_3, F_4 , организуется путем выделения областей оперативной памяти a_i, b_j, c_k , называемых *буферами ввода — вывода*. Каждый из буферов обеспечивает размещение по крайней мере одной записи из соответствующего файла. Буфер a_i , обслуживающий входной файл F_2 , используется для хранения записи a_i из файла F_2 . Запись вызывается в буфер командами ввода — вывода, после чего данные, включенные в эту запись, становятся доступными для обработки. Аналогично, результаты, формируемые программой, заносятся в область буфера c_k и по мере формирования записей переносятся в выходной файл F_4 . Перед выполнением программы корневой модуль m_0 загружается в начало раздела, выделенного задаче. Программа иницируется путем

передачи управления первой команде корневого модуля. Начиная с этого момента выполнение программы протекает в порядке, иллюстрируемом рис. 4.3.

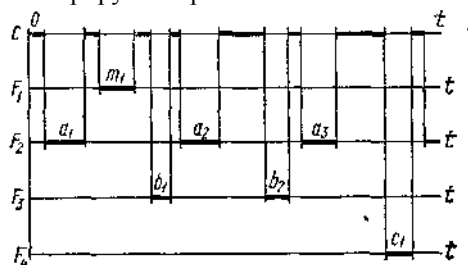


Рис. 4.3. Временная диаграмма процесса выполнения программы

Отрезки на осях времени t обозначают периоды активного состояния процессов обработки C и обращения к файлам $F_1, \dots, F_4$. Выполнение программы состоит из последовательности периодов обработки информации и обращения к файлам, которые следуют в порядке, указанном программой. Длительность периодов определяется трудоемкостью работ — количеством операций, которые должен выполнить процессор, количеством информации, передаваемой при обращении к файлу, и быстродействием устройств, используемых для выполнения соответствующих работ. В распоряжение выполняемой программы должны быть предоставлены необходимые ресурсы — разделы оперативной и внешней памяти и определенные устройства. Выполнение программы, обеспеченной необходимыми ресурсами, можно рассматривать как последовательность этапов обслуживания запросов программы к устройствам ЭВМ. Разделы оперативной и внешней памяти, закрепленные за программой, остаются занятыми в течение всего периода выполнения программы. Устройства — процессор и внешние запоминающие устройства — используются программой периодически и в периоды, свободные от обслуживания данной программы, могут быть предоставлены в распоряжение других программ.

4.2. Сложность задач и потребность в ресурсах памяти

Затраты памяти на хранение программ и данных в ОВС зависят от класса задач, решение которых является функцией ОВС. Количество информации, необходимое для решения задачи, характеризует

сложность задачи. Сложность задачи складывается из сложности алгоритма и количества данных.

СЛОЖНОСТЬ алгоритма — минимальное количество информации, необходимое для его описания. Исчисляется сложность алгоритма в единицах информации — битах или байтах. Однако оценка количества информации, минимально необходимого для описания алгоритма, представляет собой чрезвычайно трудную задачу. В практических приложениях сложность алгоритма целесообразно определять *длиной записи алгоритма* в терминах определенной алгоритмической системы. Так, сложность алгоритма можно характеризовать числом операторов программы или числом команд программы в машинном коде. Программы в свою очередь могут быть охарактеризованы в единицах информации, например числом байтов (символов), составляющих запись программы. В этом случае сложность алгоритма определяется косвенно, поскольку количество информации в записи алгоритма зависит не только от сложности последнего, но и от информационной эффективности алгоритмической системы, в терминах которой описывается алгоритм. Чтобы отметить этот факт, сложность, оцениваемую длиной программы, будем рассматривать как *сложность программы*, но не как сложность алгоритма.

Количество данных, относящихся к задаче, характеризуется числом байтов, посредством которых представляются данные. Таким образом, располагая сведениями о сложности программы и количестве данных, можно определить потребность задачи в (ресурсах памяти. Применяя процедуру разделения информации на части, хранимые в оперативной и внешней памяти, можно определить потребность программы в ресурсах оперативной и внешней памяти. Точная оценка сложности программы и количества данных может быть получена только апостериорно — после программирования задачи и, возможно, после получения машинной программы. Поскольку процесс программирования достаточно трудоемок, то во многих случаях приходится ориентироваться на приближенные оценки сложности программ, которые находятся путем неглубокого анализа задачи. (Задача расчленяется на небольшое число в какой-то степени самостоятельных частей и оценивается сложность каждой части на основании личного или коллективного опыта.

Класс задач, решаемых на ОВС общего назначения, не может быть определен перечнем задач и характеризуется свойствами этих задач — в первую очередь присущей им сложностью и трудоемкостью.

Сведения о сложности прикладных задач собираются путем статистической обработки данных о задачах, решаемых на ОВС. Экспериментальные данные корректируются с учетом перспективы

роста сложности задач и используются в качестве исходных данных при проектировании новых систем.

4.3. Трудоемкость алгоритмов и марковские модели облачных вычислительных процессов

Трудоемкость алгоритма — количество вычислительной работы, требуемой для реализации алгоритма. Если сложность алгоритма характеризует потребность алгоритма в памяти, то трудоемкость — его потребность во времени, связанном с периодом работы совокупности устройств, средствами которых реализуется алгоритм. Трудоемкость алгоритма иначе называют сложностью вычислений. Оценивается трудоемкость алгоритма количеством операций, выполняемых с целью обработки, ввода и вывода информации в процессе решения задачи. Каждой реализации алгоритма присущ элемент случайности, связанный с тем, что исходные данные представляют собой в общем случае случайную выборку из множества исходных данных, к которым применим алгоритм. Поэтому полная характеристика трудоемкости предполагает описание количества операций, выполняемых за одну реализацию алгоритма, случайными величинами, т. е. предполагает определение законов распределения числа операций в реализации алгоритма. Получение таких сведений об алгоритме — сложный и длительный процесс. В связи с этим трудоемкость алгоритмов обычно характеризуют приближенно, например, только математическими ожиданиями числа выполняемых операций.

В первом приближении трудоемкость алгоритма можно охарактеризовать следующей совокупностью параметров: θ — среднее количество процессорных операций, выполняемых за одну реализацию алгоритма (при одном прогоне программы); $N_1, \dots, N_n$ — среднее количество обращений к файлам $F_1, \dots, F_n$ соответственно за одну реализацию алгоритма; $\theta_1, \dots, \theta_n$ — среднее количество информации (байтов), передаваемое за одно обращение к файлам $F_1, \dots, F_n$ соответственно. Значение θ характеризует трудоемкость обработки информации (счета), а значения $N_1, \dots, N_n, \theta_1, \dots, \theta_n$ — трудоемкость процесса ввода — вывода информации.

Модели облачного вычислительных процессов. При решении задач анализа и синтеза ОВС возникает необходимость в описании свойств облачных вычислительных процессов, порождаемых алгоритмами решения задач. С наибольшей полнотой и точностью свойства облачных вычислительных процессов представляются самими Алгоритмами, определяющими облачные вычислительные процессы

до мельчайших нюансов. Однако при решении задач теории ОВС интерес представляют не все без исключения детали облачных вычислительных процессов, а только те, которые характеризуют порядок использования ресурсов системы в процессе решения задач. Следовательно, облачные вычислительные процессы целесообразно представлять моделями, несущими в себе информацию только о свойствах облачных вычислительных процессов, учет которых необходим и возможен при решении задач анализа и синтеза ОВС. Количество действий, выполняемых в процессе вычислений, определяется трудоемкостью алгоритма, и если при построении модели облачного вычислительного процесса исходить только из сведений о трудоемкости алгоритма, то степень приближения модели к реальным процессам целиком определяется полнотой и достоверностью сведений о трудоемкости алгоритма. С учетом сведений о трудоемкости алгоритма, которыми обычно располагают, и возможных подходов к анализу и синтезу ОВС можно сформулировать следующие требования к моделям облачных вычислительных процессов:

1. Модель должна определять порядок порождения алгоритмом запросов на каждый из видов обслуживания — счет и ввод — вывод информации, хранимой в каждом из файлов.
2. Модель должна определять трудоемкость обслуживания запросов — количество операций, которое должен выполнить процессор при обслуживании запроса на счет, и количество символов информации — выводимой информации.
3. Модель должна отображать облачные вычислительные процессы как реализации случайного процесса, т. е. порождать запросы в случайные моменты времени и характеризовать трудоемкость запросов случайными величинами.
4. Облачные вычислительные процессы, порождаемые моделью, должны соответствовать реальным процессам с точностью до совпадения по крайней мере математических ожиданий их одноименных характеристик.

Первые два требования выделяют круг сведений об облачных вычислительных процессах, наиболее существенно влияющих на порядок функционирования ОВС. На данном уровне рассмотрения вопросов прочие сведения об облачных вычислительных процессах не принимаются во внимание, т. е. алгоритмы и облачные вычислительные процессы считаются различными постольку, поскольку они различаются по количеству и характеру запросов на обслуживание. Необходимость третьего требования продиктована результативностью вероятностного подхода к исследованию многообразных процессов, какими являются облачные

вычислительные процессы. Фактор случайности, вводимый в модель, позволяет порождать бесконечное число реализаций облачного вычислительного процесса, что характерно для процессов выполнения алгоритма. Последнее требование определяет минимальную норму «точности» модели. Конечно, идеальная модель должна представлять реальные процессы с точностью до равенства распределений соответствующих случайных величин, описывающих вычислительные процессы. Однако от построения идеальной модели приходится отказаться по следующим причинам. Во-первых, в подавляющем большинстве случаев сведения об алгоритмах, для выполнения которых создается система, далеко не полные. В лучшем случае известны оценки математических ожиданий характеристик алгоритма и лишь иногда — их дисперсии. Поэтому бесполезно вводить в модель параметры, значения которых не удастся определить. Во-вторых, оправданное стремление получить общее решение задачи, пусть и приближенное, заставляет идеализировать вычислительные процессы — представлять их в виде, обеспечивающем разрешимость задачи аналитическими методами. Конечно, всегда должна устанавливаться степень достоверности результатов, получаемых на основе идеализированных моделей.

С учетом вышеперечисленных требований построим модель облачного вычислительного процесса, порождаемого алгоритмом с заданной трудоемкостью

Будем рассматривать облачный вычислительный процесс как последовательность этапов счета и ввода — вывода информации при обращении к файлам $F_1, \dots, F_N$. Типичная диаграмма такого процесса изображена на рис. 4.3. Состояние облачного вычислительного процесса, соответствующее этапу счета, обозначим символом S_0 , а состояния, соответствующие обращениям к файлам $F_1, \dots, F_N$, — символами $S_1, \dots, S_N$. Окончание облачного вычислительного процесса будем рассматривать как переход процесса в состояние S_{N+1} , поглощающее вычислительный процесс. В этих обозначениях облачный вычислительный процесс — это последовательность состояний $S_{t_0}, S_{t_1}, \dots, S_{t_M}$, изменяющихся в моменты времени

$t_0, t_1, \dots, t_M$, причем $S_{t_i} \in \{S_0, \dots, S_N\}$ и заключительное состояние процесса $S_{t_M} = S_{N+1}$.

Марковская модель облачного вычислительных процессов.

Наиболее простую модель можно получить, если принять допущение об отсутствии последствия в облачном вычислительном процессе, означающем, что следующее состояние облачного вычислительного

процесса зависит только от текущего состояния и не зависит от предыдущих состояний. В таком случае облачный вычислительный процесс становится марковским процессом, определяемым множеством присущих ему состояний $\{S_0, \dots, S_H\}$,

матрицей вероятностей переходов

$$P = [p_{ij}] = \begin{matrix} & \begin{matrix} S_0 & S_1 & \dots & S_{H+1} \end{matrix} \\ \begin{matrix} S_0 \\ S_1 \\ \dots \\ S_{H+1} \end{matrix} & \begin{bmatrix} p_{0,0} & p_{0,1} & \dots & p_{0,H+1} \\ p_{1,0} & p_{1,1} & \dots & p_{1,H+1} \\ \dots & \dots & \dots & \dots \\ p_{H+1,0} & p_{H+1,1} & \dots & p_{H+1,H+1} \end{bmatrix} \end{matrix}$$

и распределением вероятностей $(a_0, \dots, a_{H+1})$ состояний $S_0, \dots, S_H$ в момент времени 0. Элементы p_{ij} матрицы P определяют вероятности перехода процесса из состояний S_i в состояния S_j , т. е. вероятности того, что процесс, находящийся в состоянии S_i в следующий момент времени будет находиться в состоянии S_j . Матрица P — стохастическая матрица, построчные суммы элементов которой

$$\sum_j p_{ij} = 1.$$

Вероятности a_j определяют первое возможное состояние S_{t_0} процесса.

Будем считать, что облачный вычислительный процесс развивается следующим образом. Процесс начинается с состояния S_0 , т. е. программа начинает выполняться с этапа счета. Этап ввода — вывода может быть инициирован только процессором, т. е. может следовать только за этапом счета. Это одновременно означает, что после каждого этапа ввода — вывода следует этап счета. В таком случае вероятности начальники состояний

$$(a_0, a_1, a_2, \dots, a_{H+1}) = (1, 0, 0, \dots, 0) \quad (4.1)$$

и матрица вероятностей переходов

$$P = \begin{matrix} & \begin{matrix} S_0 & S_1 & S_2 & \dots & S_H & S_{H+1} \end{matrix} \\ \begin{matrix} S_0 \\ S_1 \\ S_2 \\ \dots \\ S_H \\ S_{H+1} \end{matrix} & \begin{bmatrix} 0 & p_{0,1} & p_{0,2} & \dots & p_{0,H} & p_{0,H+1} \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} \end{matrix}. \quad (4.2)$$

Из состояния счета S_0 процесс с соответствующей вероятностью может перейти в состояния $S_1, \dots, S_H$, представляющие этапы обращения к файлам $F_1, \dots, F_H$, или в поглощающее состояние S_{H+1} . Из состояний $S_1, \dots, S_H$ процесс с вероятностью 1 возвращается в состояние счета S_0 .

Достигнув поглощающего состояния S_{N+1} процесс с вероятностью 1 навсегда остается в нем. Порядок смены состояний наглядно проявляется на графе марковской цепи (рис. 4.4).

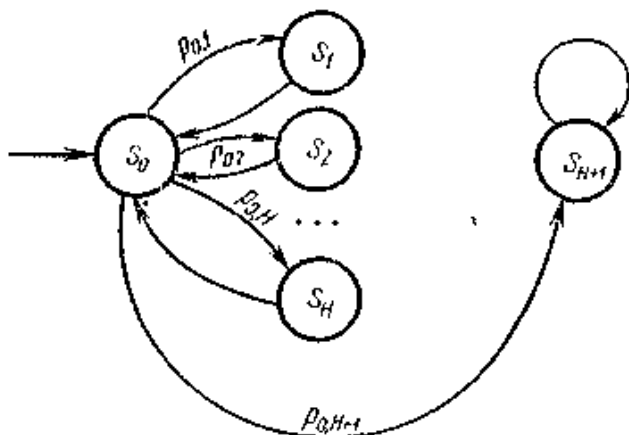


Рис.4.4. Граф марковской цепи, являющейся моделью облачного вычислительного процесса

Переходы между состояниями $S_0, \dots, S_{N+1}$ представляются на графе дугами, на которых обозначены вероятности переходов, отличные от 1. Значения вероятностей $p_{0,1}, \dots, p_{0,N+1}$ предопределяют ход облачного вычислительного процесса и зависят от параметров трудоемкости алгоритма. Эти значения вычисляются следующим образом. Трудоемкость алгоритма определяет, в частности, среднее число $N_1, \dots, N_N$ обращений к файлам $F_1, \dots, F_N$. Следовательно, среднее число переходов из состояния S_0 в состояния $S_1, \dots, S_N$ должно быть $(N_1 + \dots + N_N)$. Один раз процесс переходит из состояния S_0 в поглощающее состояние S_{N+1} . Таким образом, облачный вычислительный процесс должен выходить из состояния S_0 в среднем

$$N = \sum_{h=1}^N N_h + 1 \quad (4.3)$$

раз, т. е. в процессе реализации алгоритма этапы счета встречаются в среднем N раз. Значение $p_{0,h}$ определяет долю переходов в состояние S_h по отношению к всевозможным переходам из состояния S_0 в состояния $S_1, \dots, S_{N+1}$. Эта доля равна в среднем N_h/N , где N_h — среднее число переходов в состояние S_h . Следовательно,

$$p_{0,h} = N_h/N \quad (h=1, \dots, N);$$

$$p_{0,N+1} = 1/N. \quad (4.4)$$

Количество работы, выполняемой на каждом из этапов, характеризуется параметрами $\theta_1, \dots, \theta_H$ алгоритма. Значение θ определяет среднее количество процессорных операций, выполняемых за одну реализацию алгоритма, и значения $\theta_1, \dots, \theta_H$ — среднюю трудоемкость этапов, соответствующих состояниям $S_1, \dots, S_H$. Средняя трудоемкость этапа счета

$$\theta_0 = \theta / N, \quad (4.5)$$

где N — среднее число этапов счета, определяемое (4.3).

Трудоемкость каждого этапа будем рассматривать как случайную величину ϑ_h с математическим ожиданием θ_h , $h=0, \dots, H$; закон распределения случайных величин $\vartheta_0, \dots, \vartheta_H$ будем оговаривать особо.

Таким образом, под моделью облачного вычислительного процесса будем понимать марковскую цепь с $(H+2)$ состояниями, начальным состоянием S_0 и матрицей вероятностей переходов (4.2). *Реализация облачного вычислительного процесса* — случайная последовательность состояний $S_{t_0}, S_{t_1}, \dots, S_{t_m}$, порядок смены которых определяется в вероятностном смысле матрицей вероятностей переходов. С состояниями $S_0, \dots, S_H$ связано определенное количество работы, характеризуемое значениями случайных величин $\vartheta_0, \dots, \vartheta_H$ соответственно. Диаграмма облачного вычислительного процесса, порождаемого марковской моделью, изображена на рис. 4.5.

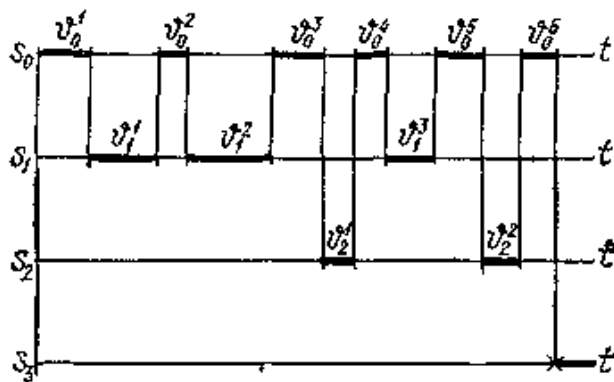


Рис.4.5. Временная диаграмма облачного вычислительного процесса, порождаемого марковской моделью

Значение ϑ_i^j определяет трудоемкость соответствующего этапа и представляет собой j -е значение случайной величины ϑ_i , математическое ожидание которой θ_i . В данном случае состояние S_3 является поглощающим: достигнув его, процесс прекращается.

Пример 4.1. Построим марковскую модель облачного вычислительного процесса, порождаемого алгоритмом, трудоемкость которого характеризуется следующими параметрами: $\theta = 100$ млн. операций; $N_1 = 19$ обращений к файлу F_1 ; $N_2 = 180$ обращения к файлу F_2 ; $\theta_1 = 2000$ байтов; $\theta_2 = 500$ байтов за обращение к файлу. В соответствии с (4.3) среднее число этапов счета при одном прогоне алгоритма $N = N_1 + N_2 + 1 = 200$. Вероятности перехода процесса из состояния счета S_0 в состояния S_1 , S_2 и S_3 определяются (4.4) и равны соответственно: $p_{0,1} = N_1/N = 19/200 = 0,095$; $p_{0,2} = N_2/N = 180/200 = 0,9$; $p_{0,3} = 1/N = 0,005$.

Подставляя эти значения в (4.2), получаем следующую матрицу вероятностей переходов:

$$P = \begin{matrix} & \begin{matrix} S_0 & S_1 & S_2 & S_3 \end{matrix} \\ \begin{matrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{matrix} & \begin{bmatrix} 0 & 0,095 & 0,9 & 0,005 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix}.$$

Из матрицы видно, что с вероятностью 0,095 за этапом счета произойдет обращение к файлу F_1 с вероятностью 0,9 — обращение к файлу F_2 и с вероятностью 0,005 вычислительный процесс прекратится.

Средняя трудоемкость этапа счета в соответствии с (4.5) $\theta_0 = \theta/N = 0,5$ млн. операций.

Свойства марковской модели облачных вычислительных процессов. Количество этапов в облачном вычислительном процессе (пребываний процесса в состоянии S_0) оценивается следующим образом. Процесс всегда начинается с этапа счета, поэтому количество этапов счета $n_0 \geq 1$. На следующем шаге с вероятностью $\gamma = p_{0,n+1}$ процесс может прекратиться, а с вероятностью $(1-\gamma)$ произойдет обращение к какому-то файлу F_n , после чего начнется второй этап счета. Эта ситуация может иметь место любое число раз, причем отсутствие последствия приводит к тому, что каждый раз процесс начинается как бы снова. В таком случае вероятность того, что при одной реализации процесса будут иметь место точно n_0 этапов счета,

$$Pr(n_0) = (1-\gamma)^{n_0-1} \gamma \quad (n_0 = 1, 2, \dots). \quad (4.6)$$

Первый сомножитель в (4.6) определяет вероятность того, что за $(n_0 - 1)$ этапов счета процесс не закончится, а второй сомножитель — вероятность того, что на n_0 -м этапе процесс перейдет в конеч-

ное состояние S_{H+1} . Распределение (4.6) — геометрическое распределение. Его математическое ожидание и дисперсия

$$E(n_0) = \sum_{n_0=1}^{\infty} n_0 \Pr(n_0) = 1/\gamma; D(n_0) = \sum_{n_0=1}^{\infty} n_0^2 \Pr(n_0) - [E(n_0)]^2 = 1/\gamma^2.$$

Поскольку $\gamma = p_0 = 1/N$, то среднее число этапов счета $E[n_0] = N$, где N определено (4.3). Характер распределения числа этапов счета иллюстрируется рис. 4.6.

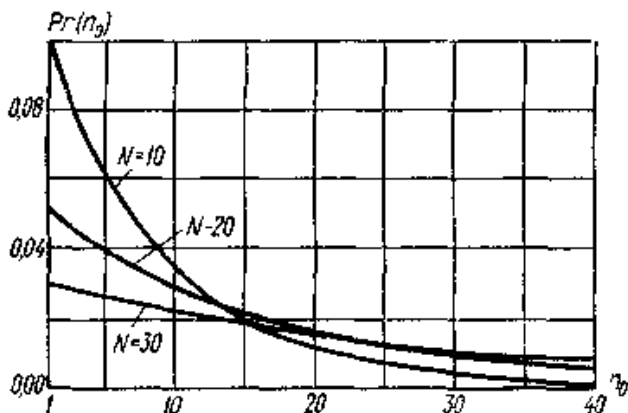


Рис. 4.6. Распределение числа этапов счета

Если среднее число этапов счета в облачном вычислительном процессе $N = 10$, то облачный вычислительный процесс ограничится одним этапом счета с вероятностью 0,1, десятью этапами — с вероятностью 0,037 и ста этапами — с вероятностью 0,000005. Вероятность того, что количество этапов счета в облачном вычислительном процессе не превысит среднего значения N , будет

$$\Pr(n_0 \leq N) = \sum_{n_0=1}^N \Pr(n_0) = 1 - (1 - \gamma)^N = 1 - (1 - 1/N)^N.$$

При $N=1$ эта вероятность равна 1. С увеличением N она уменьшается, стремясь к пределу

$$\lim_{N \rightarrow \infty} \Pr(n_0 \leq N) = 1 - e^{-1} \approx 0,63.$$

Таким образом, по крайней мере в 63% реализаций вычислительных процессов фактическое число этапов счета меньше математического ожидания, т. е. модель имеет тенденцию чаще генерировать короткие реализации.

Непрерывный аналог геометрического распределения — показательное распределение. Поэтому распределению (4.6) можно поставить в соответствие показательное распределение

$$Pr(n_0) = \gamma e^{-\gamma(n_0 - 1)} = e^{-(n_0 - 1)/N} / N \quad (4.7)$$

с математическим ожиданием N . Распределение (4.7) совпадает с (4.6) при $N \rightarrow \infty$, и при $N \geq 10$ погрешность аппроксимации не превосходит 10% для любых n_0 .

4.4. Оценка трудоемкости алгоритмов методами теории марковских цепей

Применительно к задачам анализа трудоемкости операторы алгоритма будем подразделять на функциональные, перехода и ввода — вывода. *Функциональный оператор* задает преобразование на множестве данных, т. е. задает некоторую совокупность вычислительных операций. *Оператор перехода* задает порядок вычисления значений предикатов и правило выбора одного из возможных путей развития облачного вычислительного процесса, соответствующего текущим значениям данных, отношения между которыми представляются предикатами. *Оператор ввода — вывода* задает обращение к определенному файлу с целью передачи некоторого количества информации. Функциональные операторы и операторы перехода задают совокупности вычислительных операций над данными и относятся к одному классу операторов, называемых *основными операторами*.

Совокупность операторов и связей между ними наиболее наглядно представляется *графом алгоритма*, который строится как композиция вершин, соответствующих операторам алгоритма, и дуг, отображающих связи между операторами. Выделяют вершины: начальные, конечные, операторные. *Начальная вершина* не имеет ни одного входа и имеет только один выход. Такая вершина определяет начало алгоритма. *Конечная вершина* имеет не менее одного входа и ни одного выхода; определяет конец алгоритма. *Операторная вершина* соответствует основному оператору или оператору ввода — вывода. Вершина, представляющая функциональный оператор или оператор ввода — вывода, может иметь любое не меньшее 1 число входов и только один выход. Вершина, представляющая оператор перехода, может

иметь любое не меньшее 1 число входов и не меньше двух выходов.

В любой ситуации оператор перехода определяет один и только один выход из представляющей его вершины.

Граф алгоритма является корректным, если выполняются следующие условия: 1) имеется только одна начальная и только одна конечная вершина; 2) для каждой вершины кроме начальной существует по крайней мере один путь, ведущий в эту вершину из начальной; 3) из каждой вершины кроме конечной существует по крайней мере один путь, ведущий из этой вершины в конечную; 4) выход из любой вершины должен вести только к одной вершине графа; 5) при любых значениях логических условий существует путь из начальной вершины в конечную, причем любому фиксированному набору значений условий соответствует только один такой путь.

Пример графа алгоритма приведен на рис. 4.7.

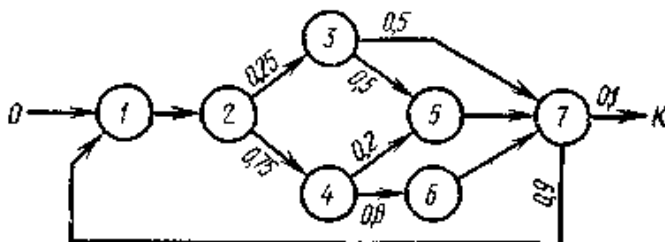


Рис.4.7. Граф алгоритма, используемый при оценке трудоемкости

Условимся вершины графа обозначать номерами $0, 1, \dots, K$: 0 соответствует начальной вершине графа; КС— конечной вершине; $1, \dots, K-1$ идентифицируют операторы алгоритма. В программировании графы алгоритмов изображаются с использованием набора фигур, обозначающих тип оператора, и называются *схемами алгоритмов (программ)*.

Граф выявляет структуру алгоритма, определяя множество операторов $V=\{V_1, \dots, V_{K-1}\}$ и дуг $D=\{(i, j)\}$, $i=0, \dots, K-1$ и $j=1, \dots, K$, связывающих операторы. Для оценки трудоемкости алгоритма необходимо, во-первых, разбить множество операторов на классы: основных операторов $S_0=\{V_{a_1}, \dots, V_{a_{m_0}}\}$, $V_{a_k} \in V$, операторов ввода — вывода

$$S_h=\{V_{b_1}, \dots, V_{b_{m_h}}\}, \quad h=1, \dots, H,$$

(каждый из этих операторов задает обращения к одному и тому же файлу F_h). Во-вторых, для каждого основного оператора V_a

необходимо определить среднее количество операций k_a , составляющих оператор, и для каждого оператора ввода — вывода V_β — среднее количество информации l_β , передаваемой при выполнении оператора. В-третьих, переходы между операторами V_i и V_j следует рассматривать как случайные события и характеризовать их вероятностями p_{ij} , т. е. каждая дуга (i, j) графа алгоритма должна быть отмечена вероятностью перехода p_{ij} , с которой переход из вершины V_i выполняется именно по этой дуге, т. е. к вершине V_j . Так как вычислительный процесс не может приостановиться в вершине V_i , то с вероятностью 1 произойдет переход к какой-либо вершине графа алгоритма. С учетом этого вероятности переходов должны отвечать условию

$$\sum_{i=1}^K p_{i,i} = 1, \quad (i=0, \dots, K-1). \quad (4.8)$$

Значения p_{ij} определяются вероятностями значений предикатов, зависящими от распределения значений данных, отношения между которыми задаются предикатами. Другими словами, вероятности p_{ij} зависят от вероятностей выполнения условия, проверяемого оператором i с целью выбора пути перехода. Например, пусть оператор 2 (рис.4.7) порождает переход к оператору 3 при отрицательном значении некоторой переменной X и к оператору 4 при положительном значении X . Если известно, что величина X равномерно распределена в диапазоне $(-1, +3)$, то с вероятностью 0,25 ее знак отрицателен и с вероятностью 0,75 положителен. Из этого следует, что переход к оператору 3 происходит с вероятностью $p_{2,3}=0,25$ и переход к оператору 4 — с вероятностью $p_{2,4} = 0,75$. Пусть далее оператор 7, замыкающий цикл, порождает переход к оператору 1 в девяти случаях, а в одном случае переход происходит в конец алгоритма (цикл, начинающийся от оператора 1 и заканчивающийся оператором 7, выполняется 10 раз). Тогда вероятности переходов $p_{7,1}=0,9$ и $p_{7,K} = 0,1$. Если за оператором i непременно выполняется оператор j , то $p_{ij} = 1$. Пусть $n_1, \dots, n_{K-1}$ — среднее число обращений к операторам $V_1, \dots, V_{K-1}$ за один прогон алгоритма. В таком случае характеристики трудоемкости могут быть вычислены следующим образом: среднее число операций, выполняемых при одном прогоне алгоритма,

$$\theta = \sum_{i \in S_0} n_i k_i; \quad (4.9)$$

среднее число обращений к файлу F_h

$$N_h = \sum_{i \in S_h} n_i \quad (h=1, \dots, H); \quad (4.10)$$

среднее количество информации, передаваемое при одном обращении к файлу F_h ,

$$\theta_h = (1/N_h) \sum_{v_i \in S_h} n_i l_i \quad (h=1, \dots, H). \quad (4.11)$$

В (4.9)—(4.11) суммирование выполняется по всем вершинам, относящимся к классу основных операторов S_0 или классу операторов ввода — вывода S_h , обращающихся к файлу F_h .

Таким образом, для оценки трудоемкости алгоритма необходимо определить среднее число обращений $n_1, \dots, n_{K-1}$ к операторам. Допустим, что вероятности переходов p_{kj} постоянны и после выполнения оператора $V_k (k = 1, \dots, K-1)$ переход к следующему оператору определяется только распределением вероятностей p_{kj} , т. е. не зависит от хода облачного вычислительного процесса в прошлом — до перехода к оператору V_k . При указанных допущениях процесс выполнения алгоритма является марковским процессом с K состояниями $S_1, \dots, S_K$, соответствующими пребыванию процесса в вершинах $V_1, \dots, V_K$ графа алгоритма. Состояния $S_1, \dots, S_{K-1}$ — невозвратные (процесс после какого-то числа шагов непременно покидает их), состояние S_K — поглощающее (достигнув его, процесс прекращается). Начальным является состояние S_i , определенное дугой $(0, i)$, выходящей из начальной вершины графа. Для упрощения обозначений условимся, что $i=1$, т. е. начальное состояние — состояние S_1 . Порядок изменения состояний определяется графом алгоритма, дуги которого отмечены вероятностями переходов p_{ij} . Отмеченный граф алгоритма можно рассматривать как граф марковской цепи. Среднее число $n_1, \dots, n_{K-1}$, пребываний марковского процесса в невозвратных состояниях $S_1, \dots, S_{K-1}$ определяется корнями системы линейных алгебраических уравнений

$$n_i = \delta_{1i} + \sum_{j=1}^{K-1} p_{ij} n_j \quad (i=1, \dots, K-1), \quad (4.12)$$

где δ_{ij} — символ Кронекера ($\delta_{ij} = 1$ при $i=j$ и $\delta_{ij}=0$ при $i \neq j$).

Уравнения (4.12) строятся исходя из следующих соображений.

Значение n_i будет равно по крайней мере 1, если процесс начинается от состояния с номером $i = 1$, что отмечено символом δ_{1j} . В остальных случаях процесс попадает в состояние S_i только из других состояний $j=1, \dots, K-1$ с вероятностями p_{ij} . Если процесс находился в состоянии j n_j раз и $p_{ij} \neq 0$, то процесс из этого состояния попадет в состояние i в среднем $p_{ij} n_j$ раз. Суммированием значений $p_{ij} n_j$ по всем j находится число попаданий процесса в состояние i из всех других состояний j .

Суть сетевого подхода состоит в выделении путей на графе алгоритма, соответствующих минимальной, средней и максимальной трудоемкости последовательности операторов. Эти пути могут быть выделены только на графах, не содержащих циклов. По этой причине методика сетевого подхода начинается с анализа трудоемкости алгоритмов, не содержащих циклы. Затем рассматривается прием исключения циклов из графа алгоритма путем замены их операторами с эквивалентной трудоемкостью. Этот прием позволяет распространить сетевой подход на любые алгоритмы, в том числе содержащие любое количество циклов.

Оценка средней трудоемкости алгоритмов. Алгоритм будем представлять в виде графа, состоящего из k операторных вершин и имеющего единственную конечную вершину с номером $K=k+1$; дуги графа отмечены вероятностями переходов p_{ij} . Каждой вершине графа поставлено в соответствие среднее значение трудоемкости k_i или l_j . С учетом (4.9)—(4.11) задача оценки трудоемкости алгоритма сводится к определению среднего числа $n_1, \dots, n_k$ обращений к операторам за один прогон алгоритма.

Рассмотрим методику вычисления значений $n_1, \dots, n_k$ для алгоритма, не содержащего циклы. Для применения сетевого подхода к оценке трудоемкости этого алгоритма вершины графа должны быть пронумерованы в порядке их следования, т. е. так, чтобы любая вершина имела номер, больший любого номера предшествующих ей вершин. Нумерация проводится следующим образом. Начальной вершине присваивается номер 0. Очередной номер $i=1, 2, \dots$ присваивается вершине, в которую входят дуги от уже пронумерованных вершин с номерами, меньшими i . При этом любым двум вершинам должны соответствовать разные номера. Такой порядок нумерации является результативным для любого графа без циклов. Причем конечная вершина графа будет иметь максимальный номер K . Пример корректной нумерации вершин графа приведен на рис.4.8.

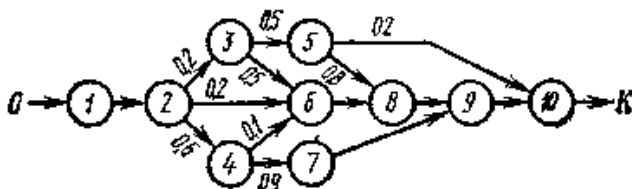


Рис.4.8. Граф алгоритма, не содержащего циклов

Для упрощения описания метода примем, что алгоритм состоит из однотипных операторов, например только основных операторов, разделим циклы по рангам. К рангу 1 относятся циклы, которые не содержат внутри себя ни одного цикла, к рангу 2 — циклы, внутри которых есть циклы ранга не выше ранга 1, и т. д. Например, алгоритм, изображенный на рис.4.9, содержит два цикла C_1 и C_2 ранга 1 и один цикл C_3 ранга 2. Совокупность операторов, входящих в цикл, и связывающих их дуг, за исключением дуги, замыкающей цикл, называют *телом цикла*. Тело цикла ранга 1 является графом без циклов. Применяя к этому графу вышеописанную методику, можно определить значения n_i для каждого из операторов, принадлежащих телу цикла, и, следовательно, трудоемкость тела цикла C равна $\sum_{v_j \in C} k_j n_j$. Здесь суммирование проводится по всем вершинам V_j , содержащимся в цикле C .

Пусть известно среднее число повторений цикла n_C , равное числу выполнений тела цикла при одном прогоне алгоритма. Если вероятность перехода по дуге, замыкающей цикл, равна p_{kl} , то $n_C = 1 + p_{kl} n_C$, откуда $n_C = 1/(1 - p_{kl})$. В таком случае средняя трудоемкость цикла

$$k_C = n_C \sum_{v_j \in C} k_j n_j$$

и цикл C можно заменить оператором с трудоемкостью k_C .

Применяя указанную процедуру замены циклов операторами ко всем циклам ранга 1, затем к циклам ранга 2 и т. д., в конце концов приходим к графу без циклов, трудоемкость которого находится вышеописанным способом.

Пример 4.4. Определим трудоемкость алгоритма, заданного графом, изображенным на рис. 4.9.

Положим, что трудоемкость всех операторов одинакова и равна 1. Средней число повторений циклов C_1 , C_2 и C_3 определяется из вероятностей переходов, указанных на рис. 4.9, следующими значениями: $n_{C_1} = 5$, $n_{C_2} = 20$ и $n_{C_3} = 10$.

Выделим тела циклов C_1 и C_2 ранга 1. Их структура изображена на рис.4.10, а, б.

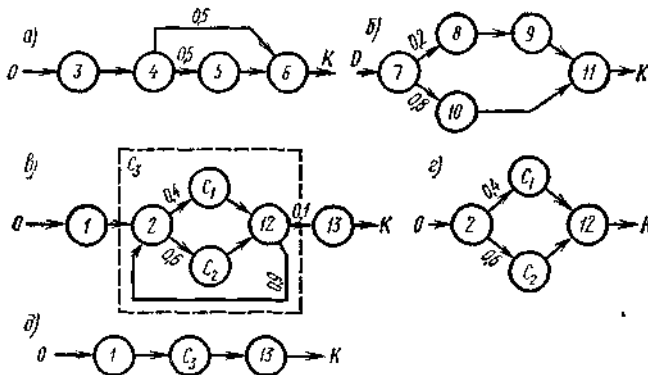


Рис.4.10. К определению времени выполнения программы, содержащей циклы

Применяя вышеописанную методику к этим графам, определяем среднюю трудоемкость k_{C_1} и k_{C_2} выполнения тел этих циклов:

$$\begin{aligned}
 n_3 &= 1; & n_7 &= 1; \\
 n_4 &= p_{3,4} n_3 = 1; & n_8 &= p_{7,8} n_7 = 0,2; \\
 n_5 &= p_{4,5} n_4 = 0,5; & n_9 &= p_{8,9} n_8 = 0,2; \\
 n_6 &= p_{4,6} n_4 + p_{5,6} n_5 = 1; & n_{10} &= p_{7,10} n_7 = 0,8; \\
 \sum_{i=3}^6 k_i n_i &= 3,5; & n_{11} &= p_{9,11} n_9 + p_{10,11} n_{10} = 1; \\
 & & \sum_{i=7}^{11} k_i n_i &= 3,2.
 \end{aligned}$$

Средняя трудоемкость циклов C_1 и C_2 вычисляется умножением полученных значений на среднее число повторений этих циклов:

$$k_{C_1} = 17,5 \text{ и } k_{C_2} = 64 \text{ операций.}$$

Заменяя в графе (см. рис.4.9) циклы C_1 и C_2 операторами C_1 и C_2 с трудоемкостью k_{C_1} и k_{C_2} , получим граф, вид которого показан на

рис.4.10, в. Тело цикла C_3 имеет вид, изображенный на рис.4.10, г.

Трудоемкость тела цикла C_3 определяется следующим образом: $n_2 = 1$;

$$n_{C1} = p_{2,C1} n_2 = 0,4; \quad n_{C2} = p_{2,C2} n_2 = 0,6;$$

$$n_{12} = p_{C1,12} n_{C1} + p_{C2,12} n_{C2} = 1; \quad \sum_{v_i \in C_3} k_i n_i = 1 + 17,5 \cdot 0,4 + 64 \cdot 0,6 + 1 = 47,4.$$

С учетом числа n_{C3} повторений цикла трудоемкость цикла составит $k_{C_3} = 47,4$ операции.

Заменяв цикл C_3 оператором C_3 с трудоемкостью k_{C_3} , получим граф, имеющий вид, показанный на рис.4.10,д. Трудоемкость алгоритма,

представляемого этим графом,

$$\theta = k_1 + k_{c_1} + k_{13} = 1 + 474 + 1 = 476 \text{ операций.}$$

Оценка минимальной и максимальной трудоемкости алгоритма.

Рассмотрим методику вычисления минимальной и максимальной трудоемкости алгоритмов. Вначале будем рассматривать алгоритмы без циклов, вершины графов которых нумеруют в порядке их следования, как это было описано ранее.

Минимально возможное и максимально возможное значения трудоемкости на момент окончания выполнения оператора V_i обозначим соответственно через A_i и B_i . Имеем $A_0 = 0$ и $B_0 = 0$. Тогда для остальных вершин с номерами $i=1, 2, \dots, k$

$$A_i = \min_{(j, i) \in D} (A_j) + k_{i \min}; \quad (4.15)$$

$$B_i = \max_{(j, i) \in D} (B_j) + k_{i \max}, \quad (4.16)$$

где (j, i) — дуга, выходящая из вершины j и входящая в вершину i ; D — множество дуг графа программы; минимальное $\min (A_j)$ и максимальное $\max (B_j)$ значения определяются по отношению ко всем вершинам j , из которых выходят дуги, входящие в вершину i ; значения $k_{i \min}$ и $k_{i \max}$ характеризуют минимальную и максимальную трудоемкость оператора V_i .

Для конечной вершины K графа вычисляются значения

$$A_K = \min_{(j, K) \in D} (A_j); \quad B_K = \max_{(j, K) \in D} (B_j), \quad (4.17)$$

характеризующие минимальную и максимальную трудоемкость алгоритма.

Пример 4.5. Определим минимальную и максимальную трудоемкость алгоритма, изображенного на рис.4.8.

Примем, что трудоемкость каждого оператора постоянна и равна 1. Нумерация вершин на графе рис. 4.8 отвечает вышерассмотренному правилу.

Последовательно применяя (4.15)—(4.16), получим:

$$\begin{array}{ll}
 A_0=0; & B_0=0; \\
 A_1=\min(A_0)+1=1; & B_1=\max(B_0)+1=1; \\
 A_2=\min(A_1)+1=2; & B_2=\max(B_1)+1=2; \\
 A_3=\min(A_2)+1=3; & B_3=\max(B_2)+1=3; \\
 A_4=\min(A_3)+1=3; & B_4=\max(B_3)+1=3; \\
 A_5=\min(A_4)+1=4; & B_5=\max(B_4)+1=4; \\
 A_6=\min(A_2, A_3, A_4)+1=3; & B_6=\max(B_2, B_3, B_4)+1=4; \\
 A_7=\min(A_4)+1=4; & B_7=\max(B_4)+1=4; \\
 A_8=\min(A_5, A_6)+1=4; & B_8=\max(B_5, B_6)+1=5; \\
 A_9=\min(A_7, A_8)+1=5; & B_9=\max(B_7, B_8)+1=6; \\
 A_{10}=\min(A_8, A_9)+1=5; & B_{10}=\max(B_5, B_9)+1=7; \\
 A_K=\min(A_{10})=5; & B_K=\max(B_{10})=7.
 \end{array}$$

Таким образом, минимальная трудоемкость алгоритма A_K равна 5, а максимальная $B_K=7$ операциям.

Минимальная и максимальная трудоемкость алгоритмов, содержащих циклы, находится по аналогии с методом определения средней трудоемкости алгоритмов с циклами. При этом выделяются циклы ранга 1. Находится минимальная A и максимальная B трудоемкости тела цикла. Минимальная и максимальная трудоемкость цикла определяется значениями $An_{\min}$ и $Bn_{\max}$, где $n_{\min}$ и $n_{\max}$ — минимальное и максимальное число повторений цикла. Затем цикл заменяется оператором с трудоемкостью $k_{\min}=An_{\min}$ и $k_{\max}=Bn_{\max}$ и вновь применяется процедура исключения циклов. Процесс повторяется до тех пор, пока граф алгоритма не будет преобразован к форме без циклов.

Пример 4.6. Определим минимальную и максимальную трудоемкость алгоритма, граф которого изображен на рис.4.9. Трудоемкость всех операторов постоянна и равна 1. Минимальное и максимальное число повторений циклов C_1 и C_2 представляются следующими парами значений: $n_1=(3; 7)$; $n_2=(15; 25)$. Число повторений цикла C_3 постоянно и равно 10.

Выделяем тела циклов C_1 и C_2 , структура которых представлена графами рис.4.10, а, б. Минимальная и максимальная трудоемкость этих частей алгоритма определяется так:

для графа рис. 4.10, а —

$$\begin{array}{ll}
 A_0=0; & B_0=0; \\
 A_3=\min(A_0)+1=1; & B_3=\max(B_0)+1=1; \\
 A_4=\min(A_3)+1=2; & B_4=\max(B_3)+1=2; \\
 A_5=\min(A_4)+1=3; & B_5=\max(B_4)+1=3; \\
 A_6=\min(A_4, A_5)+1=3; & B_6=\max(B_4, B_5)+1=4; \\
 A_{C_1}=3; & B_{C_1}=4;
 \end{array}$$

для графа рис.4.10,б—

$$\begin{array}{ll}
 A_0=0; & B_0=0; \\
 A_7=\min(A_0)+1=1; & B_7=\max(B_0)+1=1; \\
 A_8=\min(A_7)+1=2; & B_8=\max(B_7)+1=2; \\
 A_9=\min(A_8)+1=3; & B_9=\max(B_8)+1=3; \\
 A_{10}=\min(A_7)+1=2; & B_{10}=\max(B_7)+1=2; \\
 A_{11}=\min(A_9, A_{10})+1=3; & B_{11}=\max(B_9, B_{10})+1=4; \\
 A_{C_1}=3; & B_{C_1}=4.
 \end{array}$$

Трудоемкость циклов C_1 и C_2 представляется следующими минимальными и максимальными значениями:

$$\begin{array}{l}
 k_{C_1 \min} = A_{C_1} n_1 \min = 9; \quad k_{C_2 \min} = A_{C_2} n_2 \min = 45; \\
 k_{C_1 \max} = B_{C_1} n_1 \max = 28; \quad k_{C_2 \max} = B_{C_2} n_2 \max = 100.
 \end{array}$$

Заменив в графе на рис. 2.10 циклы C_1 и C_2 эквивалентными им операторами C_1 и C_2 , получим граф рис.4.10, в. Тело цикла C_3 имеет структуру, изображенную на рис.4.10,г. Минимальная и максимальная трудоемкость тела цикла

$$\begin{array}{ll}
 A_0=0; & B_0=0; \\
 A_2=\min(A_0)+1=1; & B_2=\max(B_0)+1=1; \\
 A_{C_1}=\min(A_2)+k_{C_1 \min}=10; & B_{C_1}=\max(B_2)+k_{C_1 \max}=29; \\
 A_{C_2}=\min(A_2)+k_{C_2 \min}=46; & B_{C_2}=\max(B_2)+k_{C_2 \max}=101; \\
 A_{12}=\min(A_{C_1}, A_{C_2})+1=11; & B_{12}=\max(B_{C_1}, B_{C_2})+1=102; \\
 A_{C_3}=11; & B_{C_3}=102.
 \end{array}$$

Заменив цикл C_3 эквивалентным оператором C_3 , получаем граф рис.4.10,д, минимальная и максимальная трудоемкость которого

$$\begin{array}{l}
 A_K = k_1 \min + k_{C_3 \min} + k_{12 \min} = 112; \\
 B_K = k_1 \max + k_{C_3 \max} + k_{12 \max} = 1022.
 \end{array}$$

Средняя трудоемкость этого алгоритма, вычисленная в примере 4.5, составляла 476 операций.

4.6. Оценка дисперсии трудоемкости алгоритмов

Для обеспечения необходимой точности решения некоторых задач анализа ОВС необходимы сведения о дисперсии трудоемкости алгоритма. Дисперсия характеризует степень разброса значений случайной величины относительно своего среднего значения и. чем больше дисперсия, тем больше разброс значений.

Процедура определения дисперсии трудоемкости сложная. Поэтому рассмотрим только формальную сторону этого вопроса .

Для вычисления дисперсии алгоритм представляется как марковская цепь с матрицей вероятностей переходов

$$P = [p_{ij}] = \begin{matrix} & \begin{matrix} 1 & 2 & \dots & K \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ \dots \\ K \end{matrix} & \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1K} \\ p_{21} & p_{22} & \dots & p_{2K} \\ \dots & \dots & \dots & \dots \\ p_{K1} & p_{K2} & \dots & p_{KK} \end{bmatrix} \end{matrix}.$$

Здесь K — количество операторов алгоритма и элементы p_{ij} определяют вероятности перехода из состояний $i=1, \dots, K$ в состояния $j=1, \dots, K$. Например, алгоритму рис. 4.7 соответствует матрица вероятностей переходов

$$P = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,25 & 0,75 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,5 & 0 & 0,5 \\ 0 & 0 & 0 & 0 & 0,2 & 0,8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0,9 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}.$$

Трудоемкость операторов представим вектор-столбцом:

$$L = \begin{bmatrix} l_1 \\ l_2 \\ \dots \\ l_K \end{bmatrix}.$$

Для определения дисперсии необходимо вычислить квадратную матрицу $N = [n_{ij}] = (I - P)^{-1}$, где

$$I = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}_K.$$

— единичная матрица; $(I - P)$ — поэлементная разность матриц I и P , представляющая собой матрицу K -го порядка:

$$(I - P) = \begin{bmatrix} (1 - p_{11}) & -p_{12} & \dots & -p_{1K} \\ -p_{21} & (1 - p_{22}) & \dots & -p_{2K} \\ \dots & \dots & \dots & \dots \\ -p_{K1} & -p_{K2} & \dots & (1 - p_{KK}) \end{bmatrix}.$$

Обращением матрицы $(I - P)$ формируется матрица N , элементы n_{ij} которой имеют смысл среднего числа попаданий процесса в состояния

с номерами $j=1, \dots, K$, если процесс начинается из состояний с номерами $i=1, \dots, K$.

Таким образом, если алгоритм начинает выполняться от оператора с номером $i=1$, то количество обращений к операторам $1, \dots, K$ задается первой строкой $n_{11}, \dots, n_{1K}$ матрицы $\mathbf{N}$. Умножив матрицу $\mathbf{N}$ на вектор-столбец $\mathbf{L}$, получим вектор-столбец

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dots \\ \theta_K \end{bmatrix},$$

элементы которого определяют среднюю трудоемкость алгоритма, инициируемого от операторов $V_1, \dots, V_K$ соответственно. Значение дисперсии можно определить следующим матричным выражением:

$$\mathbf{D} = \mathbf{N} (2\mathbf{L}_{dq} \mathbf{P} \theta + \mathbf{L}_{Sq}) - \theta_{Sq}. \quad (4.18)$$

Здесь $\mathbf{L}_{dq}$ — диагональная матрица

$$\mathbf{L}_{dq} = \begin{bmatrix} l_1 & 0 & \dots & 0 \\ 0 & l_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & l_K \end{bmatrix},$$

получаемая из элементов вектора $\mathbf{L}$; $\mathbf{L}_{Sq}$, θ_{Sq} — вектор-столбцы вида

$$\mathbf{L}_{Sq} = \begin{bmatrix} l_1^2 \\ l_2^2 \\ \dots \\ l_K^2 \end{bmatrix}, \quad \theta_{Sq} = \begin{bmatrix} \theta_1^2 \\ \theta_2^2 \\ \dots \\ \theta_K^2 \end{bmatrix},$$

определяемые путем возведения в квадрат элементов векторов $\mathbf{L}$ и θ . Элементы вектор-столбца

$$\mathbf{D} = \begin{bmatrix} D_1 \\ D_2 \\ \dots \\ D_K \end{bmatrix}$$

определяют дисперсию времени выполнения программы, начинающейся от операторов с номерами $V_1, \dots, V_K$ соответственно. Если алгоритм выполняется всегда с первого оператора, то дисперсия задается первым элементом D_1 вектора $\mathbf{D}$.

Пример 4.7. Пусть требуется вычислить дисперсию трудоемкости алгоритма рис. 4.11.

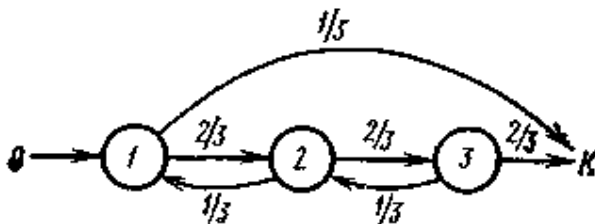


Рис.4.11. К вычислению дисперсии трудоемкости алгоритма

Трудоемкость операторов составляет $l_1 = 3$, $l_2 = 1$ и $l_3 = 2$ операций
По графу рис. 4.11 строим матрицу вероятностей переходов:

$$P = \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} 0 & 2/3 & 0 \\ 1/3 & 0 & 2/3 \\ 0 & 1/3 & 0 \end{bmatrix} \end{matrix}.$$

Вычисляем матрицу N:

$$N = (I - P)^{-1} = \left[\begin{array}{ccc|c} 1 & -2/3 & 0 & 1 \\ -1/3 & 1 & -2/3 & 1 \\ 0 & -1/3 & 1 & 1 \end{array} \right]^{-1} = \begin{bmatrix} 1,4 & 1,2 & 0,8 \\ 0,6 & 1,8 & 1,2 \\ 0,2 & 0,6 & 1,4 \end{bmatrix}.$$

Определяем вектор средней трудоемкости алгоритма:

$$\theta = N \cdot L = \begin{bmatrix} 1,4 & 1,2 & 0,8 \\ 0,6 & 1,8 & 1,2 \\ 0,2 & 0,6 & 1,4 \end{bmatrix} \times \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 6 \\ 4 \end{bmatrix}.$$

Первый элемент $\theta_1 = 7$ вектора определяет среднюю трудоемкость алгоритма, инициируемого от оператора 1, элемент $\theta_2 = 6$ — трудоемкость алгоритма, инициируемого от оператора 2, и т. д.
Применяя (4.18), находим

$$\begin{aligned} D &= \begin{bmatrix} 1,4 & 1,2 & 0,8 \\ 0,6 & 1,8 & 1,2 \\ 0,2 & 0,6 & 1,4 \end{bmatrix} \times \left\{ 2 \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 2 \end{bmatrix} \times \begin{bmatrix} 0 & 2/3 & 0 \\ 1/3 & 0 & 2/3 \\ 0 & 1/3 & 0 \end{bmatrix} \times \right. \\ &\quad \left. \times \begin{bmatrix} 7 \\ 6 \\ 4 \end{bmatrix} + \begin{bmatrix} 9 \\ 1 \\ 4 \end{bmatrix} \right\} - \begin{bmatrix} 49 \\ 36 \\ 16 \end{bmatrix} = \begin{bmatrix} 20 \\ 18 \\ 14 \end{bmatrix}. \end{aligned}$$

Среднеквадратичное отклонение трудоемкости алгоритма, инициируемого от первого оператора, $\sigma = \sqrt{D_1} = \sqrt{20} \approx 4,5$.

5. Облачный управляющий процесс, его математические модели и его моделирование.

5.1. Основные понятия и определения

При функционировании системы в ней выполняются определенные процессы. Существует ряд определений понятия «процесс». Мы под *процессом* будем понимать выполняемую совокупность действий, обеспечивающую направленное решение определенной задачи и могущая на основе исходного материала (определенных правил) получить вполне определенный продукт.

При этом можно выделить два основных типа процессов: *детерминированный* и *стохастический*. Если при применении для одного и того же исходного материала одного и того же детерминированного процесса будет получен один и тот же продукт, то для одного и того же исходного материала, применяя несколько раз один и тот же стохастический процесс, вообще говоря, можно получить различные продукты. Однако при многократном использовании для одного и того же исходного материала одного и того же стохастического процесса можно получить с определенной достоверностью в среднем один и тот же продукт. Этим свойством стохастического процесса широко пользуются, например, при проведении статистического моделирования

Частным случаем процесса может служить выполняемая на ЭВМ программа с обрабатываемыми данными. При этом исходным материалом в данном случае являются исходные данные, а продуктом — результирующие. Другим частным примером процесса может служить процесс изготовления какой-либо детали. В этом случае исходным материалом служит заготовка, а продуктом — сама деталь.

Следует заметить, что понятие процесса является понятием динамическим, т. е. процесс существует только в том случае, если происходит его выполнение. Поэтому говорят, что *процесс зарождается, когда он начинает выполняться, и процесс умирает, когда его выполнение завершается*.

Процессы могут выполняться как в объектах управления (ОУ), так и в управляющих средствах (УС).

Рассмотрим вначале процессы, выполняемые в ОУ.

Для обеспечения взаимосвязи различных проблемных блоков как единого ОУ между проблемными блоками создается система технологических связей (СТС) .

Последовательность выполнения технологических операций на каждом проблемном блоке будем называть *технологическим процессом*.

Для активизации тех или иных действий, необходимых для выполнения технологических операций и обеспечения заданной последовательности их работы создается система управления. В качестве системы управления может использоваться УС. Тогда будем говорить, что система является автоматической системой управления .

Систему управления, которая строится на основе одной или нескольких ЭВМ и в процессе ее функционирования активную роль играет оператор (человек), будем называть *автоматизированной системой управления (АСУ)*. Если АСУ обеспечивает управление *технологическим процессом* (ТП), ее называют АСУТП.

Выполняемый в УС или АСУ процесс, обеспечивающий управление технологическим процессом, будем называть *управляющим процессом* или *процессом управления*.

Технологический процесс и соответствующий ему управляющий процесс можно выделить в любой системе УС—ОУ (или АСУ—ОУ), в частности в узле коммутации (УК) сети связи, где в качестве ОУ можно рассматривать коммутационную систему, а в качестве АСУ или УС — программное управляющее устройство. В дальнейшем, если не будет особой необходимости, для простоты будем использовать в основном термин УС. Вместо понятия «технологическая операция» применительно к УК обычно используют понятие «операция по обслуживанию заявки (вызова) на установление соединения», вместо понятия РМ — блок или прибор коммутационной системы, а вместо транспортера — электрические соединения (в частности, промежуточные шнуры) коммутационной системы. При этом заданная последовательность выполнения отдельных операций по обслуживанию заявки на установление соединения может быть также названа технологическим процессом, хотя в автоматической электросвязи обычно используют другой термин — процесс обслуживания заявки (вызова) на установление соединения. В ряде случаев вместо технологического процесса удобно использовать термин «объектный процесс».

В ОУ одновременно может выполняться несколько процессов или частей одного и того же технологического процесса. В этом случае будем говорить, что в ОУ выполняется несколько параллельных

технологических процессов или один и тот же технологический процесс имеет несколько параллельных участков (технологических подпроцессов, или частных технологических процессов).

Технологический процесс, в котором допускается одновременное (параллельное) выполнение частных технологических процессов, будем называть *параллельным технологическим процессом*. Технологический процесс, в котором отсутствуют частные технологические процессы, выполнение которых может быть осуществлено параллельно, будем называть *последовательным технологическим процессом*.

Легко понять, что если в ОУ параллельно выполняется r последовательных технологических процессов, то без ограничения общности для простоты будем считать, что в ОУ выполняется один параллельный технологический процесс с r параллельно выполняемыми частными технологическими процессами.

Аналогично *технологический процесс* будем называть *параллельным*, если в нем допускается параллельное выполнение частных консультационных процессов. При этом так же, как и для технологического процесса, примем, что в УС выполняется один параллельный технологический процесс, если в УС одновременно выполняет несколько технологических процессов, которые будем называть *частными технологическими процессами*. Если в технологическом процессе нет параллельно выполняемых частных технологических процессов (технологических подпроцессов), то такой *технологический процесс* будем называть *последовательным*.

С каждой технологической операцией, из которых состоит технологический процесс, в технологическом процессе может быть сопоставлена *процедура*, начало выполнения которой определяет начало выполнения соответствующей технологической операции, а ее окончание — завершение этой технологической операции.

Условно процедуру можно представить в виде трех частей: *начальная, основная и заключительная*. Начальная (пусковая) часть процедуры обеспечивает активизацию выполнения соответствующей технологической операции в ОУ. Основная часть (тело процедуры) управляет ходом выполнения технологической операции, а заключительная часть обеспечивает останов выполнения технологической операции. По окончании и в процессе выполнения технологической операции процедура при необходимости может обеспечить снятие показаний датчиков ОУ и в зависимости от их значений может обеспечить скорректированное продолжение выполнения этой технологической операции. В частности, в процедуре может быть предусмотрен контроль за правильностью

выполнения в ОУ технологической операции после завершения каждой из трех указанных выше частей процедуры.

В качестве исполнительных средств ОУ могут использоваться разнообразные устройства и приборы, имеющие различные свойства.

По способу восприятия сигнала управления УС исполнительные средства можно разделить на два класса: *с фиксацией и без фиксации воздействия*.

При применении в ОУ исполнительных средств с фиксацией воздействия в УС может реализовываться такой процесс управления, процедуры которого имеют только две части: начальную, обеспечивающую включение некоторого исполнительного средства, и заключительную, обеспечивающую выключение этого исполнительного средства. Такое исполнительное средство будем называть *пусковым*.

В данном случае предполагается, что после активизации технологической операции блок ОУ при включенном пусковом исполнительном средстве сам обеспечивает контроль за ходом ее выполнения и УС необходимо после ее завершения только выключить пусковое исполнительное средство, приостановив выполнение технологической операции, или просто установить факт завершения выполнения технологической операции, что обеспечивается заключительной частью процедуры. Такие блоки ОУ будем называть *активными*.

Те блоки ОУ, которые требуют со стороны УС постоянного управления и контроля за ходом выполнения технологической операции в нем, будем называть *пассивными*.

Пусть в качестве блока ОУ рассматривается электродвигатель с компрессором, нагнетающим воздух в резервуар. Если электродвигатель включается электромеханическим реле, т. е. пусковым исполнительным средством, которое после срабатывания блокируется, то после подачи из УС сигнала включения пускового реле последнее запускает электродвигатель и, оставаясь заблокированным, обеспечивает непрерывную работу электродвигателя. Если при этом необходимо контролировать какие-либо параметры, например давление в резервуаре, создаваемое компрессором, приводимым в действие электродвигателем, то после создания необходимого давления в резервуаре датчик, сработав, отключит от электропитания электродвигатель, который после этого остановится. Если в резервуаре давление упадет ниже нормы, то датчик вновь создаст цепь электропитания электродвигателя. Таким образом, необходимое давление в резервуаре будет поддерживаться до тех пор, пока из УС не

поступит сигнал на выключение пускового реле — это есть пример активного блока ОУ.

Управляющие процессы, процедуры которых не имеют тела, а включают только начальную и заключительную части, будем называть *стартстопными*.

Частным случаем стартстопных управляющих процессов является процесс, процедуры которого имеют только начальные части. В этом случае УС подает только сигнал активизации выполнения технологической операции в блок ОУ, после чего блок ОУ функционирует автономно и выключается после завершения выполнения технологической операции. При этом предполагается, что время выполнения i -й технологической операции ограничено некоторым значением t_i . Поэтому УС нет нужды посылать в блок ОУ сигнал остановки выполнения технологической операции. При этом в УС реализуется такой процесс, в котором после начала выполнения i -й процедуры переход к следующей, за ней $(i + 1)$ -й (соседней) процедуры произойдет через время t_i .

Процесс, процедуры которого имеют только начальные части, а интервалы времени между двумя соседними процедурами π_i и π_{i+1} составляют t_i , будем называть *стартовым управляющим процессом*.

Стартовый процесс, у которого время t_i постоянно и одинаково для всех соседних пар имеющихся в процессе процедур, будем называть *синхронным стартовым управляющим процессом*.

При реализации синхронного стартового управляющего процесса в УС, очевидно, необходимо иметь тактовый генератор.

Если время t_i не является постоянным и тем более неопределенным, то стартовый управляющий процесс реализовать невозможно, поэтому вместо стартового процесса можно использовать стартстопный или обычный управляющий процесс с тремя частями в каждой процедуре. В этом случае будем говорить, что УС реализуется *простой управляющий процесс*. Если в дальнейшем не будет специально оговорен вид процесса, то это будет означать, что подразумевается простой управляющий процесс.

Если реализуется в УС стартстопный управляющий процесс и длительности выполнения каждой из трех частей его процедур постоянны и одинаковы, то может быть реализован такой управляющий процесс, при котором моменты начала выполнения начальной и заключительной частей процедуры будут задаваться тактовым генератором. В этом случае будем говорить, что в УС реализуется *синхронный стартстопный управляющий процесс*.

Синхронный стартстопный управляющий процесс можно использовать и в том случае, когда времена выполнения тела различных процедур

различны, но возможно установить максимальное время выполнения тела процедуры. Тогда интервалы времени между сигналами от тактового генератора будут определяться этим максимальным временем. Если же времена выполнения тел процедур не только различны, но и непостоянны и неопределенны, то вместо синхронного стартстопного управляющего процесса можно использовать *квазисинхронный стартстопный управляющий процесс*, при котором выполнение заключительной части процедуры повторяется через равные интервалы времени, определяемые тактовым генератором, до тех пор, пока не будет установлено, что тело процедуры выполнено, т. е. в ОУ закончено выполнение соответствующей технологической операции.

Вместо квазисинхронного стартстопного управляющего процесса в этом случае может быть использован и *асинхронный стартстопный управляющий процесс*. Однако в последнем случае из ОУ необходимо посылать сигнал об окончании выполнения технологической операции. При этом после завершения выполнения начальной части процедуры активизируется заключительная ее часть, которая ожидает появления на входе УС сигнала от ОУ об окончании соответствующей технологической операции.

Таким образом, при выполнении в УС асинхронного стартстопного управляющего процесса из ОУ должны посылаться в УС сигналы об окончании технологических операций. Если же реализуется синхронный стартстопный управляющий процесс, то эти сигналы имитируются таковым генератором.

Простой управляющий процесс также может быть *синхронным и асинхронным*.

Следует заметить, что в большинстве случаев целесообразно в УС реализовывать *смешанный управляющий процесс*, в котором сочетаются все или часть указанных выше разновидностей управляющих процессов.

В процессе выполнения управляющий процесс использует программно-аппаратурные ресурсы УС. Так, для того чтобы в ЭВМ выполнялась некоторая программа, должен использоваться процессор, заниматься один или несколько блоков оперативного запоминающего устройства. Возможно использование также и других ресурсов ЭВМ. В том случае, когда в ЭВМ используется многопрограммный режим работы, то один и тот же процессор, блок памяти или канал ввода-вывода может затребоваться различными программами. Поэтому возникает их конкуренция при занятии того или иного ресурса, в частности процессора.

При наличии нескольких процессоров конкуренция параллельно выполняемых в них программ (процессов) может возникнуть при занятии одного и того же блока ОЗУ. Конкуренция процессов может возникнуть и при управлении формированием рекомендаций. Таким образом, конкуренция управляющих процессов может возникнуть не только от использования общих ресурсов УС, но и из-за общих ресурсов ОУ.

Для того чтобы устранить конкуренцию параллельно выполняемых процессов при занятии некоторого общего ресурса, необходимо предусмотреть определенные меры.

Процесс (технологический, управляющий) будем называть *правильно построенным*, или *правильным процессом*, если в нем отсутствует конкуренция параллельных частных процессов из-за занятия общих ресурсов.

Легко понять, что если в технологическом и в управляющем процессах нет параллельно выполняемых участков (частных процессов, подпроцессов), то конкуренции частных процессов из-за общих ресурсов не может быть — общие ресурсы используются частными процессами последовательно. Таким образом, если в ОУ выполняется последовательный технологический процесс, т. е. процесс без параллельных участков, или такой параллельный консультационный процесс, в котором параллельные участки (подпроцессы) не требуют общих ресурсов ОУ, то, очевидно, любой управляющий процесс для такого технологического процесса свободен от конкуренции из-за общих ресурсов ОУ. Если же в ОУ допускается выполнение параллельного технологического процесса, в котором при выполнении подпроцессов может возникнуть необходимость использования общего ресурса, то даже в последовательном управляющем процессе возможна конкуренция подпроцессов из-за общих ресурсов ОУ. В этом случае требуется перевод технологического процесса в правильный для обеспечения устранения конкуренции подпроцессов в управляющем процессе из-за общих ресурсов ОУ.

Однако не всегда целесообразно при формировании технологического процесса добиваться того, чтобы он был правильным. Может оказаться целесообразным перевод его в правильный осуществлять в процессе разработки управляющего процесса. При этом конкуренция подпроцессов технологического процесса и подпроцессов управляющего процесса из-за общих ресурсов ОУ может быть устранена выполнением технологических операций, которые требуют одних и тех же ресурсов ОУ.

Если используется параллельный управляющий процесс, то в нем может быть конкуренция подпроцессов за счет общих ресурсов ОУ.

Таким образом, в качестве исходных данных при построении правильного (корректного) управляющего процесса, а затем и управляющего алгоритма (алгоритма функционирования УС) служит технологический процесс. Процесс построения управляющего алгоритма на основе технологического процесса из-за сложности может быть представлен в виде *двух основных этапов*.

На первом этапе формируется управляющий процесс, в котором устраняются все конкуренции из-за общих ресурсов ОУ. При этом на основе параллельного технологического процесса может быть построен как параллельный, так и последовательный управляющие процессы. На втором этапе на основе правильного (корректного) управляющего процесса формируется управляющий алгоритм (алгоритм функционирования УС), который рассматривается в виде условий работы УС и является исходным данным для его синтеза.

5.2. Формальное писание и преобразование процессов

5.2.1. Исчисление высказываний как язык описания процессов

Развитие методов построения логических схем позволило создать методы построения систем автоматизированного проектирования процессов. Рассмотрим аппарат математической логики, открывший широкие возможности для построения схем автоматизированных процессов.

Логика является наукой о формах и законах мышления. Одна из отраслей логики, развивающаяся применительно к потребностям математики, называется математической логикой, а одним из ее разделов является алгебра логики или Булева алгебра.

Начало разработке логического исчисления, позволяющего оперировать логическими суждениями так же, как алгебраическими символами в элементарной математике, положил Буль.

Исчисление высказываний — это первое и наиболее широко применяемое в математической логике понятие.

Высказывание - это всякое предложение, которое может быть либо истинным, либо ложным и не может иметь никакого третьего значения. Отдельные простые высказывания могут быть связаны при помощи различных логических связей в сложные, которые также могут быть либо истинными, либо ложными. Используем двоичный код: истинное высказывание обозначим единицей (1), ложное — нулем (0). Благодаря этому на базе нескольких простейших электронных или иных

элементов представляется возможным синтезировать любые сложные логические высказывания.

Рассмотрим основные связи между простыми высказываниями. Предложение «Коля пойдет гулять, если будет тепло и сухо» может быть разбито на два простых высказывания или предложения, связанные союзом *И*: «Коля пойдет гулять, если будет тепло» и «Коля пойдет гулять, если будет сухо».

Здесь возможны четыре варианта: не будет тепло (0) и не будет сухо (0), т. е. оба составляющие высказывания оказываются ложными. Очевидно, что в этом случае Коля гулять не пойдет, и все предложение в целом оказывается ложным. Во-вторых, возможно, что будет тепло (1), но не будет сухо (0), т. е. первое высказывание оказывается истинным, а второе, ложным. И в этом случае Коля не пойдет гулять и сложное предложение окажется ложным. В третьем варианте будет холодно (не тепло), но будет сухо (1); здесь первое высказывание ложно, а второе истинно и сложное высказывание опять-таки ложно.

Сложное высказывание, объединяющее союзом *И* два и более простых высказываний, будет истинным только тогда, когда истинны все составляющие его высказывания, и будет ложным во всех остальных случаях. В нашем примере все предложение будет истинным, когда будет тепло и сухо. Это и является четвертым возможным вариантом, когда Коля действительно пойдет гулять.

Обозначим сложное высказывание — возможность прогулки Коли — через *P*, а составляющие его простые высказывания «Коля пойдет гулять, если будет тепло» — *a* и «Коля пойдет гулять, если будет сухо» — *b*. Тогда сложное высказывание можно записать в виде формулы

$$P = ab,$$

которая обычно называется логическим умножением, или логической функцией *И*.

Кроме союза *И*, для объединения простых высказываний употребляется также союз *ИЛИ*. Например, предложение «Вечером Петя пьет чай или ужинает» будет истинным в трех случаях: когда Петя пьет чай, но не ужинает; когда Петя ужинает, но не пьет чай, и, наконец, когда он и ужинает, и пьет чай. Следовательно, сложное предложение, объединяющее союзом *ИЛИ* два или более простых высказывания, будет ложным только тогда, когда ложны все составляющие его высказывания, и истинным во всех остальных случаях.

Эта логическая связь — логическое сложение — может быть записана в виде формулы

$$P = a + b.$$

Третья логическая функция носит название *НЕ* — отрицание. Например, мы имеем простое высказывание «Дождь идет». Образует из него сложное при помощи отрицания — «Дождь *НЕ* идет». Если простое высказывание истинно, т. е. действительно идет дождь, то сложное высказывание будет ложным. Если же простое высказывание ложно, т. е. дождь не идет, то выражение «Дождь *НЕ* идет» будет истиной. Эта функция имеет вид

$$P = \bar{A}$$

и читается как «*P* равно не *A*».

Основные законы алгебры логики

Алгебра логики подчиняется законам, иногда совпадающим с законами обычной алгебры, а иногда — своим своеобразным законам. Рассмотрим основные законы алгебры логики.

Законы множеств:

$$\begin{aligned}0 \cdot a &= 0; \\ 0 + a &= a; \\ 0 \cdot abc \dots w &= 0,\end{aligned}$$

т. е. произведение любого числа переменных обращается в нуль, если какая-либо одна переменная имеет значение 0, независимо от значений других переменных;

$$\begin{aligned}1 \cdot a &= a; \\ 1 + a &= 1; \\ 1 + a + b + c + d + \dots + w &= 1,\end{aligned}$$

т. е. сумма любого числа переменных обращается в единицу, если одна из ее переменных имеет значение 1, независимо от значений других переменных.

Законы перемещения:

$$\begin{aligned}ab &= ba; \\ a + b &= b + a,\end{aligned}$$

т. е. результаты выполнения операций умножения и сложения не зависят от того, в каком порядке следуют переменные.

Законы тавтологии (повторения):

$$\begin{aligned}aa &= a, \\ aaa \dots a &= a^n = a, \\ a + a &= a, \\ a + a + a + \dots + a &= na = a.\end{aligned}$$

Здесь можно сказать, что истина или ложь всегда остается истиной (или ложью), сколько ее не повторяй.

Законы дополнительности:

а) логическое противоречие:

$$a\bar{a} = 0$$

т. е. произведение любой переменной и ее инверсии есть 0. Как пример можно привести строку из известной песни «Речка движется и не движется» — заведомая ложь,

б) закон исключенного третьего:

$$a + \bar{a} = 1,$$

т. е. сумма любой переменной и ее инверсии есть 1. Так, например, утверждая, что «студент сдаст экзамен или не сдаст», мы всегда будем правы.

Законы инверсии (Де Моргана):

$$\overline{ab} = \bar{a} + \bar{b},$$

т. е. инверсия произведения равна сумме инверсий;

$$\overline{a + b} = \bar{a}\bar{b},$$

а инверсия суммы есть произведение инверсий.

Здесь записаны законы для двух переменных, но они справедливы для любого числа переменных.

Законы распределительные (дистрибутивные)}

а) произведения относительно суммы:

$$a(b + c) = ab + ac.$$

Справедливость этого закона можно подтвердить высказываниями. Например: «Я зайду за Вами *И* мы пойдем в театр *ИЛИ* в кино». Так можно формулировать левую часть приведенного выше выражения, а правая часть тогда может быть прочитана так: «Я зайду за Вами *И* мы пойдем в театр *ИЛИ* я зайду за Вами *И* мы пойдем в кино». Смысл один и тот же, но правая часть несколько длиннее:

б) суммы относительно произведения:

$$a + bc = (a + b)(a + c).$$

Справедливость этого закона можно доказать, опираясь на предыдущие. Раскрыв скобки, получим

$$a + bc = aa + ac + ba + be = a + ac + ba + bc.$$

Из первых двух членов вынесем за скобки переменную a :

$$a(1+c),$$

но

$$1 + c = 1, \text{ а } a \cdot 1 = a.$$

Рассматривая следующее выражение $ba + bc$, мы устанавливаем, что и оно равно a . Тогда вся правая часть превращается в $a + bc$, т. е. в такое же выражение, как и в левой части.

Законы склеивания:

$$ab + a\bar{b} = a;$$

$$(a + b)(a + \bar{b}) = a.$$

Эти законы легко подтверждаются на основании рассмотренных ранее законов, например:

$$ab + a\bar{b} = a(b + \bar{b}) = a \cdot 1 = a.$$

Законы поглощения:

$$a(a + b) = a;$$

$$a(a + b)(a + c) \dots (a + w) = a;$$

$$a + ab = a;$$

$$a + ab + ac + \dots + aw = a;$$

$$a(\bar{a} + b) = ab;$$

$$a + \bar{a}b = a + b.$$

Эти законы можно легко доказать с помощью других законов алгебры логики, например, умножая в последнем выражении первый член a на $(1 + b)$, получаем

$$a(1 + b) + \bar{a}b = a + ab + \bar{a}b = a + b(a + \bar{a}) = a + b \cdot 1 = a + b.$$

Так как для логического сложения и умножения характерны все свойства сложения и умножения алгебры чисел, то над многочленами алгебры высказываний можно производить те же действия, что и над многочленами алгебры чисел. Но логическое сложение и умножение обладают и некоторыми необычными свойствами и это приводит к необычности действий над логическими многочленами. Разъясним это на примере.

Пример. Пусть необходимо умножить $(a + b)$ на $(a + c)$. Умножаем по обычным правилам умножения многочлена на многочлен:

$$(a + b)(a + c) = aa + ac + ab + bc.$$

Так как в алгебре высказываний $aa = a$, то

$$(a + b)(a + c) = a + ac + ab + bc.$$

Но работу над полученным произведением можно продолжить. Рассмотрим два первых слагаемых a и ac . Сгруппируем их и общий множитель a вынесем за скобки:

$$a + ac = a(1 + c),$$

и далее

$$a \cdot 1 = a.$$

Также поступим и с суммой $a + ab = a$. Тогда окончательно

$$(a + b)(a + c) = a + bc.$$

Результат оказался проще, чем мы ожидали, так как выражение $a + ab$, согласно закону поглощения, заменили множителем a .

Таким образом, если высказывание логически складывают с логическим произведением, в состав которого оно входит, то оно поглощает это произведение. Отметим еще одну важную особенность алгебры высказываний. Если в формуле $a + ab$ заменить знак $+$ на знак

$\times$ и знак $\times$ на знак $+$, то полученное новое высказывание $a(a + b)$ будет эквивалентно заданному. В этом легко убедиться, раскрыв скобки.

Это свойство распространяется на любую формулу алгебры высказываний.

Например,

$$a(b + c) = ab + ac.$$

К обеим частям применим упомянутую замену знаков и получим:

$$a + bc = (a + b)(a + c).$$

Еще пример. Дано $ab + a\bar{b}$. В левой части заменим знаки:

$$ab + a\bar{b} = (a + b)(a + \bar{b}) = a$$

или

$$(a + b)(a + \bar{b}) = a.$$

Эту особенность преобразования формул в алгебре высказываний называют законом двойственности. Опираясь на закон двойственности, легко преобразовать эквивалентные высказывания.

Упрощение логических выражений

Следует иметь в виду, что каждое логическое высказывание можно воплотить с помощью логических элементов в конкретный действующий автоматический механизм. Для этого каждое логическое сложение, т. е. знак плюс в формуле высказывания, следует осуществить логическим элементом *ИЛИ*, каждое логическое умножение — элементом *И*, а каждое отрицание или инверсию — элементом *НЕ* (рис. 5.1).

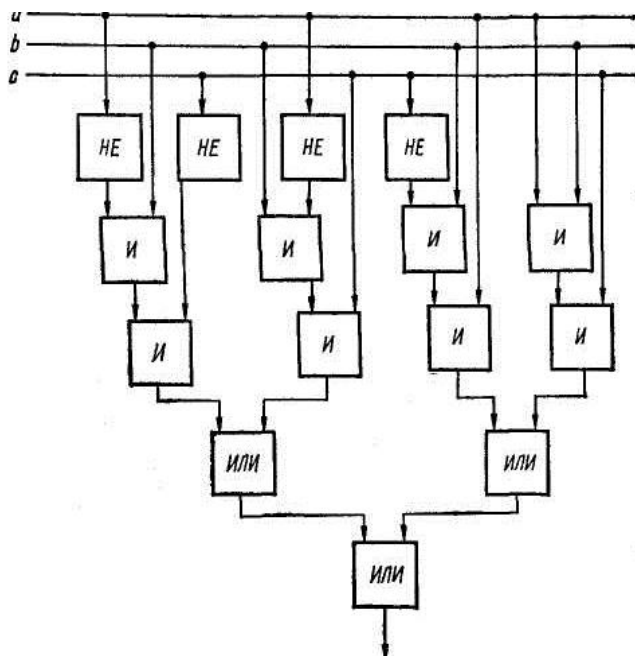


Рис. 5.1. Логическая схема.

Рассмотрим выражение

$$x = abc + \bar{a}bc + ab\bar{c} + \bar{a}b\bar{c}.$$

Заметим, что первое и третье, второе и четвертое слагаемые склеиваются по букве a .

Действительно,

$$abc + \bar{a}bc = bc$$

и

$$ab\bar{c} + \bar{a}b\bar{c} = b\bar{c}.$$

Имеем

$$x = bc + b\bar{c}.$$

В этой формуле можно произвести склеивание по букве c и тогда окончательно записанное ранее сложное выражение принимает очень простой вид: $x = b$.

Значит, математическая обработка выражения, построенного по законам математической логики, устранила необходимость в 15 логических элементах.

Мы произвели упрощение логического выражения, используя основные законы математической логики, путем последовательных рассуждений. Пример, который мы рассмотрели, относительно прост, да и то разные люди могли бы его решать по-разному и получать разные результаты. Тем более, такое явление может иметь место при упрощении более сложных логических выражений, в которых участвует большое число переменных, и которые выражаются более сложными зависимостями.

Задача упрощения логических выражений, или, как говорят, их «минимизация», является одной из наиболее сложных в алгебре логики. Есть много различных способов минимизации.

Рассмотрим один из распространенных способов минимизации логических выражений с помощью карт Карно.

Карты Карно (Karnaugh) наглядно изображают логические функции. Карта Карно (рис. 5.2) разделена на квадратики, и каждому из них отвечает определенная комбинация значений всех входных переменных. Кроме того, каждая сторона квадрата представляет собой границу между значениями переменных (верхний и нижний, равно как и боковые квадратики карты, являются соседними).

Обозначения входных переменных указываются сверху и сбоку карты и относятся ко всему столбику или строке квадратиков, причем значения этих входных переменных в них принимаются равными единице. В соседних с обозначенными в столбцах или строках входные переменные соответственно равны нулю. В квадратах записывается значение самой функции при данных комбинациях значений входных переменных. Значения входных переменных не принято записывать в квадратах, они подразумеваются, поэтому на карте остается только значение функции (рис. 5.2, *в, г, д, е*). Из примеров, приведенных на рисунке для двух, трех и четырех переменных, видно, что прибавление каждой новой переменной удваивает карту.

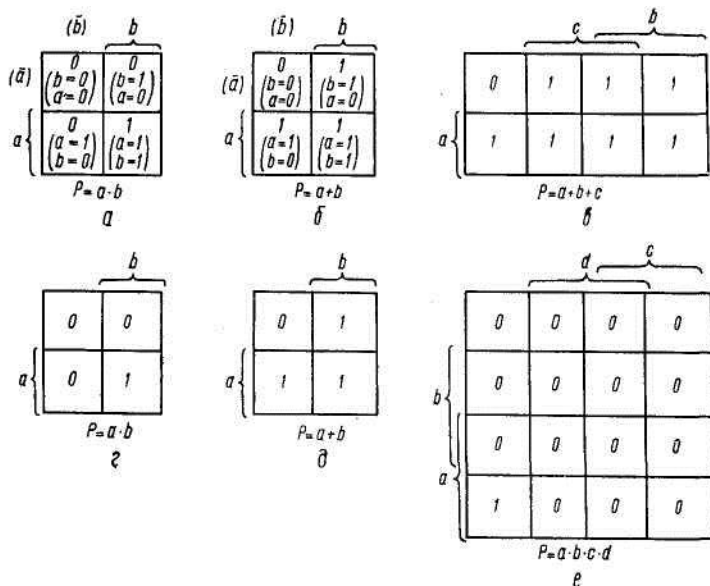


Рис. 5.2. Карты Карно.

Применяя карты Карно для изображения алгебраического выражения функций, можно записать функции либо в виде суммы произведений, либо в виде произведения сумм.

Выражение суммы произведений определяется суммой произведений значений всех входных переменных (прямых и инверсных) в каждом из квадратиков карты, содержащих единицу. Так, например, для карты, показанной на рис. 5.2, а, г, $P = ab$, а для карты, показанной на рис. 5.2, б, д, $P = ab + ab + \bar{a}b$.

Выражения сомножителей в произведении сумм определяются суммами инверсных значений входных переменных в каждом из нулевых квадратиков. Так, например, для карты, представленной на рис. 5.2, а, в,

$$P = (a + b)(a + \bar{b})(\bar{a} + b);$$

для карты, показанной на рис. 3.3, б, д,

$$P = a + b.$$

С помощью карт Карно можно получить упрощенное выражение функций, для чего определяют суммы произведений и произведения сумм, объединяя квадратики, в которых значения функции соответственно равны 1 или 0, в контуры. Последние должны иметь

форму прямоугольников и содержать четное число квадратилов или только один квадратик.

Из свойств карт Карно вытекает, что при переходе контура из одного квадратила к другому одна из переменных инвертируется. Поэтому выражение контура из двух квадратилов не зависит от этой переменной, а определяется только остальными переменными, т. е. выражения, соответствующие контурам, «не содержат тех переменных, чьи границы пересекаются данным контуром». Так, контур, ограничивающий четыре квадратила, пересекает две границы двух переменных и поэтому соответствующее ему выражение содержит $n-2$ переменных и т. д.

Для получения наиболее простых выражений, реализуемых минимальным количеством возможно более простых логических выражений, т. е. при минимизации, логическое выражение должно иметь как можно меньше членов, каждый из которых должен содержать как можно меньше переменных.

Правила минимизации выражения логической функции по карте Карно сводятся к следующему. Чем большее число квадратилов с одним значением функции объединяется в общем контуре на карте и чем меньше будет таких контуров, тем проще будет аналитическое выражение функции. При этом все квадратики с одним значением функции должны входить в какой-нибудь контур. Нужно также следить за тем, чтобы какой-либо контур не входил полностью в другие контуры.

Рассмотрим подробнее карту Карно для трех переменных (рис. 5.3).

Для наглядности в квадратилах указаны значения переменных. Пусть функция должна быть заложена в карту Карно и минимизирована.

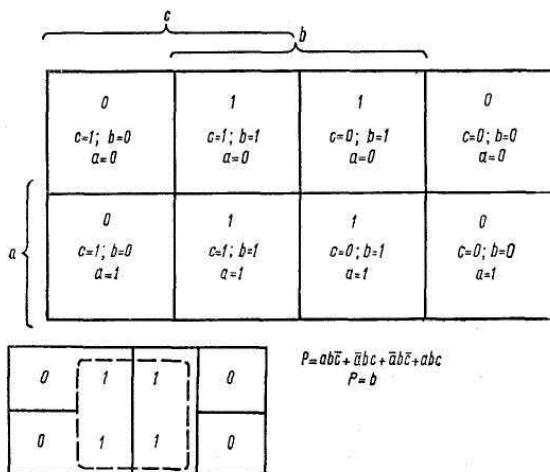


Рис. 5.3. Карта Карно для трех переменных.

Подставив значения переменных, соответствующие левому верхнему квадратику, в выражение функции, получим

$$X = abc + ab\bar{c} + \bar{a}bc + \bar{a}b\bar{c} = 001 + 000 + 101 + 100 = 0 + 0 + 0 + 0 = 0.$$

Это значит, что значение сложного выражения в этом квадратики равно 0, что и записываем.

Определяя таким образом значение выражения для всех квадратиков, получаем карту, показанную на рис. 3.3, б.

Охватываем контуром средние четыре квадратика со значениями 1. В этом контуре только лишь переменная сохраняет свое значение, равное единице во всех квадратах. Следовательно, результат минимизации определяется выражением

$$X = b.$$

Такое же значение мы получили ранее более сложным приемом минимизации — аналитическим путем с использованием основных законов алгебры логики,

Разница в затратах труда на минимизацию становится тем значительнее, чем больше переменных в логическом выражении и чем оно сложнее.

Схемы систем автоматизированного проектирования процессов основаны на так называемых двухпозиционных приборах, т. е. устройствах, способных занимать только одно из двух устойчивых положений. Сигнал, поступающий на вход в систему или снимаемый с выхода системы, может либо присутствовать (1), либо отсутствовать (0).

Поэтому в дальнейшем под переменными будем понимать сигналы на входе в схему, а под сложными высказываниями — сигналы на выходе, являющиеся логическими функциями этих переменных. Задача логической части схемы — выработать сигналы на выходе, являющиеся логическими функциями сигналов на входах.

Для решения задачи составления схемы необходимы электрические, электронные или пневматические устройства, осуществляющие элементарные логические связи *И*, *ИЛИ*, *НЕ*. Такие устройства называются элементами. Допустим, что наличие сигнала соответствует 1, т. е. истина, а отсутствию сигнала — 0, т. е. ложь. В электрических системах истине соответствует подача тока, а отсутствие тока — лжи. В пневматических системах наличие сигнала означает подачу сжатого воздуха под давлением, а отсутствие сигнала — соединение с атмосферой.

Логический элемент типа *И* должен иметь два или больше входа и один выход, с которого сигнал снимается.

В электрическом устройстве, в котором два реле *a* и *b* включены последовательно (рис. 5.4), сигнал на выходе появится при подаче напряжения на катушки обоих реле. В этом случае элемент выполнит логическую операцию умножения $P = ab$.

В пневматическом устройстве сигналы, т. е. сжатый воздух, поступают от двух пневматических кнопок *a* и *b*. Если нажать на одну из кнопок и подать воздух под давлением на один из входов, то две тарели клапана, сидящие на одной оси, передвинутся в одну сторону и свяжут выход клапана с атмосферой через вторую кнопку. Если нажать обе кнопки, то независимо от того, какое положение займут тарели, сжатый воздух пройдет на выход.

Элемент *ИЛИ* также имеет два входа и один выход. Если два реле *a* и *b* соединены параллельно, то сигнал на выходе появится, если подать напряжение на любое реле, и тогда элемент выполняет операцию логического сложения $P = a + b$.

В пневматическом варианте, если обе кнопки отпущены, выход элемента связан с атмосферой по крайней мере через одну из кнопок. Нажмем, например, кнопку *a*. Под действием давления сжатого воздуха шарик переместится вправо, прижмется к резиновому кольцу и не даст воздуху выходить в атмосферу через кнопку *b*, вследствие чего воздух поступит на выход клапана—элемента. Если отпустим кнопку *a* и нажмем кнопку *b*, то шарик переместится влево, не давая воздуху выходить в атмосферу через кнопку *a*, и на выходе тоже появится сигнал. Нажмем обе кнопки вместе, и, в каком бы ни был положении шарик, появится сигнал на выходе.

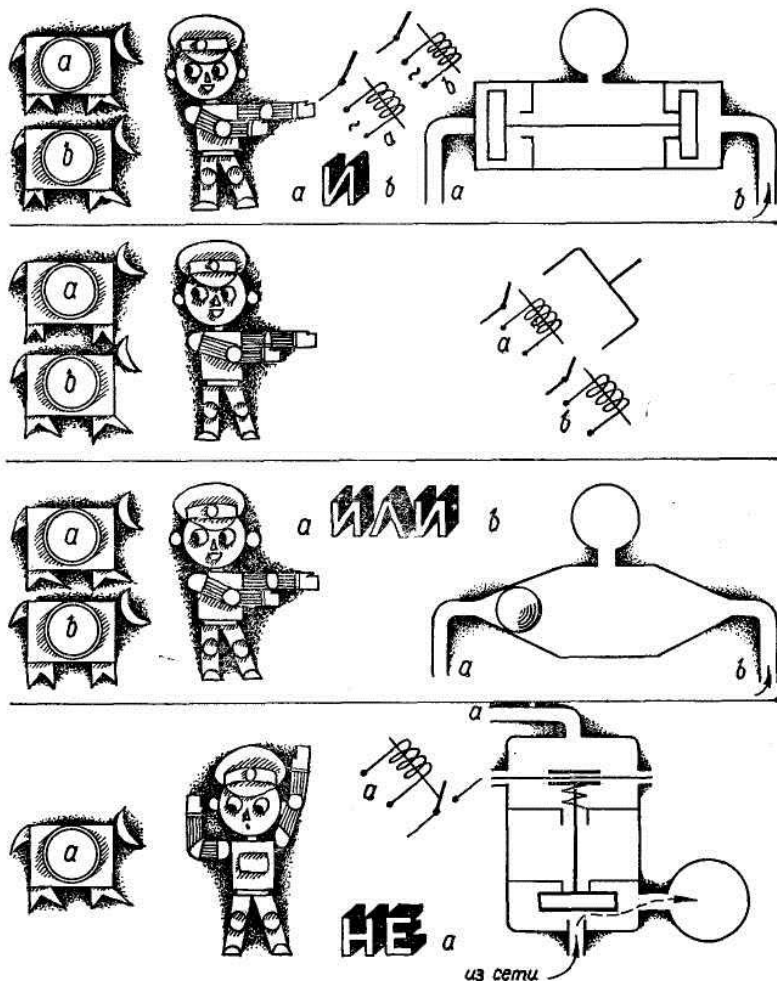


Рис. 5.4. Логические элементы И, ИЛИ, НЕ.

Логическим элементом *НЕ* в релейном варианте является переключатель. Когда напряжение в катушке *a* отсутствует, на выходе *P* протекает ток, т. е. имеется сигнал на выходе. В пневматическом варианте этот элемент выглядит несколько сложнее; оно носит название пневматического реле. Полость над мембраной 1, с которой соединяется пневматическая кнопка *a*, является полостью управления реле: она же представляет собой вход. Если кнопка *a* отпущена, т. е.

сигнал на входе отсутствует, то тарель 6 с толкателем 3 под действием пружины 2 находится в положении, показанном на рисунке. Резиновая прокладка 5 прижимается к верхнему седлу корпуса и закрывает проход к отверстию 4, ведущему в атмосферу. Сжатый воздух, подводимый к нижнему отверстию 7, свободно проходит на выход реле 8. При отсутствии сигнала на входе имеется сигнал на выходе.

Теперь нажмем кнопку *a*. Сжатый воздух начнет поступать в полость управления реле. Под действием силы давления воздуха мембрана прогибается, и тарель с толкателем движется вниз, сжимая пружину. Резиновая прокладка отходит от верхнего седла, освобождая проход к отверстию 4, а другая прокладка 6 перекрывает нижнее отверстие 7. Выход реле 8 оказывается связанным с атмосферой через отверстие 4. Имеется сигнал на входе — нет сигнала на выходе.

Логические элементы, из которых складываются логические системы, чаще всего основываются на применении электронных, пневматических, пневмонических схем. В обычных пневматических устройствах процессы совершаются во много раз медленнее, чем в электронных, и поэтому логические машины на пневматике работают значительно медленнее. Однако применение пневмоники позволяет создавать устройства на сжатом воздухе, способные выполнять до двух тысяч и более операций в секунду. Вместе с тем при решении целого ряда задач автоматизации производства и, очень часто, в машиностроении, большая скорость выполнения отдельных операций вообще и не требуется. Зато пневматические устройства имеют ряд качеств, выгодно отличающих их от электронных. Они по самой своей природе взрывобезопасны, просты и надежны. Для их обслуживания и ремонта не требуется высокой квалификации.

Пример использования языка исчисления высказываний

При решении задачи на основе языка исчисления высказываний, в САПП необходимо последовательно выполнить ряд следующих приемов:

- 1) составить подробные требования к формируемым процессам;
- 2) установить число входов и выходов процесса;
- 3) составить таблицу функционирования процесса по форме;
- 4) на основании таблицы составить структурную формулу процесса;
- 5) осуществить минимизирование логической функции, т. е. структурной формулы процесса;
- 6) составить функциональную схему процесса по минимизированной логической функции.

Когда функциональная схема составлена, можно считать, что задача по формированию процесса решения сформулированной проблемы будет решена.

Рассмотрим пример. На одном заводе имеются три цеха A , B и C . Электроэнергией их обеспечивает небольшая электрическая станция, на которой установлено два электрогенератора X и Y . Мощность генератора X в два раза выше, чем генератора Y .

Если в энергии нуждается один из цехов, то достаточно включить генератор Y , если же любые два цеха — генератор X . Снабжение электроэнергией всех трех цехов сразу обеспечивает одновременная работа двух генераторов. На электрической станции дежурный следит за сигналами из цехов A , B и C и соответственно регулирует включение того или иного генератора.

Стоит вопрос, нельзя ли сформировать техническое задание на создание автомата, который заменил бы дежурного по заводской электрической станции и, получая сигналы от цехов A , B , C , сам бы решал, какой из генераторов включать? Приведенное задание обычно рассматривают как словесное задание автомата, которое содержит лишнее высказывание о его работе.

Разработчик, ознакомившись с таким заданием, анализирует его с помощью следующих рассуждений: будущий автомат должен получать сигналы из трех цехов A , B и C , а это значит, что у него три входа. Сигналы, вырабатываемые автоматом, направляются в два адреса: на генератор X и на генератор Y . Значит, у него два выхода. Теперь можно составить таблицу работы автомата.

A	B	C	X	Y	A	B	C	X	Y
1	1	1	1	1	0	1	0	0	1
1	1	0	1	0	0	1	1	1	0
1	0	1	1	0	0	0	1	0	1
1	0	0	0	1	0	0	0	0	0

Если в энергии нуждаются три цеха, включены оба генератора, если два цеха — только генератор X или генератор Y .

Пользуясь составленной таблицей, следует составить структурную формулу автомата. Для этого следует брать те строки в таблице, в которых выход имеет значение, равное единице. В таблице таких строк четыре для выхода X и четыре для выхода Y . Составим формулу для выхода X .

На выходе X появится сигнал при поступлении сигналов от цехов A , B , C одновременно или от любых двух цехов одновременно — всего в четырех случаях. Теперь несложно составить структурную формулу,

по которой должен действовать дежурный или заменяющий его автомат:

$$X = abc + ab\bar{c} + a\bar{b}c + \bar{a}bc. \quad (C)$$

Аналогично составляют формулу автомата, вырабатывающего сигнал на выход Y; эта формула будет иметь вид

$$Y = abc + a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c. \quad (D)$$

Теперь следует минимизировать полученные выражения.

Составим карту Карно для формулы (C). Используя правила минимизации, получаем новую формулу для автомата, управляющего генератором X (рис. 5.5):

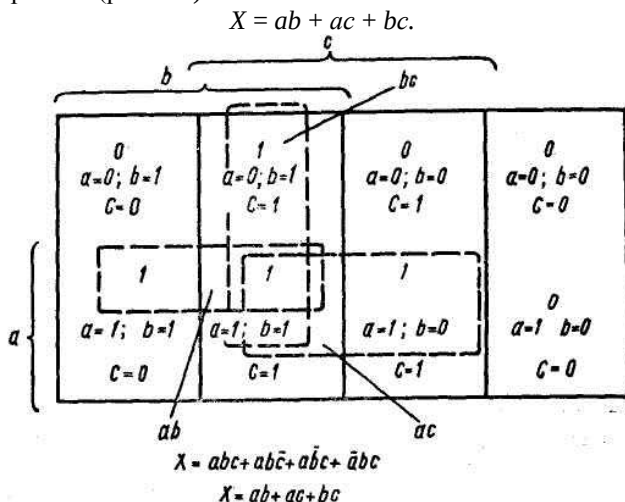


Рис. 5.5. Минимизация логического выражения для менее мощного генератора.

Наносим на карту Карно формулу (D). Так как единицы и нули на карте чередуются и нет ни одной пары смежных, которые можно было бы взять в контур, то выражение не поддается минимизации (рис. 5.6).

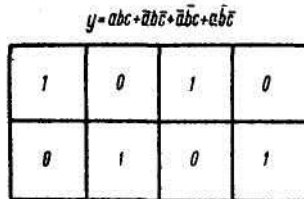


Рис. 5.6. Карта Карно для более мощного генератора.

Структурная формула автомата, управляющего включением в работу генератора Y , остается прежней:

$$Y = abc + a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c.$$

Функциональная схема автомата приведена на рис. 5.7.

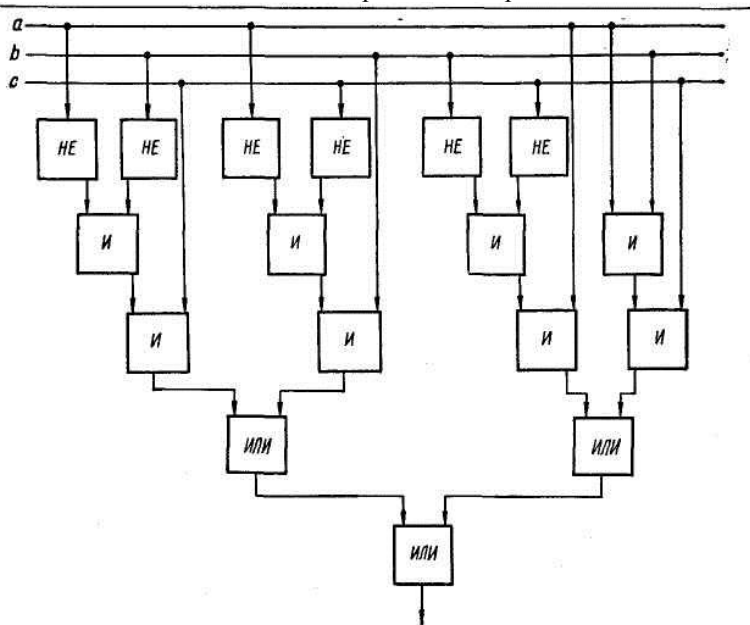


Рис. 5.7. Функциональная схема автомата, управляющего снабжением электроэнергией трех цехов

5.2.2. Исчисление предикатов как язык описания процессов

5.2.2.1. Введение в исчисление предикатов

Для осуществления автоматизации процессов необходим, как мы уже говорили, адекватный язык. Этот язык должен служить не только и не столько средством представления знаний (информации), сколько средством логического анализа задач.

Обычные человеческие языки, развивавшиеся под влиянием практических потребностей простоты общения (что далеко не всегда

совместимо с точностью и надежностью логического анализа!), для этой цели плохо подходят. По этой причине желательно, даже практически необходимо, использовать в качестве языка логического анализа процессов специально созданный формализованный язык. Такой язык в противоположность обычному языку должен следовать за логической формой и воспроизводить ее даже в ущерб краткости и легкости общения, если это будет необходимо. Главной отличительной чертой такого формализованного языка является наличие в нем особой системы логического вывода или дедукции.

В качестве такого формализованного языка, удовлетворяющего указанным требованиям, мы возьмем *исчисление предикатов*. В терминах исчисления предикатов можно сформулировать многие предложения и утверждения, выраженные на естественных языках, а также формализовать процесс рассуждений и доказательств, который является базовым процесса. Благодаря этому может быть устранен или во всяком случае резко снижен «языковый барьер» между ЭВМ и человеком.

Для того чтобы описать исчисление предикатов, мы должны воспользоваться какой-то частью обычного языка и в терминах этого языка образовать словарь и сформулировать правила формализованного языка, включая правила логического вывода. Исчисление предикатов (точнее, его синтаксис и семантика) определяется следующим очень экономным словарем символов и правилами их соединения и интерпретации:

1. Имена. Это — заимствованные из обычных языков выражения, служащие для непосредственного обозначения предмета. Примерами имен являются: «консультант», «мозг», «робот», «манипулятор», «процесс», «источник энергии», «искусственный интеллект» и т. д.

Следует отметить, что в одном или различных языках разные имена могут быть синонимами и выражать один и тот же смысл. С другой стороны, одно имя в различных языках или даже в одном языке (при омонимии) может выражать разный смысл.

Полное понимание языка требует знания смысла всех слов языка. **Естественно потребовать, чтобы каждое имя имело точно один смысл. Такая однозначность обеспечивается в логике предикатов.** А вот в обычных языках, как мы знаем, дело обстоит совсем не так.

2. Константы и переменные. *Константа* — это собственное имя. Примерами констант являются собственные имена чисел, людей, роботов, процессов, объектов.

Переменная — это символ, содержание которого совпадает с содержанием константы, за исключением лишь того, что единственный денотат константы заменен здесь возможностью

различных значений переменной. С каждой переменной связана некоторая непустая область ее возможных значений. Поэтому к содержанию переменной относится в некотором смысле и содержание собственного имени области ее значений. Нужно особо подчеркнуть, что переменная **в исчислении предикатов есть определенного рода символ, а не предмет (например, число)**, который этот символ обозначает.

3. Функции и термы. *Функция*— это **операция**, которая будучи применена к чему-то как к аргументу, дает некоторый объект в качестве значения функции для данного аргумента. В природе всякой функции лежит свойство быть применимой лишь к некоторым предметам.

Предметы, к которым функция применима, составляют область определения функции, а ее значения составляют область значений функции. Сама функция состоит в определении некоторого значения для каждого аргумента из области ее определения. Например, функция распознавания состоит в определении номера класса, к которому принадлежит объект, трактуемый как аргумент.

Для того чтобы обозначить значение функции для некоторого аргумента, обычно пишут имя этой функции и приписывают к нему справа имя аргумента, взятое в скобки. Так, если f — функция, а x принадлежит к области ее определения, то $f(x)$ есть значение функции f для аргумента x . Если функция применима к упорядоченной системе из n аргументов, то она называется n -арной.

Важную роль в дальнейшем играют выражения для функций, значения которых принадлежат той же области, что и их аргументы. Такие выражения называются термами. **Терм — это выражение, построенное, исходя из символов предметных переменных и констант, с помощью символов функций.** Например, если f есть n -арная функция и уже известно, что $x_1, \dots, x_n$ —термы, то $f(x_1, \dots, x_n)$ есть терм. Содержательно терму соответствует имя некоторого предмета.

4. Предложения, высказывания и предикаты. Простейшим выражением в обычных языках является предложение. **Предложение - это такое соединение слов, которое имеет самостоятельный смысл, т. е. выражает законченную мысль.** Каждому предложению сопоставим *высказывание* (выражаемое этим предложением), предполагая при этом, что каждое высказывание или истинно, или ложно и не может быть одновременной истинно и ложно. Таким образом, **высказывание можно рассматривать как величину, принимающую только два значения: «истина» (И) или «ложь» (Л).**

Предположим теперь, что x представляет собой произвольный предмет из некоторого множества $\{x\}$, а $F(x)$ — какое-либо высказывание о x . Выражение $F(x)$ становится определенным, когда переменная x заменена определенным значением (именем предмета) из множества $\{x\}$. Например, выражение « x есть животное» становится вполне определенным высказыванием, если x — это робот (ложное высказывание) или если x — это собака (истинное высказывание).

Так как с нашей точки зрения каждое определенное высказывание представляет собой И или Л, то выражение $F(x)$ означает, что каждому предмету из $\{x\}$ поставлен в соответствие один из двух символов: И или Л. Иначе говоря, $F(x)$ представляет собой функцию, определенную на множестве $\{x\}$ и принимающую только два значения: И и Л. Аналогично неопределенное высказывание о двух, трех и более предметах представляет функцию со значениями И и Л от двух, трех и более переменных. Эти неопределенные высказывания (функции одной или нескольких переменных) вида $F(x_1, \dots, x_n)$ мы будем называть **логическими функциями или предикатами**.

5. Элементарные (атомарные) и правильно построенные формулы. Какой бы ни был символ n -местного предиката и каков бы ни был выбор термов $x_1, \dots, x_n$, (не обязательно различных), выражение $F(x_1, \dots, x_n)$ мы будем называть **элементарной**, или **атомарной формулой**. Из этого определения следует, что, например, имена предметов не являются формулами.

Рассматривая элементарные формулы как величины, способные принимать только значения И и Л, мы определим над ними операции, которые позволяют из данных формул получать новые. Эти операции, по существу, выражают употребительные в обычных языках связи.

Если A и B — какие-либо данные формулы (т. е. либо элементарные формулы, либо уже построенные сложные формулы), то $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$ также являются (сложными) формулами. Если A — данная формула, то $\neg A$ — также (сложная) формула. Первые четыре операции — бинарные (двухместные), пятая — унарная (одноместная).

Символы $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\neg$ называются соответственно **конъюнкцией**, **дизъюнкцией**, **импликацией**, **эквивалентностью** и **отрицанием**. Их можно читать, пользуясь словами, приведенными в правой части следующей таблицы:

$\wedge$ — «и»;
 $\vee$ — «или», «... или, ... или», «и/или»;
 $\rightarrow$ — «влечет», «если..., то...», «только..., если...»;
 $\leftrightarrow$ — «равносильно», «эквивалентно», «тогда и только тогда»,
 $\neg$ — «не», «неверно, что».

Прочтение сложных формул может стать неоднозначным, если не ввести скобок, указывающих, в каком порядке формулы связываются между собой. Поэтому мы будем писать $(A \rightarrow B) \rightarrow C$ или $A \rightarrow (B \rightarrow C)$, а не $A \rightarrow B \rightarrow C$. Впрочем, число скобок можно уменьшить, приписав нашим связкам убывающие «ранги» в следующем «порядке старшинства»:

$$\leftrightarrow, \rightarrow, \vee, \wedge, \neg.$$

Там, где возможны были бы два способа построения формулы, связка более высокого ранга имеет большую область действия. Так, $A \rightarrow B \wedge C$ означает $A \rightarrow (B \wedge C)$. Связка $\neg$ имеет наименьший ранг, так что, например, $\neg A \vee B$ означает $(\neg A) \vee B$, а не $\neg (A \vee B)$.

При построении сложных формул возникает вопрос, как определить значения сложных формул, зная значения простых формул, которые их составляют? Ответ на этот вопрос дается нижеследующей таблицей истинности.

A	B	$\neg A$	$\neg B$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
И	И	Л	Л	И	И	И	И
Л	И	И	Л	Л	И	И	Л
И	Л	Л	И	Л	И	Л	Л
Л	Л	И	И	Л	Л	И	И

Таким образом, $A \leftrightarrow B$ истинно тогда и только тогда, когда A и B имеют одинаковые значения (почему $\leftrightarrow$ и называют «эквивалентностью»); $A \rightarrow B$ ложно тогда и только тогда, когда A истинно, а B ложно; $A \vee B$ истинно тогда и только тогда, когда и A , и B истинны; $A \wedge B$ ложно тогда и только тогда, когда и A , и B ложны; наконец, $\neg A$ истинно тогда и только тогда, когда A ложно.

Кроме пяти упомянутых символов-связок, в исчислении предикатов употребляются еще два символа, выражающие операции утверждения всеобщности и существования. Символ $\forall x$ называется *квантором всеобщности*, а символ $\exists x$ — *квантором существования*.

Формула $\forall x F(x)$ истинна, когда $F(x)$ истинно для каждого элемента x области $\{x\}$, и ложна в противном случае. Соответствующее ей словесное выражение будет: «для всякого x $F(x)$ истинно». Формула $\exists x F(x)$ истинна, если существует элемент области $\{x\}$, для которого $F(x)$ истинно, и ложна в противном случае. В обычном языке этой

формуле соответствует выражение: «существует x такое, что $F(x)$ истинно».

Мы будем говорить, что в формулах $\forall xF(x)$ и $\exists xF(x)$ переменная x связана соответствующим квантором. Ясно, что сами эти формулы от x не зависят. Заметим, что $\neg(\forall xF(x)) \leftrightarrow \exists x\neg F(x)$.

Теперь мы можем дать определение *правильно построенной формулы* (ППФ) на языке консультационного процесса. ППФ называется выражение, которое может быть построено исходя из элементарных (атомарных) формул с помощью операций перехода от формулы A к формулам $\forall xF(x)$ и $\exists xF(x)$, от формул A и B к формулам $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$, $\neg A$, $\neg B$. Элементарная формула или ее отрицание, входящие в ППФ, называются литерами (или литералами), а дизъюнкция литер называется простым дизъюнктом.

6. Интерпретации. ППФ имеет смысл только тогда, когда имеется какая-нибудь интерпретация входящих в нее символов. Под *интерпретацией* мы будем понимать всякую систему, состоящую из непустого множества D , называемого предметной областью (областью консультирования проблем), и какого-либо соответствия, относящего каждому символу n -местного предиката некоторое n -арное отношение в D , каждому символу функции от n аргументов некоторую n -местную операцию в D и каждой константе — некоторый элемент из D . Например, если D есть множество всех проблем, то отношение между двумя проблемами, состоящее в том, что первая из них «проблемнее» (по каким-то определенным параметрам) второй, можно отождествлять с множеством всех упорядоченных пар проблем (x, y) таких, что x проблемнее y . Таким образом, интерпретация осуществляет связь между языком консультационного процесса и описываемой им предметной областью (консультируемой проблемой) реального мира. Она позволяет придать ППФ содержательный смысл.

При заданной интерпретации всякая ППФ (не содержащая свободных переменных) представляет собой высказывание, которое истинно или ложно. Если при данной интерпретации каждая из ППФ A_i , $i = 1, \dots, n$, имеет значение И, то будем говорить, что данная интерпретация удовлетворяет системе ППФ $\{A_i\}_{i=1}^n$. ППФ A выводима (логически следует) из некоторой системы ППФ $\{A_i\}_{i=1}^n$, если каждая интерпретация, удовлетворяющая $\{A_i\}_{i=1}^n$, удовлетворяет также и A . Так, очевидно, что ППФ $\forall xF(x)$ выводима из системы ППФ $\{\forall x\neg R(x) \vee F(x), \neg xR(x)\}$.

Согласно теореме Гёделя, если некоторая интерпретация удовлетворяет заданной системе ППФ $\{A_i\}_{i=1}^n$, то она удовлетворяет и любой ППФ A , выводимой из этой системы. Умение

продемонстрировать, что, ППФ A выводима (логически следует) из системы ППФ $\{A_i\}_{i=1}^n$, когда это на самом деле так, играет важную роль при логическом анализе, и мы сосредоточим на нем свое внимание. Предположим, что A выводима из $\{A_i\}_{i=1}^n$. Тогда любая интерпретация, удовлетворяющая $\{A_i\}_{i=1}^n$, удовлетворяет A , но не удовлетворяет $\neg A$. Следовательно, никакая интерпретация не удовлетворяет объединению $\{A_i\}_{i=1}^n \vee \neg A$. Если некоторая система ППФ не удовлетворяется ни при какой интерпретации, то она называется неудовлетворимой. Так, если ППФ A выводима из $\{A_i\}_{i=1}^n$, то объединение $\{A_i\}_{i=1}^n \vee \neg A$ неудовлетворимо. И наоборот, если $\{A_i\}_{i=1}^n \vee \neg A$ неудовлетворимо, то ППФ A должна логически следовать из системы ППФ $\{A_i\}_{i=1}^n$. Именно эта концепция выводимости лежит в основе понятия логического вывода, или дедукции, в исчислении предикатов.

Универсальным методом логического вывода является так называемый *метод резолюций*, предложенный в 1965 г. Дж. Робинсоном. Этот метод замечателен тем, что он сложный процесс логического вывода сводит к последовательности очень простых операций, каждая из которых может быть легко запрограммирована. В основе метода резолюций лежат три простых правила вывода (*резольвенции*):

1) если истинны ППФ A и $\neg A \vee B$, то истинна ППФ B (правило *modus ponens*);

2) если истинна ППФ $A \vee A$, то истинна ППФ A (правило факторизации);

3) если истинна ППФ $A(x)$, то истинна ППФ $\forall y A(y)$.

Эти правила применяются к простым дизъюнктам, на которые предварительно «раскладывается» система ППФ $\{A_i\}_{i=1}^n \vee \neg A$, из неудовлетворимости которой следует, что A выводима из $\{A_i\}_{i=1}^n$. Новые дизъюнкты, получаемые в результате применения указанных правил, называются *резольвентами*. При образовании резольвент существенную роль играет процедура *унификации*, которая для двух данных предикатов осуществляет подстановку термов вместо переменных, делающую предикаты одинаковыми. После этого к полученным ППФ применяются правила резольвенции. Например, для неудовлетворимой системы ППФ вида $\{A(x) \vee B(x), \neg B(f(z)), \neg A(f(z))\}$, используя первое правило вывода после подстановки терма $f(z)$ вместо переменной x , получим из первых двух ППФ резольвенту $A(f(z))$, которая в сочетании с третьей ППФ системы дает нулевую формулу. Таким образом, если выбрано два простых дизъюнкта и по одной литере в каждом из них, то применение правила унификации и затем правил вывода дает резольвенту. При доказательстве выводимости

ППФ A , рассматриваемой как заключение (теорема), из заданной системы ППФ $\{A_i\}_{i=1}^n$, рассматриваемых как посылки (аксиомы), процесс образования резольвент (в котором могут принимать участие и ранее полученные резольвенты) продолжается, пока не будет получена пустая формула, означающая неудовлетворимость системы $\{A_i\}_{i=1}^n \vee \neg A$ и успех доказательства. Важно отметить, что число резольвент, формируемых при доказательстве любой теоремы из заданной конечной системы аксиом, конечно.

В ряде задач, которые должны решаться с использованием автоматизированных процессов, простое доказательство выводимости ППФ A , формулирующей задание, из системы ППФ $\{\{A_i\}_{i=1}^n$, описывающих условия выполнения этого задания, оказывается недостаточным. Примером такой задачи является задача планирования поведения робота. В подобного рода задачах нужно знать то значение переменной x , при котором данная ППФ $A(x)$ логически выводима из некоторой системы ППФ $\{A_i\}_{i=1}^n$. Иными словами, пользователи (вместе с роботом) хотели бы знать, следует ли логически ППФ $\exists x A(x)$, и если да, то каково то значение x , при котором существует решение. Заметим, что умение отыскивать такие значения для переменной, связанной квантором существования, позволяет ставить роботу вопросы весьма общего характера и осуществлять диалог с ним. Например, можно спросить у робота: «Какие действия и в какой последовательности нужно совершать, чтобы собрать из деталей определенную конструкцию?». Ответом на этот вопрос будет не просто констатация факта, что сборка данной конструкции возможна, а развернутый план действий (технологический маршрут) сборки.

Рассмотрим на простейшем примере, как можно решать подобного рода задачи. Пусть роботу известно, что его манипулятор жестко закреплен на подвижной платформе, а платформа находится в цехе. Можно спросить: «где находится манипулятор?». В этой задаче сформулированы два «факта», которые можно записать в виде двух правильно построенных формул:

$A_1 \leftrightarrow \forall x P(\text{платформа}, x) \rightarrow P(\text{манипулятор}, x)$, $A_2 \leftrightarrow P(\text{платформа}, \text{цех})$, где двухместному предикату $P(y, z)$ придана очевидная интерпретация: « y находится в z ». На вопрос «где находится манипулятор?» робот может дать ответ, если сначала докажет, что правильно построенная формула

$A_1 \leftrightarrow \forall x P(\text{манипулятор}, x)$

выводима из системы ППФ $\{A_i\}_{i=1}^n$, и затем найдет то значение x (константу), которое на самом деле «существует» и служит ответом.

Используя описанный выше метод резолюций, робот сначала попытается доказать неудовлетворимость системы $\{A_i\}_{i=1}^n \vee \neg P$

(манипулятор, x). (Заметим, что отрицание ППФ A есть ППФ $\neg P$ (манипулятор, x)). Процесс доказательства неудовлетворимости $\{A_i\}_{i=1}^n \vee \neg A$ представлен на дереве вывода, изображенном на рис. 3.8.

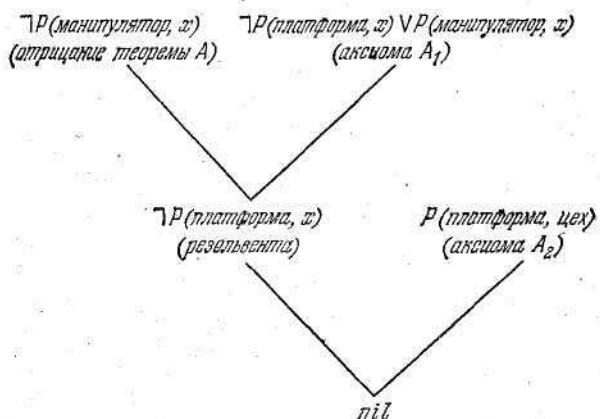


Рис. 5.8. Дерево вывода.

Из этого дерева вывода можно извлечь также ответ на вопрос: «где находится манипулятор?». Это осуществляется следующим образом. Сначала к отрицанию теоремы добавляется ее отрицание, т. е. сама теорема. В результате получается тавтология (т.е. ППФ, тождественно истинная при всех интерпретациях) вида

$$\neg P(\text{манипулятор}, x) \vee P(\text{манипулятор}, x).$$

Затем в соответствии со структурой дерева вывода, изображенного на рис. 5.8, вновь формируются резольвенты до тех пор, пока в корне дерева не получится некоторая ППФ, играющая роль ответа на языке робота. В нашем примере получим одну резольвенту

$$\neg P(\text{платформа}, x) \vee P(\text{манипулятор}, x),$$

а в корне дерева — ППФ $P(\text{манипулятор}, \text{цех})$, в которой содержится ответ на вопрос «где находится манипулятор?». Заметим, что форма ответа на языке предикатов близка к форме теоремы-вопроса. В нашем случае единственное отличие состоит в том, что в теореме-вопросе содержится переменная, связанная квантором существования, а в ответной ППФ — константа (ответный терм).

Таким образом, описанная система логического вывода, основанная на методе резолюций, представляет собой эффективное средство для автоматического поиска доказательств (отыскания логических следствий) и извлечения ответа в терминах исчисления предикатов. Мы рассмотрели основные понятия этого исчисления и связанного с

ним метода резолюций не ради них самих, а чтобы понять и продемонстрировать, как человек, используя этот язык, может логически рассуждать, обучаться новому и адаптироваться в процессе решения задач в системе автоматизированного проектирования.

5.2.2.2. Адаптация в процессе логического вывода

Первым и неизбежным этапом применения описанной выше системы логического вывода для решения задач, требующих логического анализа, является формулировка этих задач в терминах исчисления предикатов. Для этого нужно, прежде всего, задать предметную область, т. е. совокупность относящихся к решаемой задаче объектов (или процессов), и выделить их существенные свойства, от которых в наибольшей степени зависит успех решения. Далее нужно, присвоив определенный содержательный (семантический) смысл предикатным и функциональным символам, формализовать данные и условия задачи в виде ППФ, которые должны на них выполняться (т. е. истинность которых считается не требующей доказательства). Очевидно, что эти ППФ выделяют из всевозможных систем объектов, их свойств и отношений между ними такие системы, для которых они выполнены.

ППФ, посредством которых мы таким образом выделяем совокупность объектов, называются **аксиомами**. Если для какой-либо совокупности объектов, их свойств и отношений некоторые аксиомы истинны, то говорят, что данная совокупность объектов удовлетворяет системе этих аксиом или является интерпретацией данной системы аксиом.

Таким образом, **аксиомы можно рассматривать как определения системы объектов, их свойств и отношений между ними**. Делая логические выводы из аксиом, мы будем получать ППФ, истинные для любой системы объектов, удовлетворяющей данным аксиомам.

Ясно, что соответствие между аксиомами и предметами реальности, т. е. предметной областью, всегда имеет приближенный характер. Поэтому возникает вопрос, как узнать, действительно ли данная система аксиом определяет именно то, что было задумано, что требуется для решения задачи?

Ответ на этот вопрос связан с понятием *непротиворечивости системы аксиом*. Мы должны быть уверены, что делая всевозможные выводы из данной системы аксиом, не приходим к противоречию, т. е. не выведем какие-либо несовместимые ППФ. Появление противоречия означало бы, что рассматриваемой системе аксиом не может удовлетворять никакая совокупность объектов, и, таким образом, эти аксиомы ничего не описывают. Мы будем говорить, что система аксиом $\{A_i\}_{i=1}^n$ противоречива, если в ней выводима какая-либо ППФ A , а также и ее отрицание $\neg A$. Для проверки (доказательства)

непротиворечивости системы аксиом достаточно построить какую-нибудь точную интерпретацию этой системы.

Весьма важным является свойство *независимости аксиом*. Какая-либо аксиома A называется *независимой* в данной системе аксиом $\{A_i\}_{i=1}^n$, если она невыводима из остальных аксиом этой системы. Для проверки (доказательства) независимости какой-либо аксиомы достаточно найти совокупность объектов, удовлетворяющую всем аксиомам, кроме исследуемой, и не удовлетворяющей этой последней. Иными словами, для проверки независимости аксиомы A достаточно найти интерпретацию следующей системы аксиом: $\{A_i\}_{i=1}^n, \neg A$.

Таким образом, система аксиом, которой пользуется робот, должна иметь точную интерпретацию в том мире объектов, свойств и отношений, в котором он функционирует. Этому требованию можно удовлетворить путем тривиальной формулировки тех задач, которые робот должен решать. Остановимся на этом вопросе подробнее.

Формулировка задачи на языке предикатов — это первый и наиболее ответственный этап организации его целенаправленного поведения. На этом этапе от «формулировщика» задачи требуются глубокие знания не только и не столько исчисления предикатов, сколько существа решаемой задачи, ее специфических черт, консультируемой проблемы, той цели, которая должна быть достигнута в результате реализации процесса решения задачи. Возможна, но совершенно бессмысленна постановка на языке предикатов таких, например, задач: «рекомендую переместиться туда, сам не знаю куда», или «рекомендую найти то, сам не знаю что».

Практически весьма важно, чтобы формулировка задачи (связанная с заданием системы аксиом и теорем-заданий) была по возможности простой, не «засоренной» массой мелких, второстепенных факторов, так как учет их существенно осложняет логический анализ и делает трудно обозримыми результаты решения. Отметим две типичные трудности, которые всегда подстерегают «формулировщика» задачи. Первая — это возможность «утонуть в деталях и подробностях», т. е. «из-за деревьев не увидеть леса»; вторая — слишком огрубить задачу, или, как принято говорить в подобных случаях, «вместе с водой выплеснуть и ребенка». Ниже на примерах формулировки задач по формированию планирования поведения робота и распознавания сложных ситуаций мы увидим, что искусство формулировки задач на языке предикатов есть именно искусство. Здесь нет общих рецептов, а опыт «формулировщика» задач в этом трудном деле приобретаетс постепенно.

При построении аксиом будем различать два типа предикатов, использование которых по-разному сказывается на скорости решения

задачи. Предикаты первого типа описывают простейшие свойства конкретных объектов (например, «робот находится в точке x », «объект z большой» и т. п.). Предикаты второго типа определяют общую картину отношений между различными объектами (процессами) и их свойствами. Один такой предикат может описывать набор свойств большого числа объектов (процессов) (например, «если между точками a и b нет препятствий, то робот может проехать между этими точками по прямой» и т. п.). Количество подобных предикатов, необходимое для описания (с требуемой степенью подробности) данных и условий задачи, обычно невелико. Однако для сложных предикатов существенно возрастает сложность термов, участвующих в их определении.

Многие задачи часто связаны с изменением во времени свойств, которые обычно известны в начальный момент времени. В таких задачах удобно ввести предикат позиции, определяющий все «интересные» свойства всех объектов. При этом аксиомы, описывающие изменение предиката позиции во времени, в наиболее простой форме могут быть составлены из **трех литер**, а именно, если имеется некоторая позиция (ситуация, проблема) и если выполняется некоторое дополнительное условие, характеризующее принципиальную возможность применения данной аксиомы, то получится новая позиция (ситуация, проблема). Основным преимуществом такого способа построения аксиом является то, что свойства, связанные между собой, определяются одним предикатом и поэтому меняются одновременно. При этом на каждом шаге решения задачи учитывается все многообразие сложившейся ситуации, вследствие чего уменьшается число резольвент в процессе логического вывода.

Из дальнейшего изложения (и, в частности, из примеров) будет ясно, что эффективность системы логического вывода можно увеличить путем уменьшения числа предикатов и аксиом, определяющих данные и условия задачи. С этой целью разумно использовать ранее доказанные теоремы или ввести более сложные предикаты, образующие новые аксиомы, которые можно рассматривать как результат обучения постановщика задач в процессе решения задачи. Такие аксиомы, описывающие на языке исчисления предикатов приобретаемый постановщиком задач опыт, мы будем называть *аксиомами обучения*. Введение аксиом-обучения как бы моделирует феномен мышления, о котором еще Р. Декарт писал в своем «Рассуждении о методе»: «Каждая решенная мною задача становилась образцом, который служил впоследствии для решения других задач». Образно говоря, **аксиомы обучения играют роль лемм при доказательстве новых теорем, определяющих целевые условия**

задачи. Тем самым они позволяют оперировать более крупными «блоками» (фрагментами) доказательств, освобождая от рассмотрения многочисленных деталей, имеющих в данном доказательстве лишь вспомогательное значение. Заметим, что введение аксиом обучения позволяет постановщику задач увеличивать и улучшать знания о решаемом классе задач в процессе их непосредственного решения. Таким образом, **аксиомы обучения являются средством обучения новым понятиям и фактам и уточнения старых.**

Скорее всего, в будущем постановщикам задач так и не удастся прийти ни к какой определенной конечной системе аксиом, рассматриваемой как окончательная. Напротив, подобно тому, как это происходит в мире живого, будут появляться (автоматически формироваться) все новые аксиомы обучения, отображающие изменения в окружающем постановщика задач мире и в решаемом им классе задач.

Построение системы аксиом в каждой задаче важно не само по себе, а имеет целью выявление оптимальных путей логического вывода. Под *эффективностью системы логического вывода* мы будем понимать меру успешности и поиска решения. Для того чтобы выбрать количественный показатель эффективности решения, нужно прежде всего спросить себя: чего мы хотим от системы логического вывода, к чему стремимся при решении задачи? Выбирая (формируя) задачу, мы предпочитаем такую, которая при ее реализации обращает показатель эффективности в максимум или же в минимум.

Очень часто в качестве показателя эффективности систем логического вывода фигурируют затраты на формирование доказательства, которые, естественно, нужно минимизировать. Заметим, что неправильный выбор показателя эффективности очень опасен, так как он может привести к плохим решениям. Решения, выбранные под углом зрения неудачно выбранного показателя эффективности, могут привести к большим неоправданным потерям и затратам.

В рассматриваемом круге задач под *эффективностью* системы логического вывода будем понимать число шагов доказательства (возможно, усредненное по классу решаемых задач), т. е. число резольвент, формируемых в процессе поиска решения. С целью увеличения эффективности системы логического вывода введем некоторые ограничения на процесс образования резольвент, связанные с выбором стратегии формирования решения.

Стратегией логического вывода называется способ выбора очередной пары дизъюнктов и литер в них для образования резольвент. Именно стратегия определяет, в каком порядке будут образовываться

резольвенты и, следовательно, насколько быстро будет найдено решение. Стратегия «запускает» процесс доказательства — *начинается дедукция*: с помощью аксиом, резольвент и теорем строится та или иная конструкция доказательства. При этом стратегия решает, какие понятия и факты (аксиомы и литеры в них) несущественны, а какие — необходимы для доказательства. Таким образом, выбор и подстройка стратегии являются основным средством увеличения эффективности системы логического вывода, определяющим быстроту сходимости реализуемого ею метода поиска доказательства. Если правила резольвенции есть правила дедуктивного вывода следствий, то стратегия — это та активная часть, способная к обучению и адаптации, которая имитирует способ «мышления», например, интеллектуального робота, уровень его познаний в логике, степень его интеллектуальности.

Образно говоря, *стратегия системы логического вывода — это идея формирования доказательства, исходя из заданной системы аксиом, в которой заключены все необходимые для решения задачи знания*. Если идея (стратегия) хороша, то решение будет сформировано быстро. Однако рассчитывать на хорошую стратегию постановщик задач может лишь тогда, когда в системе аксиом достаточно полно отражены не только необходимые знания о задаче, но и прошлый опыт решения задач в подобных проблемах. Хорошие стратегии имеют своим источником прошлый опыт и ранее приобретенные знания.

Стратегия называется *полной*, если она находит (в конечное число шагов) доказательство любой ППФ, выводимой из аксиом. Примерами полных стратегий являются стратегия опорного множества, стратегия предпочтения единичным элементом, а также тривиальная стратегия полного перебора. Все перечисленные стратегии характеризуются тем, что для них **предикат** является **«неразложимым»** **понятием**, а критерием выбора очередной пары дизъюнктов могут быть количество предикатов в дизъюнкте, порядок их расположения и т. п. Такие стратегии, не зависящие от внутренней структуры, смысла используемых предикатов, будем называть *синтаксическими*.

Стратегию будем называть *адаптивной*, если она целенаправленно меняется (подстраивается) в процессе логического вывода в зависимости от приобретаемого опыта. Примерами адаптивных стратегий могут служить *семантические* стратегии, в которых критерий выбора очередных дизъюнктов зависит от вхождения в них определенного термина. *Согласно семантической стратегии сначала выбираются термы, соответствующие «интересным» проблемам, затем — предикаты, описывающие их свойства, и, наконец, ППФ, содержащие эти свойства*.

В предыдущем подразделе мы отмечали, что процессу поиска доказательства может быть поставлено в соответствие дерево вывода, которое заканчивается пустым дизъюнктом, означающим конец и успех доказательства. Дерево вывода строится и ветвится под каждой резольвентой так же, как и под теоремой (точнее, ее отрицанием). Отсюда ясно, что резольвенты, не приводящие к пустому дизъюнкту, резко увеличивают число шагов доказательства, и их получение крайне нежелательно. В связи с этим задача построения адаптивной стратегии может быть переформулирована как задача отсечения ненужных (лишних) ветвей на дереве вывода. Для решения этой задачи необходимо указать критерий предпочтения ветвей.

Действительно, в процессе доказательства теоремы можно указать, как правило, несколько подходящих аксиом и несколько путей (ветвей) доказательства. Если нет критерия предпочтения одной ветви другой, приходится действовать по методу случайного поиска, что соответствует образованию пучка ветвей на дереве вывода. Введение подходящего критерия предпочтения позволяет исключить лишние тупиковые ветви и благодаря этому существенно увеличить эффективность системы логического вывода. При этом особую роль играют критерии предпочтения, формируемые в процессе решения задач. **Примером такого критерия является критерий предпочтения аксиом обучения (хранящихся в памяти наряду с исходной системой аксиом), которые позволяют уменьшить исходную неопределенность относительно условий решения задачи.**

Введение аксиом обучения, о которых шла речь выше, особенно эффективно в тех случаях, когда в них либо раскрывается неопределенность (т. е. содержится новая информация), либо «запоминается» в компактной форме часто встречающийся в рассматриваемом классе задач «фрагмент» доказательства. В самом деле, если в процессе формирования доказательства потребуется доказать уже доказанную ранее теорему, то критерий предпочтения аксиом обучения сократит общее число шагов доказательства по крайней мере на длину доказательства соответствующей аксиом обучения.

Важной особенностью адаптивной системы логического вывода является ее способность логически рассуждать, т. е. сводить сложное заключение к последовательности утверждений, истинность каждого из которых проверяется очень просто и чисто механически. Такая система может также не только автоматически доказывать теоремы, трактуемые как некоторые задания или вопросы, но и обучаться способам их доказательства.

Рассмотрим теперь применение адаптивной системы логического вывода для автоматизированного формирования рекомендаций по

решению задач планирования поведения робота, распознавания и описания сложных изображений трехмерной среды, получаемых с помощью системы автоматизированного консультирования в условиях неопределенности.

5.2.2.3. Логические алгоритмы формирования рекомендаций по планированию поведения робота

Содержание задачи формирования рекомендаций по планированию поведения робота поясним на примере того, как эту задачу решает человек. Первое, с чем мы сталкиваемся ежедневно, — это задача утреннего одевания. Мы должны сформировать для себя рекомендации выработки плана действий, который позволит нам одеться, причем так, чтобы выполнялись естественные общепринятые ограничения (рубашку надевать необходимо, но не поверх пиджака, и т. п.). При этом время — наш основной ресурс, и выбранный план должен быть наилучшим в том смысле, в каком каждый понимает расход своего утреннего времени. Если отбросить некоторые «несущественные» детали, план одевания должен оперировать такими предметами, как туфли, носки, брюки, рубашка, галстук, пиджак и пальто. *Рекомендация по реализации плана действий представляет собой любой порядок, в котором можно надеть эти предметы.* Всего в этом случае существует $7! = 5040$ различных вариантов плана. Многие из них недопустимы, так как либо не удовлетворяют общепринятым ограничениям (рубашка поверх пиджака, носки поверх ботинок), либо непрактичны (галстук под рубашкой) и являются нереализуемыми рекомендациями. Но даже после того, как эти недопустимые рекомендации будут отброшены, все равно придется исследовать некоторое количество допустимых планов (рекомендаций). Как же выбрать окончательный (желательно, оптимальный) рекомендуемый план? Прежде всего заметим, что в рассматриваемой задаче имеется некоторая мера эффективности, некий критерий, позволяющий нам сравнивать эффективность рекомендуемых допустимых планов. Если мы можем каким-то образом сравнить значение этого критерия для различных планов, то мы сможем тем самым выбрать из них оптимальный. В данной конкретной задаче естественно рекомендовать минимизировать время, необходимое для того, чтобы одеться. Это и есть та мера эффективности рекомендации, с помощью которой можно сравнивать рекомендуемые допустимые планы. Тогда в качестве рекомендованного оптимального плана, позволяющего одеться, не нарушая общепринятых ограничений, можно выбрать следующий план: носки, рубашка, брюки, галстук, туфли, пиджак, пальто. Ясно, что при другой критерии эффективности поведения оптимальным может оказаться иной план.

Характерной особенностью консультационных задач планирования является наличие многих допустимых рекомендаций. После того, как эти рекомендации сформированы, возникает следующая задача: выбрать среди них по крайней мере одну оптимальную (в смысле определенного критерия) операцию плана действий. В рассмотренном нами простейшем примере план действий формировался без специальных обоснований, просто на основе опыта и здравого смысла. Оптимизация такого плана действий происходит как бы сама собой, в процессе жизненной практики. Нередки, правда, ситуации (соответствующие, например, планированию мероприятий, осуществляемых в первый раз), когда использовать эвристические решения, основанные на опыте и здравом смысле, просто невозможно. В подобного рода ситуациях «опыт» молчит, а «здравый смысл» легко может обмануть, если не будет опираться на математический расчет. Но бывают рекомендации несравненно более сложные, а главное ответственные — при их реализации от них очень многое зависит. Конечно, при планировании поведения в подобного рода ситуациях можно действовать интуитивно, опираясь опять-таки на опыт и здравый смысл. Но гораздо более разумными могут оказаться рекомендации, подкрепленные количественными, математическими расчетами соответствующего плана действий. Эти предварительные расчеты помогут избежать длительного и накладного поиска «на ощупь». «Семь раз примерь, один раз отрежь», — говорит известная поговорка. *Планирование поведения* как раз и представляет собой своеобразное математическое «примеривание» к потребному будущему, позволяющее заранее оценить последствия каждой рекомендации, заранее отбросить недопустимые планы и рекомендовать наиболее удачные. Эти последние позволят установить, достаточно ли имеющаяся у нас информация для правильного формирования рекомендации, и если нет — какую информацию нужно дополнительно получать и обрабатывать. Все это позволяет при реализации плана экономить время, энергию и материальные средства. Необходимость планирования поведения возникает у робота при выполнении им сложных заданий в условиях большой априорной неопределенности (например, сборка сложного изделия по чертежу, поиск и транспортировка нужного объекта на неизвестной местности с препятствиями и т. п.). Задача автоматизированного планирования поведения решаемая на втором уровне иерархии системы управления робота, может быть переформулирована на языке исчисления предикатов как задача логического вывода (автоматического доказательства теорем). При таком подходе априорные сведения о свойствах и функциональных

возможностях робота и окружающей его среды необходимо прежде всего представить в виде правильно построенных формул (ППФ). Совокупность таких ППФ мы будем называть *априорными аксиомами* и разобьем их на четыре класса:

- 1) сенсорные аксиомы (СА);
- 2) моторные аксиомы (МА);
- 3) аксиомы среды (АС);
- 4) аксиомы начальных условий (АНУ).

Сенсорные аксиомы описывают функциональные возможности информационно-измерительной системы робота, а *моторные аксиомы* — функциональные возможности исполнительных механизмов робота. *Аксиомы среды* определяют состояние и эволюцию среды, а *аксиомы начальных условий* описывают начальные состояния робота и среды. В табл. 5.1 и 5.2 приведены типичные для задачи планирования поведения робота на местности с препятствиями: термы, функции, предикаты и априорные аксиомы вместе с их интерпретацией на обычном (русском) языке.

Таблица 5.1

Функции и предикаты	Интерпретация на естественном языке
$f(a, b, s)$	Ситуация, наступающая после выполнения роботом, находящимся в ситуации s , действия «переехать из a в b по прямой».
$p(s)$	Ситуация, наступающая после выполнения манипулятором, находящимся в ситуации s , действия «погрузить объект на тележку».
$q(s)$	Ситуация, наступающая после выполнения манипулятором, находящимся в ситуации s , действия «сгрузить объект с тележки».
$G(a, b)$	Истинен, если из a в b можно проехать по прямой.
$Pos(s, x, y)$	Истинен, если в ситуации s робот находится в состоянии x , а объект — в состоянии y (предикат позиции).
$Pos(s, x, fin)$	Истинен, если в ситуации s объект находится на складе, указанном в задании, а робот — в состоянии x .
$Pos(s, fin, y)$	Истинен, если в ситуации s робот находится в конечной точке заданного маршрута, а объект — в состоянии y .

Наряду с априорными аксиомами введем *аксиомы обучения* (АО), автоматически формируемые по мере накопления роботом опыта и знаний в процессе выполнения тех или иных заданий. Задания роботу будем трактовать как заключения теорем, посылками которых служат априорные аксиомы и аксиомы обучения. Заметим, что формирование рекомендаций по выбору аксиом и теорем диктуется окружающим робота миром, который он воспринимает своими органами чувств и на который воздействует своими исполнительными механизмами, а также структурно-функциональными особенностями робота и целями (задачами) его функционирования.

Таблица 5.2

Тип	Аксиомы	Смысл на естественном языке
	Логическое представление	
МА	$\forall s \forall x \forall y \forall z (\neg Pos(s, x, y) \vee \neg G(x, z) \vee Pos(f(x, z, s), z, y))$	Если в ситуации s , робот находится в состоянии x , а объект — в состоянии y , и из x можно переехать по прямой в z , то в ситуации $f(x, z, s)$ (т. е. после реализации функции движения «переехать из точки x в точку z ») робот окажется в точке z .
АС	$G(a, b)$	Из точки a в точку b можно проехать по прямой.
СА	$K(x)$	Объект x принадлежит k -му классу.
АНУ	$Pos(s_0, O, C_1)$	В начальной ситуации s_0 робот находится в точке O , а объект — на складе C_1 .

Для автоматического доказательства теорем-заданий и извлечения ответа на языке робота целесообразно применить адаптивную систему логического вывода, описанную выше. Такая система в результате доказательства теоремы-задания (или теоремы-вопроса) сформирует рекомендации по указанию, какие действия и в какой последовательности нужно роботу совершить для выполнения задания,

т. е. выдаст рекомендации по реализации искомого план поведения робота.

Продemonстрируем работу адаптивной системы логического вывода в задаче формирования рекомендаций по планированию поведения робота на примере. Пусть робот находится в цехе с оборудованием (трактуемом как препятствия), где имеются склад заготовок C_1 и склад готовых изделий C_2 (см. рис. 5.9). Вначале робот находится в точке O и перед ним ставится задача: перевезти определенный объект (который еще нужно распознать) со склада C_1 на склад C_2 (местоположение складов известно) и после этого покинуть цех через выход. Предполагается, что выход задан набором признаков (его координаты роботу неизвестны), а сенсорная система может измерять значения признаков и координаты видимых ею точек, причем она не может «видеть» сквозь препятствия. Кроме перечисленного и исходной системы аксиом, представленной в табл. 5.2, роботу ничего неизвестно.

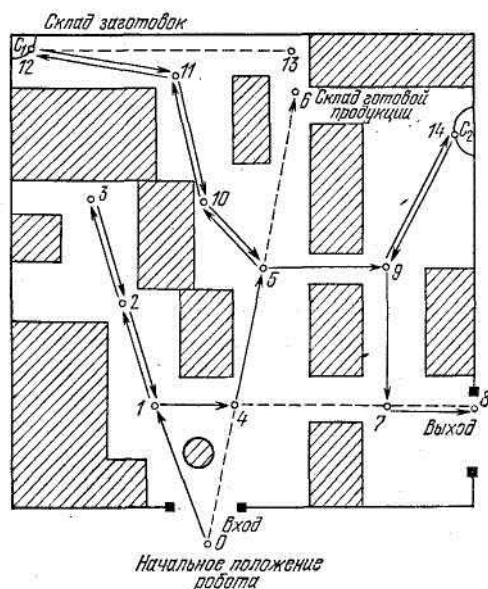


Рис. 5.9. Планирование поведения робота в незнакомом помещении.

Поскольку решение этой задачи требует дополнительной информации об обстановке в цехе, робот вначале опрашивает сенсорную систему. При этом отыскиваются все видимые границы препятствий и около них фиксируются некоторые точки 1-6, к которым

робот может проехать по прямой. В результате автоматически строятся аксиомы среды: $G(0, 1)$, $G(1, 2)$, $G(2, 3)$, $G(0, 4)$, $G(4, 5)$, $G(5, 6)$. По этим данным, а также по аксиоме начальных условий $Pos(s_0, O, C_1)$ робот пытается доказать теорему-задание $\exists s \text{ Pos}(s, \text{fin}, \text{fin})$ (где s — переменная, описывающая ситуацию). Однако, поскольку знаний о среде, заключенных в построенных АС, явно недостаточно (теорема не выводима из АС), ответ, т. е. рекомендация по реализации искомого плана поведения, не будет получен. В процессе логического вывода робот убеждается, что маршруты через точки $O, 1, 2, 3$ и $O, 4, 5, 6$ к выполнению задания не приводят. Далее, используя критерий близости к складу C_1 , робот принимает решение переместиться в точку 3. Последовательность действий на этом этапе определяется термом ситуации в резольvente $f(2, 3, f(1, 2, f(0, 1, s_0)))$, который расшифровывается в обратном порядке и в соответствии с определением функции f означает: переехать из точки O в точку 1, затем в точки 2 и 3. Передвигаясь согласно выбранному маршруту, робот останавливается в каждой из них и пополняет свои знания о среде посредством опроса сенсорной системы. Так возникают новые аксиомы среды (АС): $G(1, 4)$, $G(4, 7)$, $G(7, 8)$, $\neg \text{Pos}(s, 8, y) \vee \text{Pos}(s, \text{fin}, y)$ (последняя аксиома среды означает, что точка 8 находится у выхода).

Приехав в точку 3, робот не формирует ни одной новой АС, а поэтому и новой резольвенты. Он в тупике. «Осознав» это, робот вынужден развернуться и исправить те действия, которые привели его в «тупиковую» точку 3, в обратном порядке, пока не появится первая возможность образования новой резольвенты. В результате он выбирает маршрут через точки 1, 4, 5, используя ранее построенные аксиомы обучения (АО) (к числу которых относятся АС и СА, формируемые в процессе функционирования робота). Так, формируя и корректируя рекомендации по реализации локальных планов поведения на основе целенаправленной переработки новой информации, робот, в конце концов, решает поставленную задачу: отыскивает (распознает) нужный объект на складе C_1 , погружает (с помощью манипулятора) его на тележку, подвозит к складу C_2 , сгружает объект и покидает цех через выход. При этом адаптивная система логического вывода строит 47 резольвент. Окончательный маршрут робота, реализующий выработанный план поведения, изображен на рис. 5.9 сплошными линиями со стрелками.

Рассмотренный пример формирования рекомендаций по планированию поведения робота в условиях большой априорной (начальной) неопределенности замечателен тем, что он ясно демонстрирует, что обычная (неадаптивная) система логического вывода принципиально не способна решать такого рода задачи без

использования элементов обучения и адаптации. Важно отметить, что автоматическое формирование АО и адаптивная подстройка стратегии, использующей АО, не только делает задачу разрешимой, но и существенно сокращает (за счет отсеечения многих тупиковых ветвей на дереве вывода) число шагов в процессе поиска плана поведения как при полной, так и при частичной информированности об условиях функционирования робота.

5.2.2.4. Алгоритмы распознавания ситуаций

Задача по описанию, распознаванию и анализу ситуаций, котрыми являются консультируемые проблемы, является одной из центральных проблем формирования рекомендаций с использованием автоматизированных консультационных процессов. Для формализации и автоматизации решения этой задачи, как мы увидим ниже, опять-таки удобно использовать язык исчисления предикатов и адаптивную систему логического вывода. В качестве примера, на котором мы будем демонстрировать использование языка исчисления предикатов и адаптивной системы логического вывода, рассмотрим функционирование робота. Поскольку для робота наиболее важное значение имеет зрительная информация, мы сосредоточим внимание на методах формированию рекомендаций по описанию, распознаванию и анализу трехмерных сцен по их изображениям. Что же касается задач переработки речевой, тактильной и другой сенсорной информации, то они могут решаться по существу теми же методами. Трудности, возникающие при формировании рекомендаций для решения задачи распознавания отдельных предметов на сложной сцене, связаны с наличием «порочного круга»: для того чтобы распознать некоторый предмет на сцене, нужно прежде всего его «выделить», а для того чтобы «выделить» этот предмет, нужно его распознать. Это приводит к тому, что классические методы распознавания «перцептронного» или статистического типа, в этой задаче практически не применимы. На первый взгляд кажется, что выход из указанного «порочного круга» только один — полный перебор элементов изображения предмета и сцены. Однако более глубокий анализ этой задачи позволяет сформировать рекомендации по ее формулировке и решению как задачи логического вывода.

Идея предлагаемого решения основана, во-первых, на том, что применяется предварительное обучение робота путем показа ему отдельных предметов из различных классов и сообщения ему не только названия предмета, но и, возможно, его описания. Это обучение ведется постановщиком задач в диалоговом режиме, позволяющем

оперативно выявлять и исправлять ошибки робота. Во-вторых, специфика задачи ярко проявляется в большой вариативности изображений реальных предметов и сцен, которая имеет двоякую природу. С одной стороны, она порождается естественной вариативностью характеристик самих предметов, с другой стороны, — перемещением предметов в пространстве. Вариативность второго рода можно трактовать как результат действия некоторых известных преобразований изображения. Априорное знание этих преобразований позволяет построить алгоритм распознавания, инвариантный по отношению к этим преобразованиям. Благодаря этому удастся не только «избавиться» от вариативности второго рода, но и существенно облегчить задачу переработки зрительной информации.

Сама эта задача подразделяется на следующие подзадачи: *описание классов (формирование понятий о классах объектов), распознавание изображения данного предмета, анализ изображения сцены (распознавание всех предметов на сцене)*. Результаты решения перечисленных подзадач могут использоваться для формирования рекомендаций по моделированию внешней среды (на четвертом уровне иерархии) путем преобразования изображений предметов и сцен в их пространственное представление, а также для описания сцен на естественном языке или, наоборот, для синтеза изображения сцены по ее описанию.

Рассмотрим сначала *задачу описания классов* (формирования понятий). Эта задача решается в режиме обучения. Роботу последовательно предъявляют предметы из различных классов (и, возможно, в различных ракурсах) с указанием, к какому классу каждый такой предмет принадлежит. **Эти предметы (а также их изображения) называют эталонными, а совокупность классифицированных предметов — обучающей выборкой.** По этим данным робот должен автоматически сформировать описание классов в терминах тех свойств предметов, которые непосредственно измеряются сенсорной системой. Примерами этих свойств, которые мы будем называть первичными признаками, являются следующие: «красное», «ближе», «правее», «выше», «две точки соединены отрезком», «два отрезка параллельны», «зоны одинаковой яркости» и т. п. Таким образом, **изображения предметов и сцен задаются полным набором своих первичных признаков.**

Каждому признаку поставим в соответствие предикат $\xi_i(x_1, \dots, x_{ni})$, где $x_1, \dots, x_{ni}$ — элементы изображения ω , определяющие наличие на нем i -го признака. Тогда каждому эталону ω_h (предмету из обучающей выборки) соответствует набор значений предикатов $\xi_i^{(h)}(c_1, \dots, c_{ni})$, истинных на изображении данного предмета ω_h . Здесь c_j —

предметные константы, означающие фиксированные части изображения. Описанием изображения эталона ω_h будем называть конъюнкцию

$$\bigwedge_{i=1}^{P_h} \xi_i^{(h)}(c_1, \dots, c_{ni}).$$

Поскольку к одному и тому же классу могут принадлежать несколько эталонов (например, предмет из этого класса, показанный в разных ракурсах), то описанием всех эталонных изображений, принадлежащих данному классу Ω_k , является дизъюнкция

$$\bigvee_{\omega_h \in \Omega_k} \bigwedge_{i=1}^{P_h} \xi_i^{(h)}(c_1, \dots, c_{ni}).$$

Если теперь в этой дизъюнкции все предметные константы c_j заменить на соответствующие предметные переменные x_j , то получим ППФ, которую естественно назвать описанием класса Ω_k . Введем предикат $\sigma(k)$, означающий принадлежность изображения классу Ω_k . Тогда каждый класс Ω_k описывается аксиомой класса (АК) вида

$$\bigvee_{\omega_h \in \Omega_k} \bigwedge_{i=1}^{P_h} \xi_i^{(h)}(x_1, \dots, x_{ni}) \rightarrow \sigma(k), \quad (5.1)$$

(которая по существу является логическим определением класса (или соответствующего ему понятия). Из вышеизложенного ясно, что АК вида (5.1) могут строиться роботом автоматически в режиме обучения по мере последовательного предъявления ему эталонов.

Практически важно, чтобы система АК обладала свойствами полноты, независимости и инвариантности в естественных смыслах. Дадим развернутое определение этих свойств.

Систему АК будем называть *полной* на множестве изображений $\{\omega\}$, если для всякого изображения ω из этого множества найдется АК, принимающая на нем значение «истинно». Следует отметить, однако, что полнота системы АК не исключает того, что для некоторого изображения могут найтись две АК, принимающие на нем значение «истинно».

В некоторых случаях требуется, чтобы ни одно исследуемое изображение не было отнесено одновременно к нескольким классам (например, если априори известно, что распознаваемые классы изображений не пересекаются). Это требование должно быть отражено в АК. Систему АК будем называть *непротиворечивой*, если существует только одна АК, истинная на любом данном изображении. Из приведенного определения следует, что непротиворечивая система АК исключает возможность пересечения классов, описываемых этой

системой аксиом. Очевидно, что непротиворечивость системы АК всегда можно эффективно проверить.

Вариативность изображений, порождаемая пространственными преобразованиями воспринимаемых предметов, а также действием разного рода помех и искажений, требует, чтобы система обладала определенной инвариантностью и помехозащищенностью. Например, всевозможные изображения стола, отличающиеся от эталонного (по которому строится соответствующая АК) сдвигом, поворотом, масштабом, а также некоторыми незначительными искажениями или помехами (лишние линии, незначительные изменения пропорций и т. п.), должны описываться одной и той же АК «стол», т. е. должны классифицироваться как эквивалентные. Систему АК будем называть *инвариантной* по отношению к заданной группе преобразований, если каждая входящая в нее аксиома принимает одно и то же значение на изображениях, отличающихся преобразованиями из этой группы. Таким образом, инвариантность системы АК позволяет «снять» охарактеризованную выше вариантность изображений и тем самым облегчить распознавание сцен по их изображениям.

Задачи распознавания и анализа изображений могут быть переформулированы как задачи логического вывода. Эти задачи решаются в режиме распознавания. Роботу предъявляются сцена, изображение $\tilde{\omega}$ которой может содержать одно или несколько изображений предметов. Эти изображения отличаются от эталонов некоторыми преобразованиями и даже могут частично перекрываться. Описание изображения сцены $S(\tilde{\omega})$ представляет собой конъюнкцию всех первичных предикатов, истинных на данном изображении $\tilde{\omega}$. В этих условиях задача отыскания на изображении $\tilde{\omega}$ изображения из определенного класса Ω_k , т. е. задача распознавания, сводится к нахождению доказательства теоремы $S(\tilde{\omega}) \rightarrow \sigma(k)$. Саму эту теорему можно трактовать как вопрос: имеется ли на данном изображении $\tilde{\omega}$ предмет из k -го класса? Ясно, что в процессе ответа на этот вопрос описание $S(\tilde{\omega})$ может быть использовано не полностью. Поэтому измерение тех или иных первичных признаков и предикатов должно производиться по мере необходимости в процессе логического вывода. Задача анализа изображения сцены заключается в распознавании на ней всех изображений предметов из различных классов. Формально эта задача сводится к последовательному доказательству теорем $S(\tilde{\omega}) \rightarrow \exists \sigma(k), i = 1, \dots, N - 1$, где $S_i(\tilde{\omega}) = S(\tilde{\omega})$, а $S_{i+1}(\tilde{\omega})$ получается из $S_i(\tilde{\omega})$ вычеркиванием всех предикатов, участвовавших в выводе i -й теоремы. Содержательно это означает, что, как только выделяется

очередное изображение предмета из некоторого класса, оно при дальнейшем анализе не рассматривается. Полный анализ изображения сцены заканчивается распознаванием (и тем самым выделением) всех видимых изображений предметов, составляющих сцену.

В режиме обучения и распознавания на различных изображениях может встречаться один и тот же набор первичных признаков, характеризующих, например, фрагмент изображения. В таких случаях естественно ввести вторичные признаки, каждый из которых представляет собой некоторую совокупность из первичных признаков, а также вторичные предикаты, определяющие соответствующие фрагменты изображения как новые понятия.

Аксиомами обучения (АО) будем называть ППФ вида

$$\bigwedge_{j=1}^r \xi_j(x_{j1}, \dots, x_{jm}) \rightarrow \alpha_i, \quad (5.2)$$

где $\xi_j(x_{j1}, \dots, x_{jm})$, $j = 1, \dots, r$ — первичные предикаты, дающие полное описание вторичного предиката α_i , т. е. определяющие некоторый фрагмент изображения как новое понятие. Использование АО позволяет не только более экономно представить АК, но и повысить эффективность системы логического вывода в процессе распознавания и анализа.

Как мы уже отмечали, универсальным средством логического вывода в исчислении предикатов является метод резолюций. Поэтому любой конкретный алгоритм распознавания или анализа определяется стратегией метода резолюций. Важно отметить, что для распознавания на изображении сцены нужного предмета не обязательно строить доказательство теоремы $S(\tilde{\omega}) \rightarrow \sigma(k)$ полностью, т. е. перебирать все элементы искомого простого изображения на изображении сцены $\tilde{\omega}$. Вместо этого достаточно найти фрагмент искомого изображения, который содержится лишь в изображениях k -го класса и не содержится ни в одном изображении предметов из других классов. Именно это обстоятельство позволяет сильно ограничить число шагов логического вывода, а также распознавать частично закрытые изображения предметов.

Качество работы алгоритмов распознавания и анализа естественно характеризовать числом обращений к информационно-измерительной (сенсорной) системе с целью определения нужных признаков. Заметим, что в нашей формализации это в точности совпадает с числом шагов логического вывода, т. е. с числом резольвент, формируемых в процессе распознавания и анализа. Стратегию логического вывода будем называть *оптимальной*, если число шагов доказательства (число резольвент) минимально.

Важно, отметить, что система логического вывода способна совершенствовать алгоритмы распознавания и анализа за счет использования элементов обучения, а именно: АО и адаптивной подстройки стратегии. Адаптивная подстройка стратегии осуществляется путем ее перестройки (например, путем перестройки оптимального распознающего графа) по мере распознавания новых изображений и формирования аксиом обучения (АО). Использование АО сокращает логический вывод на длину ее описания, а процесс построения оптимального распознающего графа — на значение экспоненциальной функции от этой длины.

Проиллюстрируем описанный метод на примере формирования рекомендаций по решению задачи описания, распознавания и анализа обстановки в цехе. Предположим, что в результате предварительной фильтрации исходные изображения предметов и сцен превращаются в контурные изображения, составленные из отрезков. Введем необходимые для дальнейшего первичные признаки и предикаты, представленные в табл. 5.3. Вторичные признаки и предикаты (аксиомы обучения) представлены в табл. 5.4.

Таблица 5.3

Первичный признак	Предикат	Интерпретация
Вертикальный отрезок	$BT(x, y)$	Истинен, если точки x и y соединены вертикальным отрезком.
Горизонтальный отрезок	$GP(x, y)$	Истинен, если точки x и y соединены горизонтальным отрезком.
Параллельность двух отрезков	$ПРЛ(x, y, u, v)$	Истинен, если отрезки (x, y) и (u, v) параллельны.

Таблица 5.4

Вторичный признак-понятие	Символ понятия	Логическое описание понятия
Параллелограмм	$ПР (x_1, x_2, x_3, x_4)$	$ПРЛ (x_1, x_2, x_3, x_4) \wedge$ $ПРЛ (x_1, x_4, x_2, x_3)$
Горизонтальный параллелограмм	$ГПР (x_1, x_2, x_3, x_4)$	$ПР (x_1, x_2, x_3, x_4) \wedge$ $ГР (x_1, x_2) \wedge ГР (x_2, x_3)$
Вертикальный прямоугольник	$ВПР (x_1, x_2, x_3, x_4)$	$ПР (x_1, x_2, x_3, x_4) \wedge$ $ВТ (x_1, x_2) \wedge ГР (x_2, x_3)$

В режиме обучения роботу предъявляются эталонные контурные изображения токарного и сверлильного станков, представленные на рис. 5.10.

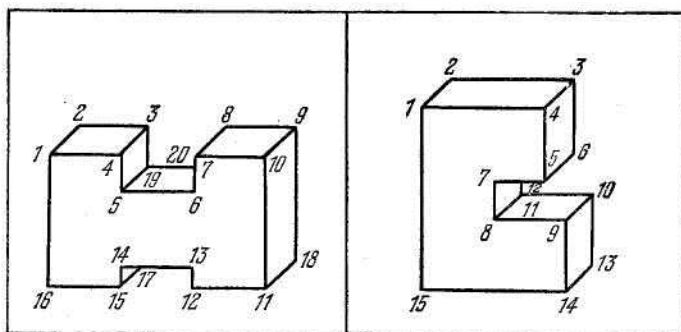


Рис. 5.10. Эталонные изображения (режим обучения).

По этим данным он строит АК, общий вид которых приведен в табл. 5.5.

Таблица 5.5

Класс-понятие	Символ класса	Логическое описание класса
Токарный станок	σ (ТС)	$ГПР(x_1, x_2, x_3, x_4) \wedge ВПР(x_5, x_4, x_3, x_{19}) \wedge ГР(x_{19}, x_{20}) \wedge ГР(x_5, x_6) \wedge ВТ(x_6, x_7) \wedge ГПР(x_7, x_8, x_9, x_{10}) \wedge ВПР(x_{11}, x_{10}, x_9, x_{18}) \wedge ГР(x_{12}, x_{11}) \wedge ВТ(x_{12}, x_{13}) \wedge ГР(x_{14}, x_{13}) \wedge ВТ(x_{15}, x_{14}) \wedge ГР(x_{15}, x_{17}) \wedge ГР(x_{16}, x_{15}) \wedge ВТ(x_{16}, x_1)$
Сверлильный станок	σ (СС)	$ГПР(x_1, x_2, x_3, x_4) \wedge ВПР(x_5, x_4, x_3, x_6) \wedge ГР(x_7, x_5) \wedge ВТ(x_8, x_7) \wedge ГПР(x_9, x_{11}, x_{10}, x_9) \wedge ВТ(x_{11}, x_{12}) \wedge ВПР(x_{14}, x_9, x_{10}, x_{13}) \wedge ГР(x_{15}, x_{14}) \wedge ВТ(x_{15}, x_1)$

Так в режиме обучения автоматически формируется описание классов

В режиме распознавания роботу предъявляется изображение произвольной сцены — обстановки в цехе, попавшей в «поле зрения» робота. Для определенности пусть это будет изображение сцены, представленное на рис. 5.11.

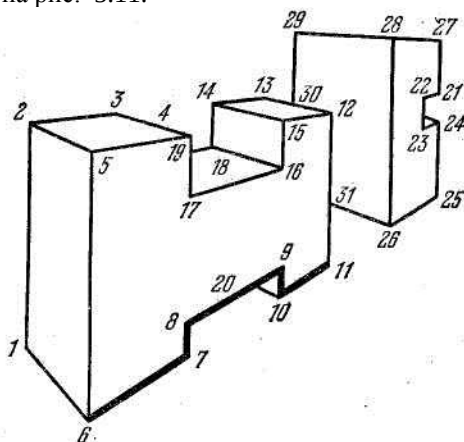


Рис. 5.11. Изображение сцены (режим распознавания).

Описание этой сцены (на языке первичных и вторичных предикатов) имеет вид

1. $ВПР(6, 5, 2, 1)$; 2. $ГПР(5, 2, 3, 4)$; 3. $ВТ(17, 4)$; 4. $ГР(17, 16)$; 5. $ВПР(16, 15, 14, 18)$; 6. $ГР(19, 18)$; 7. $ГПР(15, 14, 13, 12)$; 8. $ВТ(11, 12)$; 9.

ГР (10, 11); 10. ВТ (10, 9); 11. ГР (10, 20); 12. ГР (8, 9); 13. ВТ (7, 8); 14. ГР (6, 7); 15. ВТ (30, 28); 16. ГР (29, 28); 17. ГР (29, 27); 18. ВТ (21, 27); 19. ГР (22, 21); 20. ВТ (23, 22); 21. ГР (23, 24); 22. ВТ (25, 24); 23. ГР (26, 25); 24. ВТ (26, 29); 25. ГР (26, 31).

Предположим, что мы спрашиваем робота (или он сам задается этим вопросом): «есть ли на изображении сцены токарный станок?». Для ответа на вопрос робот пытается доказать теорему

$$\bigwedge_{j=1}^{25} \xi_j (c_{j1}, \dots, c_{jk}) \rightarrow \sigma (TC),$$

где $\xi_j (c_{j1}, \dots, c_{jk}), j = 1, \dots, 25$, — предикаты, входящие в приведенное выше описание изображения сцены.

Доказательство этой теоремы с помощью стратегии лозы потребовало в зависимости от порядка записи АК от 8 до 54 резольвент. Однако, как уже отмечалось, для распознавания токарного станка на изображении сцены полное доказательство соответствующей теоремы не обязательно, — возможно распознавание по фрагменту. Использование оптимальной стратегии в этом примере позволило сократить число резольвент до 5. При этом был автоматически выделен и существенно использовался в процессе логического вывода фрагмент изображения, обведенный на рис. 5.11 жирной линией. Интересно отметить, что решение той же задачи без использования АО приводит к существенному снижению эффективности системы логического вывода. В этом случае описание изображения сцены требует уже не 25, а 47 первичных предикатов. В процессе распознавания токарного станка с помощью стратегии лозы строится по меньшей мере 73 резольвенты.

Резюмируя содержание п. 5.2.2.1—5.2.2.4, отметим, что *язык исчисления предикатов и адаптивная система логического вывода* позволяют эффективно решать не только задачу автоматизированного формирования рекомендаций для планирования поведения робота в условиях априорной неопределенности, но и задачи описания классов, распознавания и анализа ситуаций в окружающей робота среде.

5.2.2.5. Моделирование внешней среды

Любая система, вырабатывающая целесообразное поведение, — будь то человек, животное или робот, — всегда использует знания о своих собственных функциональных возможностях и о свойствах внешней среды. О таких системах принято говорить, что они обладают «внутренней моделью внешнего мира». Наличие модели среды позволяет быстро и глубоко проникать в механизмы внешних явлений, намного облегчает процессы обучения на опыте и адаптации.

Каковы же особенности и способы представления знаний о внешнем мире в памяти робота? Прежде всего заметим, что способность к моделированию внешней среды (включая свое собственное «я») присуща лишь интеллектуальным роботам.

Окружающий нас мир настолько сложен, что не только робот, но и человек бывает доволен, если ему удастся уловить и понять хотя бы некоторые самые простые из присущих этому миру закономерностей. Для этого человек строит упрощенные и идеализированные модели, освобожденные от маловажных подробностей и отражающие, как он надеется, наиболее существенные свойства рассматриваемых реальных объектов. По такой же схеме должен действовать и интеллектуальный робот, желающий создать в себе адекватную модель внешнего мира.

Восприятие внешнего мира осуществляется с помощью искусственных органов чувств робота. Это значит, что реальные ситуации описываются в памяти робота с помощью набора показаний сенсорных датчиков. Именно в терминах этих показаний — элементарных высказываний о свойствах среды и самого робота — и формируется «внутренняя модель внешнего мира».

Однако совокупность показаний сенсорных датчиков — это «сырая» информация, которую мы будем называть **первичным описанием реальной ситуации**. Анализ и обработка этой информации, как мы видели в предыдущем разделе, позволяют строить **обобщенные описания ситуаций, называемые понятиями**. В эти информационные представления о мире могут входить не только показания органов чувств (например, «красное», «горячее», «твердое» и т. п.), но и утверждения о соотношениях между этими показаниями. Одним из наиболее удобных средств для формирования и обработки этих представлений является описанное выше исчисление предикатов; вместе с адаптивной системой логического вывода. **На языке исчисления предикатов свойства внешнего мира задаются с помощью элементарных предикатов и функций от показаний сенсорных датчиков, а также с помощью логических средств формирования из этих элементов сложных утверждений (описаний) в зависимости от имеющегося опыта и поступающей информации**. В двух предыдущих пунктах мы видели, каким образом осуществляется уточнение (коррекция) и расширение представлений робота о внешнем мире в процессе обучения на опыте и адаптации к существующим (заранее неизвестным) условиям.

Моделирование внешнего мира в памяти робота не является самоцелью. Оно служит главным образом для «мысленного» планирования поведения и принятия решений о тех или иных целенаправленных действиях робота. При этом весьма важно, чтобы в

модели внешнего мира был отражен сам робот, его структурные и функциональные свойства, особенности взаимодействия с окружающей средой. Благодаря этому робот получает возможность анализировать не только окружающую среду, в которой он функционирует, но и свое поведение в этой среде. Мы можем смело сказать, что интеллектуальный робот обладает «сознанием» и «самосознанием». Уточним эти интуитивно понятные термины.

Следуя терминологии, предложенной Д. А. Поспеловым, «сознанием» **робота мы будем называть его способность отображать (моделировать) внешнюю среду в своей памяти и анализировать закономерности этой среды, а также результаты своих воздействий на среду.** Под «самосознанием» будем понимать свойство робота **отображать себя в модели среды и анализировать закономерности воздействия среды на свою структуру и функционирование.** Именно «сознание» и «самосознание» принципиально отличают интеллектуальных роботов.

5.2.2.6. Алгоритмы построения программных движений

Задача построения программных движений (ПД) робота решается на пятом уровне иерархии его системы управления. Особенность этой задачи заключается в том, что движение исполнительных механизмов робота задается законом изменения обобщенных координат, а происходит в реальной внешней среде. При этом допустимость тех или иных конфигураций исполнительных механизмов непосредственно наблюдается в реальной же среде и никаким простым способом не выражается «на языке» пространства обобщенных координат. Другой особенностью является наличие двигательной избыточности, присущей роботам с большим числом степеней свободы. Эта избыточность, однако, полезна. Она позволяет среди множества возможных ПД отобрать наиболее «экономные», оптимальные программные движения.

Рассмотрим кратко некоторые идеи и алгоритмы построения ПД. Эти алгоритмы служат для того, чтобы робот мог, еще не совершая каких-либо реальных движений, «мысленно» рассчитать, а если нужно — скорректировать программу своих целенаправленных движений. В качестве примера рассмотрим задачу управления многозвенным манипулятором. Цель управления в этом случае заключается в отслеживании схватом некоторой траектории, рассчитанной на более высоком уровне планирования поведения. Задача осложняется наличием разного рода препятствий и конструктивных ограничений.

Программным движением называется такой закон изменения обобщенных координат, при котором достигается цель

управлений, удовлетворяются конструктивные ограничения и обеспечивается обход препятствий.

Идея «глобального» подхода к построению ПД заключается в следующем. Вначале на основе информации о траектории схвата с учетом конструктивных ограничений и препятствий формируется **план движения — последовательность промежуточных конфигураций, ведущая к цели.** Затем по «кускам» (например, в классе кусочно-полиномиальных функций) строится само ПД в соответствии с этим планом.

Большой интерес представляет также «локальный» подход, основанный на моделировании так называемого тропизма.

Тропизм — это свойство, присущее простейшим живым организмам; оно заключается в направленных движениях организмов в результате действия односторонних раздражителей. Применительно к уровню построения ПД под таким «раздражителем» будем понимать информацию о взаимном расположении манипулятора и препятствий в реальном трехмерном пространстве, а под тропизмом — движения манипулятора, направленные на удаление от препятствий. Не приводя (ввиду громоздкости) явные формулы алгоритма построения ПД, имитирующего свойства тропизма, отметим только, что синтезируемые им целенаправленные движения реализуются за счет двигательной избыточности манипулятора, в то время как схват движется по заданной траектории.

Мы бегло охарактеризовали различные способы построения ПД манипулятора. А как строить ПД колесного или гусеничного шасси для робота, функционирующего на местности с препятствиями? Эта задача естественным образом распадается на две: *прокладка безопасного маршрута и построение закона изменения обобщенных координат, удовлетворяющего конструктивным ограничениям и обеспечивающего движение по этому маршруту.*

Рассмотрим один из подходов к построению *оптимального маршрута* на местности с препятствиями, основанный на методе динамического программирования. Для простоты предположим, что местность представляет собой плоскость, а препятствия задаются ломаными линиями. Пусть заданы координаты *исходной точки* (где находится робот) и *целевой точки* (куда он должен передвинуться). **Задача заключается в построении маршрута (ломаной линии) из исходной точки в целевую, который не пересекает ломаных препятствий и имеет наименьшую длину.** Такой маршрут будем называть оптимальным.

Прежде всего заметим, что если сформулированная задача не имеет тривиального решения (когда маршрутом является просто отрезок,

соединяющий исходную и целевую точки), вершинами ломаной наименьшей длины должны быть вершины ломаных — препятствий. Поэтому в дальнейшем будем рассматривать только эти вершины. Введем функцию f_i определяющую минимальную длину пути из некоторой точки i в целевую точку. Сразу же возникает вопрос — как найти эту функцию? Один из классических способов нахождения функции состоит в том, чтобы задать уравнения, определяющие эту функцию.

Приступая к поиску такого уравнения, зададимся вопросом: существует ли такое внутреннее свойство процесса поиска оптимального пути, которое можно использовать для получения уравнения? Ответ, к счастью, положительный. В самом деле, анализируя задачу, мы видим, что отправляясь из точки i , робот на первом шаге попадет в точку j , пока неясно, какую именно. Далее, мы замечаем, что, если робот ищет кратчайший путь от точки i до целевой точки, то в какую бы точку он ни попал, путь из точки j в целевую должен иметь наименьшую длину. Это почти очевидное свойство докажем от противного. Пусть путь из точки i в целевую точку является кратчайшим, тогда часть пути из точки j в целевую точку также должна иметь минимальную длину, так как, если бы эта часть пути не была кратчайшей, то ее можно было бы заменить более кратким путем и тем самым сократить общую длину пути, а это противоречит тому, что f_i по определению есть минимальная длина пути.

Таким образом, мы установили существенное внутреннее свойство процесса поиска оптимального маршрута: «хвост» (конец) процесса — это всегда кратчайший маршрут. Что все это дает для определения функции f_i ? Обозначим через s_{ij} — длину пути между точками i и j (хотя все еще неизвестно, что это за точка j), а через f_j — наименьшую длину пути между точкой j и целевой точкой. Выбирая в качестве точки j такую точку, которая минимизирует сумму $s_{ij} + f_j$, получаем уравнение

$$f_i = \min_{j \neq i} [s_{ij} + f_j]. \quad (5.3)$$

Это — основное уравнение динамического программирования.

Оно обладает тем свойством, что задает две функции f_i и $j(i)$. В самом деле, если мы нашли f_i , то значение j , на котором достигается минимум (5.3), как раз и указывает ту точку, в которую роботу нужно двигаться дальше. Важно отметить, что функция $j(i)$ представляет собой по существу стратегию поиска оптимального маршрута, т. е. правило, которое позволяет роботу, находящемуся в произвольной точке i , определить, куда ему следует двигаться дальше.

Трудность решения уравнения (5.3) заключается в том, что неизвестная функция входит в обе части равенства. В такой ситуации приходится прибегать к классическому методу последовательных приближений в функциональном пространстве, используя рекуррентную формулу

$$f_i^{(k+1)} = \min_{j \neq i} [s_{ij} + f_j^{(k)}], \quad (5.4)$$

где $f_i^{(k)}$ — k -е приближение искомой функции. Можно доказать сходимость алгоритма (5.4) для произвольной неотрицательной начальной функции f_i^0 с единственным ограничением, что значение функции f в целевой точке равно нулю.

Решение уравнения (5.3) можно искать и в пространстве стратегий. Этот путь представляется более естественным: «мыслить» в терминах пространства стратегий роботу более удобно, чем в терминах искусственно выбранных функций f_i . К тому же понятие стратегии сохраняет смысл и в тех случаях, когда функция не может быть определена. Какую же приближенную стратегию можно выбрать в задаче прокладки оптимального маршрута? Одна из возможных стратегий такова: робот решает непосредственно двигаться из точки i в целевую точку. Тогда получается приближение $f_i^{(0)} = s_{iц}$, где $s_{iц}$ — длина пути между точкой i и целевой точкой. Следующее приближение получится, если робот будет искать решение в классе двухзвенных ломаных. Дальнейшие приближения ищутся в классе трехзвенных, четырехзвенных и т. д. ломаных.

Одно из преимуществ выбора приближений в пространстве стратегий состоит в следующем: выбрав из общих соображений стратегию, мы тут же получаем соответствующую функцию $f_i^{(k)}$. Понятно, что $f_i^{(k)} \leq f_i$. В заключение, не входя в детали, отметим только, что описанный метод последовательных приближений в пространстве стратегий действительно сходится к решению уравнения (5.3) и допускает чрезвычайно простую программную реализацию.

После того, как оптимальный безопасный маршрут построен, можно тем или иным методом (например, с помощью кусочно-полиномиальной аппроксимации) построить программное движение колесного или гусеничного шасси робота. Алгоритмическое решение этой последней задачи обычно не вызывает никаких принципиальных затруднений.

5.2.2.7. Алгоритмы адаптивного управления движением

После того как программное движение (ПД) робота построено, возникает следующая задача: *сформировать рекомендации по нахождению закона управления исполнительными приводами,*

реализующий ПД. Решение этой задачи существенно осложняется *неопределенностью* условий функционирования робота (проблемы). Природа этой неопределенности многообразна: изменение характеристик исполнительных двигателей и механизмов (например, в результате старения и износа), технологические допуски, возможные неисправности, а также изменение условий, внешних по отношению к роботу, но оказывающих на него полностью или частично неконтролируемые воздействия. Если степень неопределенности велика, то для формирования рекомендаций по построению законов управления ПД робота классические методы теории автоматического управления могут оказаться недостаточными. В подобных условиях рекомендации по широко используемым следящим приводам, параметры которых выбираются из априорных соображений о «средних» условиях функционирования, зачастую не обеспечивают реализацию ПД с требуемой точностью. Поэтому возникает необходимость сформировать рекомендации по использованию адаптивного управления, восполняющем недостающую информацию в процессе функционирования робота. Синтез *адаптивного управления ПД* осуществляется в два этапа. Вначале в предположении, что уравнения движения робота полностью известны, рекомендуется строить закон управления, обеспечивающий близость (с любой наперед заданной точностью) реального и программного движений. Эту «неадаптивную» задачу можно рекомендовать решать методами классической теории управления. Однако воспользоваться таким «идеальным» законом управления практически нельзя, так как он зависит от ряда *варьируемых параметров* уравнения движения, значения которых неизвестны. Варьируемыми параметрами могут быть, например, масса и моменты инерции объекта манипулирования, распределение нагрузки на шасси робота, коэффициенты сцепления с грунтом и т. п. Поэтому на втором этапе на основе «идеального» закона рекомендуется построить адаптивное управление, обеспечивающее близость реального и ПД при любых возможных изменениях варьируемых параметров.

Идея адаптивного управления проста. Она заключается в замене неизвестных параметров «идеального» закона управления их оценками, которые должны целенаправленно «настраиваться» в процессе функционирования робота. Алгоритмы, по которым осуществляется «настройка» параметров управления, принято называть *алгоритмами адаптации*. Многие известные в настоящее время алгоритмы адаптации с математической точки зрения представляют собой *разностные или дифференциальные уравнения*,

определяющие закон целенаправленного изменения параметров управления на основе сенсорной информации.

Ярким примером эффективных алгоритмов адаптации, допускающих простую реализацию, могут служить *конечно-сходящиеся* алгоритмы решения целевых неравенств. Смысл целевых неравенств заключается в том, что если они выполнены, то закон управления, в котором вместо неизвестных параметров используется их оценка, обеспечивает требуемую близость реального и ПД. В этом случае коррекция («настройка») параметров не производится. Если же целевые неравенства в некоторый момент времени нарушаются, то осуществляется адаптивная коррекция параметров по некоторому алгоритму. Этот алгоритм может быть выбран так, что число коррекций будет конечным (и даже минимально возможным). Описанный конечно-сходящийся процесс адаптации хорошо согласуется с содержательным представлением об адаптации. В самом деле, с интуитивной точки зрения адаптация должна проявляться в том, что в неизменяющихся («стационарных») условиях функционирования робота (проблемы) алгоритм адаптации «работает» не все время, а с течением времени «отключается», осуществляя переход на автоматическое управление без «настройки» параметров. Лишь значительное изменение условий функционирования робота (проблемы) вызывает необходимость «включения» алгоритма адаптации для коррекции параметров управления.

5.2.3. Описание облачных процессов сетями Петри

5.2.3.1. Сети Петри и их модификация

Существуют различные типы УС, однако независимо от их структуры УС и выбранного принципа его реализации исходным материалом для проектирования УС является выполняемый им управляющий процесс (УП). После распределения функций между ОУ и УС и выявления взаимодействия между этими двумя компонентами управляющей системы УП может быть описан как набор процедур, соответствующих операциям технологического процесса, который определяет и порядок их выполнения.

Выбор формализованного языка, в наибольшей степени учитывающего особенности УП, является основной задачей начального этапа проектирования УС. Одно из характерных свойств УС, влияющих на структуру УП, заключается в наличии в нем множества блоков, часть которых может работать одновременно, взаимодействуя друг с другом. Поэтому формализованный язык должен прежде всего

обладать средствами для описания параллелизма. Широкое распространение различного типа систем вызвало в свое время развитие различных языков, предназначенных для их описания и моделирования. Часть из них является расширением известных языков программирования, другие основаны на использовании различных графовых моделей. Не имея возможности дать полный обзор этих языков и провести их сравнение, рассмотрим более подробно язык сетей Петри, получивший широкое распространение и обладающий средствами, которые позволяют в значительной степени учитывать структуру и характер функционирования УС.

Сети Петри разрабатывались специально для описания и моделирования систем, состоящих из отдельных взаимодействующих блоков. При этом допускается одновременная работа нескольких блоков, и данный язык отражает присущий такой системе параллелизм. Другой характерной особенностью является то, что каждый отдельный блок сам может быть целой системой, однако ее поведение возможно описать независимо от других блоков, за исключением точно определенного взаимодействия с этими блоками.

Именно такими свойствами обладают различные типы УС, так как ясно, что даже средство с моноблочной структурой состоит из отдельных взаимодействующих компонент, что необходимо учитывать при более детальном описании УС.

Основными понятиями, на которых базируется представление рассматриваемой системы сетью Петри, являются **события и условия**. **Событие** — это некоторое действие, происходящее в системе (консультационном процессе), т. е. определенный этап функционирования или управления, выполняемый в соответствующем блоке системы. Сеть Петри описывает структуру процесса управления, она определяет последовательность процедур этого процесса и не предназначена для моделирования функционирования или получения параметров управления. Для того чтобы в системе произошло определенное событие, необходимо выполнение соответствующих условий, множество которых задает состояние системы. Реализация события может привести к возникновению других условий и, следовательно, к изменению состояния системы. Такое представление системы, в качестве которой рассматриваем УС, полностью совпадает с рассмотренными ранее правилами его функционирования, что также говорит о целесообразности использования языка сетей Петри для описания УП, реализуемого в УС.

Наличие уже достаточно обширной в настоящее время литературы по сетям Петри, содержащей богатую библиографию, избавляет от необходимости подробного описания этого языка, его свойств и

областей применения. Поэтому дадим лишь основное определение сети Петри, введем обозначения, которые далее будут использоваться, и интерпретацию символов этого языка применительно к данной структуре УС.

В сетях Петри *события* и *условия* представляются абстрактными символами двух непересекающихся множеств — множества *переходов* и множества *позиций*, которые будем обозначать $T=\{t_0, t_1, \dots, t_r\}$ и $A=\{a_0, a_1, \dots, a_f\}$ соответственно. Переходы и позиции связаны между собой входной I и выходной O функциями. Функция I отображает переход $t_v (v = 0, 1, \dots, r)$ в множество позиций $I(t_v)$, которые называются *входными позициями перехода*. Функция O отображает переход t_v в множество позиций $O(t_v)$, называемых *выходными позициями перехода*. Позиция a_μ является входной позицией перехода t_v , если $a_\mu \in I(t_v)$, позиция a_μ — выходной позицией перехода, если $a_\mu \in O(t_v)$.

Таким образом, сеть Петри является четверкой $N=(A, T, I, O)$, где A и T — конечные непересекающиеся множества.

Например, простейшая сеть Петри с пятью позициями и переходами может быть задана в таком виде:

$$\begin{aligned} A &= \{a_0, a_1, a_2, a_3, a_4\}; \quad T = \{t_0, t_1, t_2, t_3, t_4\}; \\ I(t_0) &= a_0; \quad I(t_1) = a_1; \quad I(t_2) = a_2; \quad I(t_3) = a_3; \quad I(t_4) = a_4; \\ O(t_0) &= a_1; \quad O(t_1) = a_2; \quad O(t_2) = a_3; \quad O(t_3) = a_4. \end{aligned} \quad (5.1)$$

Ясно, что функции I и O могут быть заданы также в виде матриц, которые называют матрицами *следования* и *предшествования* соответственно, либо в виде одной объединенной матрицы.

Наряду с приведенным способом задания сети Петри в виде системы входных и выходных функций широко используется графическое представление, которое более наглядно, но менее компактно. Граф сети Петри имеет два типа вершин, один из них соответствует позициям, обозначаемым кружочками, другой — переходам, последние обозначаются вертикальными черточками. Направленные дуги графа соединяют вершины разных типов. Если позиция a_μ является входной для перехода t_v , то дуга направлена от a_μ к t_v , если a_μ является выходной позицией перехода t_v , то дуга направлена от t_v к a_μ . В общем случае в сети Петри допускается существование кратных дуг от одной вершины графа к другой.

Таким образом, граф $G=(V, W)$ сети Петри является ориентированным двудольным мультиграфом, где V —множество вершин; W —множество (точнее, комплект) направленных дуг. Очевидно, $V=A \cup T, A \cap T = \emptyset$.

Как отмечалось, позиция сети Петри соответствует условию. Выполнение условий отмечается маркировкой позиций, так что позиция сети Петри содержит метку, если сопоставленное с ней условие выполняется, и не содержит метки в противном случае. На графе сети Петри каждая метка позиции отмечается точкой в кружке этой позиции. Однако для сетей Петри общего вида допускается наличие более одной метки в позиции, и тогда требуется иная интерпретация меток. Некоторые авторы, например, вводят понятие емкости условия, поэтому далее рассмотрим некоторые конкретные примеры использования маркированных позиций.

Маркировка сети Петри может быть задана в виде f -вектора, который определяет число меток в каждой позиции сети Петри. Таким образом, маркированная сеть Петри задается пятеркой $N=(A, T, I, O, M_0)$, где M_0 — вектор начальной маркировки. На рис. 5.12 приведен граф маркированной сети Петри заданной (5.1), для $M_0=(1, 0, 0, 0, 0)$.

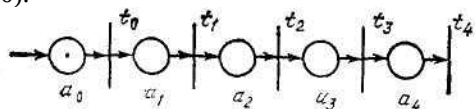


Рис.5.12.

При больших значениях чисел в векторе маркировки вместо такого количества точек в позиции указывается просто их число. Точно так же при большой кратности дуг между двумя вершинами графа сети Петри указывается только одна дуга, которая помечается числом, равным ее кратности. Заметим, что часто, когда используется графическое изображение, говорят о сети Петри, опуская слово «граф». Распределение меток в позициях сети Петри определяет порядок ее выполнения, который зависит от последовательности реализации переходов. Переход может быть реализован, только если он становится активным. Переход является активным, когда каждая из его входных позиций содержит число меток не меньшее, чем число дуг, соединяющих ее с этим переходом. Метки, число которых равно кратности дуги, называют *разрешающими*. В результате реализации активного перехода удаляются разрешающие метки из всех его входных позиций, и в каждую выходную позицию помещается по одной метке для каждой дуги. Таким образом, реализация перехода заменяет маркировку сети Петри M в общем случае на новую маркировку M' , которую называют непосредственно достижимой из M .

Из правил реализации переходов следует, что когда один переход имеет несколько выходных позиций, после его реализации все они

получат метки, т. е. произойдет распараллеливание процесса. После этого активизированные параллельные участки процесса могут выполняться независимо. Переход сети Петри, имеющий несколько входных позиций, может быть реализован, только если все эти позиции содержат метки, что соответствует выполнению всех соединяющихся в данном переходе параллельных участков процесса.

Достоинством языка сетей Петри является не только то, что он позволяет описывать параллельные процессы, но и имеет средства для задания конфликтных состояний, когда необходимо запретить одновременное развитие нескольких процессов. Если позиция сети Петри содержит q меток и является входной для $w > q$ переходов, то одновременно может быть реализовано не более q этих активизированных переходов. Вопрос о разрешении возникающего здесь конфликта, при котором одновременно несколько переходов готовы использовать метки одной и той же позиции, должен решаться дополнительными средствами.

Реализация переходов сети Петри осуществляется до тех пор, пока при очередной маркировке существует хотя бы один активный переход. Таким образом, при выполнении сети Петри возникают две связанные последовательности — реализуемых переходов и маркировок $M_0, M_1, M_2, \dots$. Маркировка сети Петри определяет ее состояние, так что можно говорить как о векторе маркировки, так и о состоянии сети Петри, которое соответствует состоянию задаваемого ей процесса.

Следует иметь в виду, что в первоначально разработанном в 1962 г. доктором Петри языке, т. е. в оригинальных сетях Петри, запрещалось существование кратных дуг между позициями и переходами и вектор маркировки мог содержать лишь 0 и 1. Кроме того, реализация активного перехода разрешается только в том случае, если ни одна из его выходных позиций не содержит меток. Очевидно, тогда число меток в любой позиции не может быть больше 1. Такие позиции называют *безопасными*, а сеть Петри, все позиции которой безопасны, является *безопасной*. Она имеет простые правила выполнения и конечное число состояний, равное 2^f при f позициях, однако ее описательные возможности весьма ограничены. Поэтому при необходимости отразить более сложные явления, например очередь заявок или число клиентов, ожидающих обслуживания, приходится отказываться от безопасных сетей Петри и допускать наличие нескольких меток в позиции. Если число меток в позиции не превышает некоторое значение k , то позицию называют *k-безопасной* или *ограниченной*. Для $k' \geq k$ *k-безопасная* позиция является и *k'-безопасной*. Хотя для разных позиций величина k может быть

различной, всегда можно найти такое число $k_{\text{макс}}$, для которого все позиции $k_{\text{макс}}$ — безопасны. Это говорит о том, что число меток в позиции ограничено, т. е. сама позиция является ограниченной. Сеть Петри называют *ограниченной*, если все ее позиции ограничены. Для описания многих процессов приходится использовать не только позиции, содержащие множество меток, но и допускать наличие кратных дуг между позициями и переходами.

В результате этого и сформировалась рассмотренная общая модель, в соответствии с которой сеть Петри представляется ориентированным двудольным мультиграфом с произвольной начальной маркировкой позиций и кратными дугами. Однако и эта модель является ограниченной в том смысле, что она представляет только структуру управления, но не отражает ее связь с ОУ. Поэтому потребовались различные расширения сети Петри, в которых поразному можно моделировать взаимодействие сети Петри с управляемым процессом. Наиболее удачным можно считать подход, когда в сеть Петри вводится специальное множество позиций, каждая из которых представляет отдельный входной или выходной символ, характеризующий рассматриваемое взаимодействие. В позицию, соответствующую входному символу, метка помещается извне, а не в результате реализации переходов сети Петри. В свою очередь метка, появившаяся при выполнении сети Петри в позиции, соответствующей выходному символу, удаляется при выполнении определенных операций в ОУ.

Другим существенным ограничением оригинальных сетей Петри является то, что реализация перехода рассматривается как мгновенное событие, занимающее нулевое время. Моделируемое таким переходом событие называют *примитивным*. Однако в реальном мире большинство событий занимают некоторое ненулевое время, т. е. они не являются примитивными и, строго говоря, не могут моделироваться переходами сети Петри. В связи с этим вводятся различные дополнения для того, чтобы моделировать сетями Петри реальные системы с непримитивными событиями. Например, непримитивное событие может быть представлено в виде двух примитивных событий, соответствующих началу и концу непримитивного события, и условия, которое принимает истинное значение, когда происходит непримитивное событие. Другое предложение состоит в том, чтобы непримитивное событие изображать на графе сети Петри не вертикальной чертой, как примитивный переход, а прямоугольником. Но чаще такое обозначение используется в тех случаях, когда моделируются сложные иерархические системы. Тогда прямоугольником обозначается не отдельный переход, а целый

уровень иерархии, который может быть представлен своей сетью Петри. При описании сетями Петри реальных систем, несмотря на введенное первоначально ограничение, обычно все же предполагается, что для реализации каждого перехода требуется некоторое время и отдельные события могут пересекаться во времени, но никаких ограничений на относительные времена реализации переходов не накладывается. В этом смысле говорят об асинхронности, одновременности и недетерминированности при выполнении сети Петри.

На основе предложенной первоначально оригинальной модели развилась целая теория сетей Петри, в которой исследуются их модификации, различающиеся как свойствами используемых переходов, дуг, позиций и их маркировкой, так и правилами выполнения сети Петри.

Оригинальную сеть Петри называют иногда *сетью позиция — переход*. Если в такой сети каждый переход имеет одну входную и одну выходную позицию, сеть описывает последовательный процесс и является *автоматной*. Сеть Петри, в которой каждая позиция имеет один входной и один выходной переходы, является *маркированным графом*, который задает *параллельно-последовательный процесс* без разветвлений. Если выполняются свойства автоматной сети и маркированного графа, то такая сеть Петри описывает *последовательный линейный процесс*, в котором нет ни *параллельных участков*, ни *альтернативных ветвей*.

Следует отметить, что если переходы имеют несколько входных и выходных позиций, то сеть Петри описывает не только параллелизм, но и ожидание. Необходимым условием реализации перехода является лишь наличие разрешающих меток в его входных позициях.

Более интересными и необходимыми для описания реальных систем являются *сети Петри с предикатами* на переходах. В такой сети каждому переходу ставится в соответствие предикат, определяющий отношения между некоторыми переменными. Активизированный переход может быть реализован только при истинном значении сопоставленного с ним предиката. Дальнейшее расширение описательных возможностей сети Петри состоит в том, что каждому переходу сопоставляется некоторое действие, или операция. В зависимости от свойств рассматриваемой систем или процесса каждая операция может иметь разный содержательный смысл. В простейшем случае используются операции над множеством переменных, от которых зависит значение предиката. В таких сетях Петри типа *предикат — действие* переход реализуется, если он активизирован и сопоставленный с ним предикат имеет истинное

значение, при этом неделимым образом выполняется соответствующая переходу операция и создается новая маркировка сети Петри.

Таким образом, сеть Петри оказалось возможным описывать пороцессы и системы, которые выполняют какие-то действия. Очевидно, эти действия занимают некоторое время, которое желательно ввести в формализованное описание пороцесса. Для этого предлагается два подхода. Первый из них используется во временных сетях Петри, где каждому переходу ставится в соответствие определенный интервал времени, равный длительности его реализации, т. е. продолжительности сопоставленной с ним операции. Когда переход становится активным, он сразу же переходит в стадию реализации, захватывая метки входных позиций и запрещая тем самым реализацию других конфликтующих с ним переходов. По истечении времени реализации перехода его выходные позиции получают метки.

Второй подход состоит в том, что каждому переходу ставится в соответствие пара величин a и b таких, что $0 \leq a \leq b$. Если переход t активизирован в момент времени q , то он не может быть реализован ранее, чем в момент $(q+a)$, и должен быть реализован до или в момент $(q+b)$, если только условия его активизации не изменились из-за реализации других переходов. Такие сети Петри называют *тайм-аутными*, их используют для описания и проверки протоколов вычислительных сетей, в которых применяется механизм тайм-аута, и для описания систем со свойством восстановления. *Временные сети Петри в большей степени предназначены для оценки времени выполнения исследуемого пороцесса.*

В рассмотренных сетях Петри все метки, содержащиеся в позициях, являются одинаковыми, поэтому для того, чтобы задать разные типы проблемных условий, проблемных объектов или проблемных ресурсов, которые характеризуют состояние системы, необходимо соответственно увеличивать число позиций сети. Избежать этого позволяют так называемые *помеченные сети Петри*, или *сети с раскрашенными позициями*. В таких сетях каждой метке сопоставляется определенный признак (цвет), так что одна позиция сети Петри может содержать метки разных цветов. Теперь для активизации перехода его входная позиция должна содержать метки с определенными признаками, которыми помечается дуга, направленная от позиции к переходу.

Обобщением сетей предикат—действие с раскрашенными позициями являются *численные сети Петри*. В этих сетях метки позиций могут иметь любую природу и величину, условия активизации и результат реализации перехода независимы, при реализации переходов изменяется как маркировка его входных и выходных

позиций, так и содержимое памяти данных. Каждая входная дуга помечается условием активизации и условием реализации перехода, которые могут не совпадать, т. е. не все метки, необходимые для активизации перехода, удаляются из входной позиции после его реализации. На каждой выходной дуге указывается, сколько меток и с какими признаками помещаются в выходную позицию в результате реализации перехода. Переходу может сопоставляться некоторый предикат и операция над данными.

Очевидно, такие сети обладают большими описательными возможностями, но слишком сложны для анализа и преобразования реальных процессов.

Другая разновидность сетей Петри использует *дуги разных типов*. При этом различают *простые дуги*, к которым относятся активизирующая, сдерживающая, входная, выходная, и *составные дуги* — это активизирующая входная и сдерживающая выходная. *Активизирующая дуга* приводит к реализации перехода, когда позиция заполнена, и запрещает его, когда позиция пуста.

Сдерживающая дуга запрещает реализацию перехода, если позиция содержит метку, и активизирует его, если позиция пуста.

Входная дуга удаляет метку из позиции при реализации перехода; если позиция была пуста, она остается неизменной.

Выходная дуга помещает метку в позицию после реализации перехода; если позиция была заполнена, она остается неизменной.

Активизирующая входная дуга является обычной дугой оригинальной сети Петри: для активизации перехода в позиции должна быть метка, которая удаляется после его реализации.

Сдерживающая выходная дуга также соответствует правилам выполнения оригинальной сети Петри: для реализации активного перехода необходимо, чтобы его выходная позиция была пуста, она заполняется меткой после реализации перехода.

Введение сдерживающих дуг позволяет осуществлять проверку на нуль значения какой-либо переменной, что существенно расширяет описательные возможности сетей Петри и в этом смысле приближает их к машинам Тьюринга. Однако большинство вопросов анализа самих сетей Петри с сдерживающими дугами для неограниченных позиций становится неразрешимым.

Таким образом, выбор из всего существующего множества разновидностей сетей Петри той модели, которая наиболее адекватно описывает рассматриваемую систему и, обладая необходимыми описательными возможностями, сама может быть проанализирована, является достаточно сложной задачей.

Сравнение разных модификаций сети Петри позволило выделить те их свойства, которые необходимы для описания облачных управляющих процессов (ОУП) и разработать некоторую обобщенную модель для описания УП общего типа. Основной задачей при этом было отразить в исходном описании структуру и свойства УС и максимально сохранить простые правила выполнения и анализа оригинальной сети Петри.

5.2.3.2. Облачные управляющие процессы и их формализованное описание

Описание облачного управляющего процесса (ОУП) , реализуемого в УС, может быть осуществлено лишь после того, как определено **назначение УС и его структура**, т. е. **число и состав функциональных блоков (ФБ), выполняемые ими функции и их взаимодействие**. При этом полнота получаемого описания зависит как от имеющихся в распоряжении проектировщика сведений об ОУП, так и от цели, которую он преследует, составляя описание процесса и проблемы. Для решения некоторых задач достаточно знать только общую структуру процесса, для других требуется задавать взаимодействие процедур процесса с ресурсами. Составление полного описания, которое необходимо иметь при построении УС, обычно является сложной итеративной процедурой, позволяющей постепенно выявлять и уточнять характер взаимодействия функциональных блоков (ФБ) УС.

Структура УС, содержащиеся в нем ФБ и, порядок их работы определяются свойствами ОУП, которые, в свою очередь, зависят от характеристик технологического процесса. Очевидно, используемый формализованный язык должен обладать средствами для описания различных типов ОУП.

Простейшим можно считать последовательный линейный процесс. Такой процесс состоит из некоторого числа процедур $A_0, A_1, \dots, A_k$, последовательность выполнения которых определяется технологическим процессом и не зависит от значений и изменений каких-либо параметров управляемой системы. УС, реализующее линейный процесс, содержит оператонные функциональные блоки (ОФБ), последовательность включения которых задает ОУП.

Считая выполнение процедуры некоторым событием и сопоставив поэтому каждой процедуре переход сети Петри, линейный ОУП можно описать оригинальной сетью Петри, которая будет содержать $t_0, t_1, \dots, t_k$ переходов и $a_0, a_1, \dots, a_f$ позиций. Появление метки в позиции a_0 , соответствующее активизации процесса, вызывает последовательную реализацию всех переходов сети Петри. Хотя такое описание ОУП возможно, оно мало отражает действительный характер

функционирования УС, поскольку в оригинальной сети Петри все переходы являются примитивными, т. е. предполагается, что они реализуются мгновенно. В действительности каждая процедура имеет определенную длительность, так что можно выделить отдельные ее фазы: *начало, выполнение и окончание процедуры*. Это соответствует *включению, работе и отключению ОФБ*, в котором выполняется процедура.

Такую процедуру A_i можно представить подсетью Петри (рис. 5.13,а), содержащей переходы t_{n_i} и t_{k_i} и дополнительные позиции z_i и w_i , которые в отличие от *внутренних* позиций назовем *внешними*, так как их маркировка зависит не только от состояния сети Петри, но и от внешних событий.

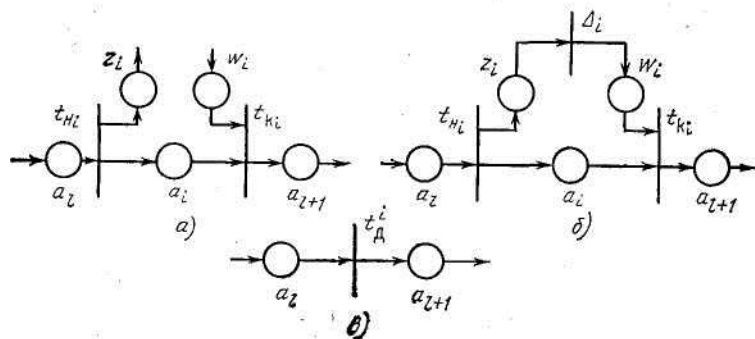


Рис. 5.13.

В результате реализации перехода t_{n_i} , который определяет начало выполнения процедуры A_i , появляются метки в его внешней z_i и внутренней a_i выходных позициях. Первая из них соответствует включению *ОФБ*, вторая — тому, что происходит событие, состоящее в выполнении процедуры A_i . Окончание работы *ОФБ* отмечается появлением метки во внешней входной позиции перехода t_{k_i} . Этот переход реализуется, в результате чего появляется метка в его внутренней выходной позиции, означающая завершение процедуры A_i . Следует отметить, что при многопрограммном принципе реализации ОУП такое описание взаимодействия с *ОФБ* не только наглядно и удобно, но и необходимо, так как позволит разделить переходы t_{n_i} и t_{k_i} .

Как уже отмечалось, ОФБ может иметь местное управление, что позволяет ему после получения включающего сигнала работать автономно. В это время могут выполняться процедуры других процессов, которым сопоставлены свои переходы. При этом переходы t_{n_i} и t_{k_i} уже не являются переходами, связанными отношением непосредственного следования. Использование внешних выходных и входных позиций позволяет отразить специфику многопрограммного управления.

Однако разделение позиций на внутренние и внешние достаточно условно, поскольку, расширив рамки рассматриваемого явления, можно перевести часть внешних позиций во внутренние. Так, в нашем случае описания взаимодействия с ОФБ работу этого блока можно представить как непримитивное событие и промоделировать непримитивным переходом Δ_i . Если теперь включить этот переход в рассматриваемую сеть Петри, то внешние позиции z_i и w_i станут внутренними, так как заполнение их метками и удаление меток будет связано только с реализацией переходов этой сети Петри (рис. 5.13, б). В тех случаях, когда выполнение процедуры можно рассматривать как неделимое событие, для упрощения графа сети Петри ее фрагмент, содержащий переходы t_{n_i} , Δ_i , t_{k_i} и позиции z_i , a_i , w_i , целесообразно представить одним переходом t_d , отметив его индексом сопоставленной с ним процедуры t_d^i (рис. 5.13, в). Такой отмеченный переход будем называть *длительным*. Следует иметь в виду, что в отличие от примитивного для реализации длительного перехода требуется некоторое время, в течение которого уже нет метки в его входной позиции, но еще не появилась метка в выходной позиции. Никаких ограничений на длительность реализации перехода не накладывается, известно только, что это время конечно. При необходимости можно вновь заменить длительный переход подсетью Петри (рис. 5.13, б).

Реализация длительных переходов связана с использованием соответствующих ОФБ, которые являются ресурсами, необходимыми для выполнения процесса. Кроме ОФБ в состав УС могут входить элементы памяти и регистры для хранения промежуточных данных, вентили, шины передачи данных и управляющих сигналов и. др. Все аппаратурно-программные средства, необходимые для выполнения процедур облачного процесса, будем называть *функциональными ресурсами* (ФР). Функциональный ресурс назовем *собственным* ресурсом процедуры A_i , если он используется только этой процедурой и не доступен для других. Функциональный ресурс назовем *разделяемым*, если он может потребоваться нескольким процедурам.

При описании облачного процесса будем рассматривать только разделяемые ФР, так как именно они могут влиять на ход развития облачного процесса.

Пусть для выполнения процедур ОУП необходимо l функциональных разделяемых ресурсов $C_1, C_2, \dots, C_l$. Если, имеется q экземпляров одного и того же ФР (например, q одинаковых ОФБ), то этот ресурс имеет кратность $q(C_j^q)$ и может использоваться одновременно $\alpha \leq q$ процедурами. Когда $q=1$, ресурс имеет два состояния: «свободен» и «занят». В общем случае ресурс имеет $q+1$ состояние.

Ресурсы, которые разделяются только процедурами рассматриваемого облачного процесса, назовем *внутренними* или *собственными* ресурсами этого процесса. Далее будем считать, что все ФР являются внутренними.

Для выполнения процедур линейного облачного процесса необходимы только функциональные ресурсы, и каждую процедуру A_i такого облачного процесса можно охарактеризовать следующими *тремя множествами*: $\{C_v^i\}$ — множество ФР, которыми процедура A_i уже владеет; $\{C_3^i\}$ — множество ФР, которые процедура A_i еще запрашивает; $\{C_o^i\}$ — множество ФР, которые освобождаются после выполнения A_i .

Если, например, для процедуры A_i $\{C_v^i\} = C_1$, $\{C_3^i\} = C_3, C_4$ и $\{C_o^i\} = C_1, C_4$, т. е. $A_i(\{C_1\}, \{C_3, C_4\}, \{C_1, C_4\})$, это означает, что при выполнении процедуры A_i ей уже предоставлен ресурс C_1 , еще необходимы ресурсы C_3 и C_4 , а в результате ее выполнения ресурсы C_1 и C_4 освобождаются.

Будем считать, что процедура не запрашивает повторно тот ресурс, который ей предоставлен в результате выполнения других процедур, поэтому $\{C_3^i\} \cap \{C_v^i\} = \emptyset$. В частном случае, когда все необходимые ресурсы запрашиваются данной процедурой и сразу освобождаются, $\{C_v^i\} = \emptyset$ и $\{C_3^i\} = \{C_o^i\}$. Очевидно, во время выполнения процедура владеет всеми запрошенными ресурсами.

Пусть перед началом выполнения облачного процесса и после его окончания все ресурсы свободны. Функциональному ресурсу C_j поставим в соответствие позицию c_j сети Петри и назовем ее *ресурсной позицией*. Если q_j — кратность ФР C_j , позиция c_j в начальной маркировке содержит q_j меток. Позиция c_j должна быть входной позицией длительного перехода t_d^i если $C_j \in \{C_3^i\}$, и выходной позицией этого перехода, если $C_j \in \{C_o^i\}$. Ясно, что когда ресурс C_j занимается только на время выполнения процедуры A_i то позиция c_j является одновременно входной и выходной позицией перехода t_d^i .

Таким образом, сеть Петри, описывающая линейный последовательный облачный процесс, содержит длительные переходы t_d^i , основные внутренние a_u и ресурсные внутренние c_j позиции. Будем считать, что заданное количество и распределение ресурсов позволяют последовательному облачному процессу развиваться от начальной процедуры A_0 до конечной A_k .

На рис. 5.14 приведен пример сети Петри для облачного процесса, состоящего из пяти последовательно выполняемых процедур A_0, A_4, A_1, A_2, A_3 при следующем распределении трех ФР C_1, C_2, C_3 : $A_0(\{C_2\}, \{-\}, \{-\})$; $A_2(\{C_2\}, \{C_1\}, \{C_2\})$; $A_3(\{C_1\}, \{C_3\}, \{C_1, C_3\})$; $A_4(\{-\}, \{C_2, C_3\}, \{C_3\})$.

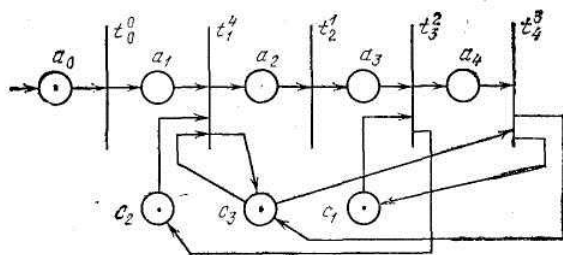


Рис. 5.14.

Рассмотрим особенности описания параллельного линейного облачного процесса. Такой ОУП может иметь сложную параллельно-последовательную структуру, определяемую управляемым технологическим процессом. Сеть Петри, задающая параллельный облачный процесс, кроме длительных переходов, соответствующих процедурам, содержит примитивные переходы для распараллеливания облачного процесса и объединения его параллельных участков. Эти переходы будем называть соответственно *переходами распараллеливания* t_R и *соединения* t_S , число таких переходов зависит от структуры ОУП.

Участок облачного процесса, соответствующий части сети Петри, заключенной между двумя последовательными переходами распараллеливания и (или) соединения будем называть *элементарным подпроцессом*. Такие подпроцессы могут быть параллельными либо последовательными, но в пределах одного элементарного подпроцесса все процедуры выполняются последовательно и его подсеть не содержит переходов распараллеливания и соединения.

Функциональные ресурсы, которые используются только процедурами одного и того же элементарного подпроцесса, будем называть *собственными* ФР этого подпроцесса. При описании

облачного процесса учитывают лишь ресурсы, разделяемые разными параллельными подпроцессами.

На рис. 5.15 приведен пример сети Петри, задающей параллельный линейный облачный процесс при следующем распределении ФР, используемых процедурами параллельных подпроцессов:

$A_1(\{C_1\}, \{C_3\}, \{-\})$; $A_2(\{-\}, \{C_3\}, \{-\})$; $A_3(\{C_3\}, \{C_2\}, \{C_3\})$;
 $A_4(\{C_1, C_2\}, \{-\}, \{C_1\})$; $A_5(\{-\}, \{C_1\}, \{-\})$; $A_6(\{C_1, C_3\}, \{-\}, \{C_3\})$;
 $A_7(\{C_2\}, \{-\}, \{C_2\})$; $A_8(\{-\}, \{C_2\}, \{-\})$; $A_9(\{C_2\}, \{C_1\}, \{-\})$;
 $A_{10}(\{C_1, C_2\}, \{-\}, \{C_2\})$; $A_{11}(\{C_1\}, \{-\}, \{C_1\})$.

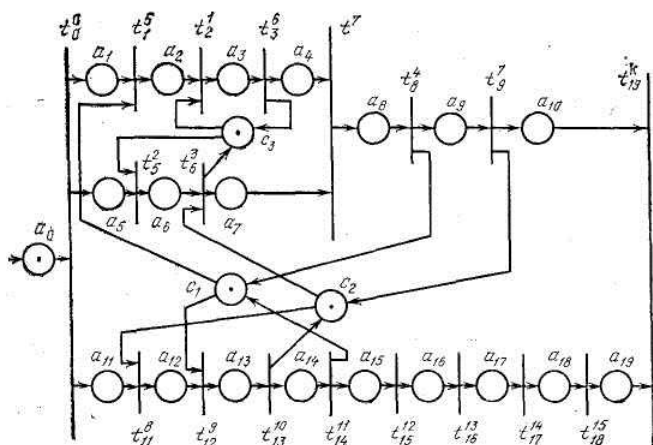


Рис. 5.15.

Заметим, что разделение ресурсов процедурами независимых параллельных облачных процессов превращает их в один облачный процесс с параллельными подпроцессами, взаимодействующими через эти ресурсы. Сеть Петри (рис. 5.16) иллюстрирует этот случай.

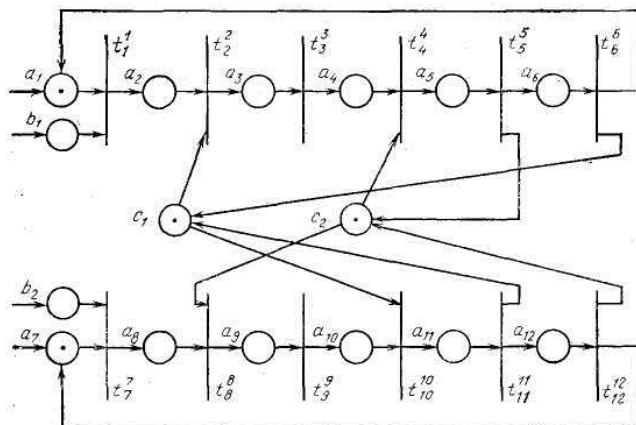


Рис. 5.16.

Разветвленный облачный процесс в отличие от линейного содержит не одну, а несколько последовательностей выполнения процедур, каждая из которых соответствует определенному набору значений логических условий, проверяемых в ЛФБ УС. Следует четко различать альтернативные ветви последовательного облачного процесса и параллельные участки (подпроцессы) параллельного облачного процесса, которые могут развиваться одновременно.

Если для сети Петри параллельного облачного процесса характерно наличие переходов распараллеливания и соединения, имеющих несколько выходных и входных позиций соответственно, то для сети Петри разветвленного облачного процесса характерно наличие *позиций альтернативного разветвления* и *альтернативного соединения*. Позиция альтернативного разветвления имеет несколько переходов-последователей, но одновременно может быть реализован только один из них. Позиция альтернативного соединения имеет несколько переходов-предшественников и получает метку в результате реализации одного из них. Следует отметить, что позиции альтернативного соединения имеются и в сети Петри облачного процесса, содержащего циклы.

В разветвленном облачном процессе конфликт, соответствующий позиции альтернативного разветвления, разрешается путем сопоставления каждому из конфликтующих переходов определенного набора значений логических условий. Очевидно, эти наборы должны быть ортогональны, чтобы при выполнении облачного процесса активизировалась только одна из альтернативных ветвей.

Как уже отмечалось, проверка значений логических условий осуществляется в ЛФБ УС. По аналогии с функциональными ресурсами ресурсы системы, которые необходимы для определения последовательности процедур облачного процесса, назовем *логическими* ЛР. В отличие от ФР для ЛР нужно знать не только его состояние, но и значение проверяемого в нем логического условия (ЛУ), которое может изменяться либо процедурами рассматриваемого ОУП, либо внешними событиями.

Пусть для выполнения облачного процесса необходимо m ЛР $D_1, D_2, \dots, D_m$ таких, что в ЛР D_s проверяется значение логического условия p_s . Если значение логического условия p_s изменяется только процедурами КП, ЛР D_s будем называть *внутренним* ЛР. Рассмотрим вначале случай, когда все ЛР являются внутренними.

Каждую процедуру A_i будем характеризовать двумя множествами ЛУ:

$$\{P_1^i\} \text{ и } \{P_2^i\} : A_i(\{P_1^i\}, \{P_2^i\}),$$

где $\{P_1^i\}$ — номера ЛУ и их значения, необходимые для выполнения процедуры A_i ; $\{P_2^i\}$ — номера ЛУ и их значения, которые они приобретают в результате выполнения процедуры A_i .

Если, например, $A_i(\{\bar{p}_1, p_2\}, \{\bar{p}_2, p_3\})$, то это означает, что для выполнения процедуры A_i необходимо, чтобы логическое условие p_1 имело нулевое, а логическое условие p_2 — единичное значение. После выполнения A_i значение логического условия p_2 изменится и станет нулевым, а логическое условие p_3 примет единичное значение, хотя само выполнение процедуры A_i не зависит от значения этого логического условия.

Если логическое условие p_s входит в множество $\{P_2^i\}$, т. е. его значение изменяется процедурой A_i то эта процедура монопольно занимает ЛР D_s и во время ее выполнения он находится в состоянии *занято*. После завершения процедуры все ЛР освобождаются, в отличие от ФР, которые могут быть заняты при выполнении нескольких последовательных процедур.

Если логическое условие p_s входит только в множество $\{P_1^i\}$, т. е. его значение просто используется для определения активной ветви облачного процесса, то ЛР D_s занимается немонопольно. Допускается одновременно несколько таких обращений к одному ЛР, поскольку его состояние и значение проверяемого им логического условия при этом не изменяются. Это соответствует тому, что для выполнения процедуры A_i необходимо определенное значение логического условия p_s , которое не изменяется этой процедурой.

Если логическое условие p_s входит только в множество $\{P_1^i\}$, процедура A_i выполняется независимо от значений логического условия p_s , но после этого p_s принимает определенное значение, которое и указывается в $\{P_2^i\}$.

Очевидно, логическое условие p_s может входить как в $\{P_1^i\}$, так и в $\{P_2^i\}$. Это соответствует тому, что выполнение процедуры A_i зависит от значения p_s и, в свою очередь, влияет на это значение, т. е. монопольно занимает ЛР D_s .

Таким образом, для описания разветвленного облачного процесса необходимо иметь средства моделирования существующих видов взаимодействия облачного процесса с ЛР на языке сетей Петри. Прежде всего нужно представить каждый ЛР в терминах этого языка так, чтобы задать и его состояние, и значение проверяемого им ЛУ. Для этого ЛР D_s поставим в соответствие не одну, как в случае ФР, а три позиции сети Петри d_s , d_s^1 , d_s^0 . Позиция d_s содержит метку, если ресурс D_s свободен, т. е. ни одна из процедур облачного процесса не владеет им монопольно. Позиция $d_s^1(d_s^0)$ содержит метку, если логическое условие p_s , проверяемое в ЛР D_s , имеет единичное (нулевое) значение.

Самым простым для описания сетью Петри является случай, когда процедура A_i изменяет значение того ЛУ, от которого и зависит ее выполнение. Например, если $A_i(\{p_s\}, \{\bar{p}_s\})$ и $A_j(\{\bar{p}_s\}, \{p_s\})_j$ то входными позициями перехода $t_d^i(t_d^j)$ кроме основной внутренней позиции должны быть позиции d_s и d_s^1 (d_s и d_s^0). Выходными позициями перехода $t_d^i(t_d^j)$ будут кроме внутренней позиции позиции d_s и d_s^0 (d_s и d_s^1) (рис. 5.17, а), что соответствует освобождению логического ресурса D_s и изменению значения логического условия p_s .

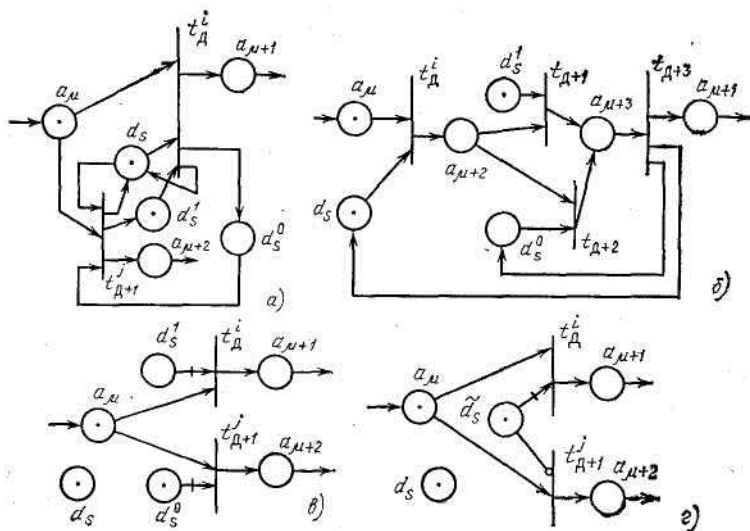


Рис. 5.17.

Другой случай, когда выполнение процедуры A_i не зависит от значения логического условия p_s , но в результате ее выполнения p_s принимает определенное значение, также может быть промоделирован существующими средствами сети Петри. Поскольку ЛР монопольно занимается на время выполнения A_i , позиция d_s является входной позицией перехода t_A^i . Так как перед выполнением A_i значение ЛУ p_s не проверяется, необходимо удалить метку из той позиции (d_s^0 или d_s^1), в которой она находилась, а затем поместить ее в d_s^0 (d_s^1), если в результате выполнения A_i p_s принимает нулевое (единичное) значение, одновременно освободив ЛР D_s (рис. 3.17, б).

Наконец, третий случай, когда для выполнения процедуры A_i необходимо определенное значение ЛУ p_s , но это значение не меняется, не удастся промоделировать существующими в сети Петри средствами. Кажущийся естественным способ сделать позицию d_s^1 (или d_s^0) входной и выходной позицией перехода t_A^i не соответствует такому немонопольному занятию ЛР, когда одновременно несколько облачных процессов могут считывать значение одного и того же ЛУ, не изменяя его.

Для описания этого взаимодействия введем в сеть Петри новый тип дуг, которые назовем *неизменяющими*. Если переход t_i соединен с входной позицией a_μ неизменяющей дугой, то для его реализации необходимо наличие метки в позиции a_μ , которая не удаляется из a_μ

при реализации t_v . На рисунках неизменяющую дугу будем обозначать перечеркнутой стрелкой. Теперь нетрудно описать возможность немонотонного занятия ЛР, которое не приводит к изменению значения проверяемого им ЛУ. Если $A_i(\{p_s\}, \{—\})$, то позиция d_s^1 должна быть соединена с t_d^i неизменяющей дугой. Когда

$A_j(\{\bar{p}_s\}, \{—\})$, позиция d_s^0 является входной для t_d^i и соединяется с ним неизменяющей дугой (рис. 5.17, в). Заметим, что в этом случае при описании последовательного облачного процесса метка позиции d_s не используется.

Очевидно, если $A_i(\{—\}, \{—\})$, то процедура A_i не взаимодействует с ЛР и переход t_d^i может быть реализован независимо от маркировки позиций логических ресурсов.

Рассмотренный способ представления ЛР тремя позициями сети Петри хотя позволяет промоделировать все случаи взаимодействия облачных процессов с ресурсами, однако является достаточно громоздким, так как требует введения большого числа дополнительных позиций. Это число можно сократить, если применить так называемые *сдерживающие*, или *тормозящие*, дуги. Эти дуги введены специально для того, чтобы использовать информацию об отсутствии метки в позиции. Если переход t_v соединен тормозящей дугой с входной позицией a_u , то он может быть реализован, только когда эта позиция не содержит метку. При этом ЛР D_s можно задать двумя позициями: d_s и $\bar{d}_s$, последняя содержит метку, когда логическое условие p_s имеет единичное значение, и не содержит ее при нулевом значении этого ЛУ.

На графе сети Петри сдерживающая дуга заканчивается не стрелкой, а маленьким кружком. При использовании таких дуг взаимодействие $A_i(\{p_s\}, \{—\})$ с логическим ресурсом D_s , показанное на рис. 5.17, в, будет моделироваться сетью Петри, приведенной на рис. 5.17, г.

Следует отметить, что позиции функциональных ресурсов и позиции состояний логических ресурсов могут не вводиться в описание последовательного облачного процесса, тогда как маркировка позиций значений ЛУ может существенно влиять на развитие процесса.

На рис. 5.18 показана сеть Петри разветвленного последовательного облачного процесса, все процедуры которого используют собственные ФР, выполнение процедур $A_1, A_3, A_4, A_5, A_6, A_7$ зависит от значений логических условий p_1, p_2 и процедуры A_1, A_3, A_7 меняют значения логических условий: $A_1(\{p_1\}, \{\bar{p}_1\})$; $A_3(\{p_2\}, \{\bar{p}_2\})$; $A_4(\{\bar{p}_1\}, \{—\})$;

$A_5(\{p_1\}, \{-\}); A_6(\{\bar{p}_1\}, \{-\}); A_7(\{\bar{p}_2\}, \{p_2\})$. Для упрощения графа сети Петри не введены позиции состояний $\bar{d}_1$ и $\bar{d}_2$.

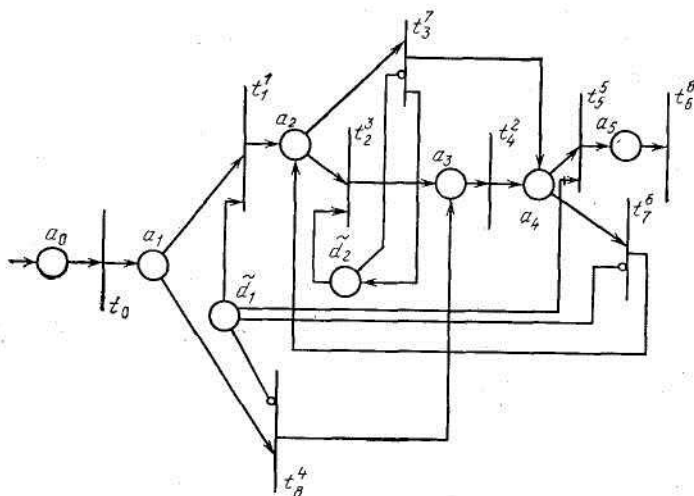


Рис. 5.18.

В общем случае облачный процесс содержит как альтернативные ветви, так и параллельные участки; пример сети Петри такого процесса приведен на рис. 5.19.

Если все логические и функциональные ресурсы ОУП являются внутренними, процесс можно назвать автономным, поскольку его развитие полностью определяется начальным состоянием, т. е. начальной маркировкой его сети Петри.

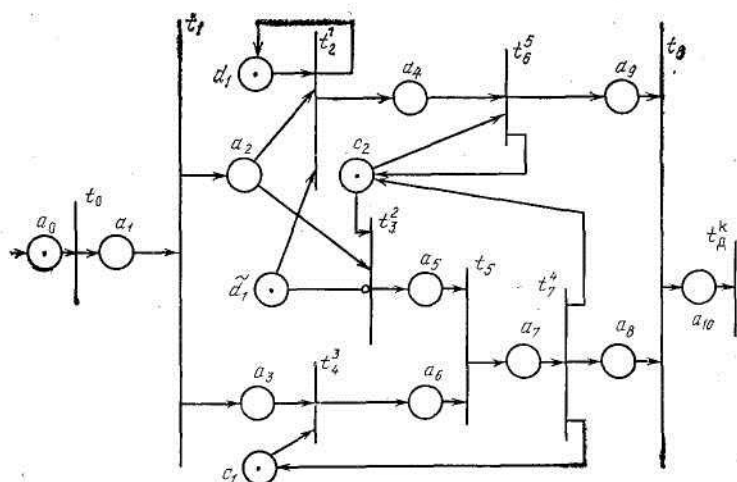


Рис. 5.19.

5.2.3.3. Обобщенная сеть Петри для описания неавтономного облачного управляющего процесса

Для автономного облачного процесса характерно то, что все ФР используются только процедурами данного процесса и значения всех ЛУ изменяются этими процедурами, что отражается в описании облачного процесса. Однако во многих случаях значения ЛУ, влияющих на развитие облачного процесса, зависят от состояния ОУ или от состояний других облачных процессов, т. е. определяются некоторыми внешними событиями. Такие ЛР являются внешними, их состояния и значения проверяемых ими ЛУ зависят не только от процедур рассматриваемого ОУП. Для описания такого процесса каждому внешнему логическому условию p_u сопоставим внешнюю позицию h_u , которая содержит метку, если $p_u=1$, и не содержит ее при $p_u=0$. В описание процедур могут входить как внутренние, так и внешние ЛУ; очевидно, последние входят только в множество $\{P_i\}$. Если процедура A_i выполняется при единичном (нулевом) значении логического условия p_u , то внешняя позиция h_u соединяется с переходом t_d^i неизменяющей (сдерживающей) дугой. Позиция состояния внешнего ЛР не включается в сеть Петри КП, так как ее маркировка не может быть полностью определена развитием этого процесса. Все процедуры, выполнение которых зависит от значения внешнего ЛУ, только считывают его значение, следовательно, может быть одновременно

несколько обращений к внешнему ЛР, а значение ЛУ должно оставаться неизменным на все время выполнения этих процедур. Облачный управляющий процесс, который использует наряду с внутренними и внешние ресурсы, является неавтономным. Развитие такого процесса зависит не только от начальной маркировки внутренних позиций сети Петри, но и от текущей маркировки внешних входных позиций. Поскольку маркировка этих позиций не изменяется в результате выполнения процедур процесса, для упрощения графа сети Петри внешние входные позиции можно заменить предикатами, зависящих от внешних ЛУ. Переход t_d^i помечается тем предиктом, истинное значение которого необходимо для выполнения процедуры A_i . Таким образом, сеть Петри неавтономного процесса содержит позиции внутренних ресурсов и помеченные предикатами переходы. Для простоты будем считать, что внешними могут быть только ЛР, а все ФР являются внутренними

Следует отметить еще один случай задания взаимодействия процедур процесса с ЛР когда не определено конкретное влияние выполнения процедуры на значение логического условия p_s , но известно, что такая зависимость существует. Обычно эта ситуация возникает, если ЛР является сложным ЛФБ, имеющим множество внутренних состояний, которые изменяются под воздействием процедуры A_i , но не все эти изменения приводят к изменению значения проверяемого ЛУ. Без учета функционирования ЛР приходится считать, что процедура A_i влияет на значение ЛУ т. е. всегда монополюльно занимает ЛР. Такое возможное изменение значения логического условия уровня p_s будем отмечать его безразличным ($\bar{p}_s$) значением в множестве $\{P_2^i\}$. Позицию состояния логического ресурса D_s в этом случае необходимо вводить в описание параллельного процесса, чтобы запретить одновременное выполнение процедур, использующих ресурс D_s . Метка из позиции d_s удаляется на время реализации перехода t_d^i . Позиция $\bar{d}_s$ аналогична внешней позиции, так как маркировка ее не может быть определена состоянием процесса.

На рис. 5.20, а показан пример сети Петри консультационного процесса, все процедуры которого используют собственные ФР: ЛР D_1 является внутренним.

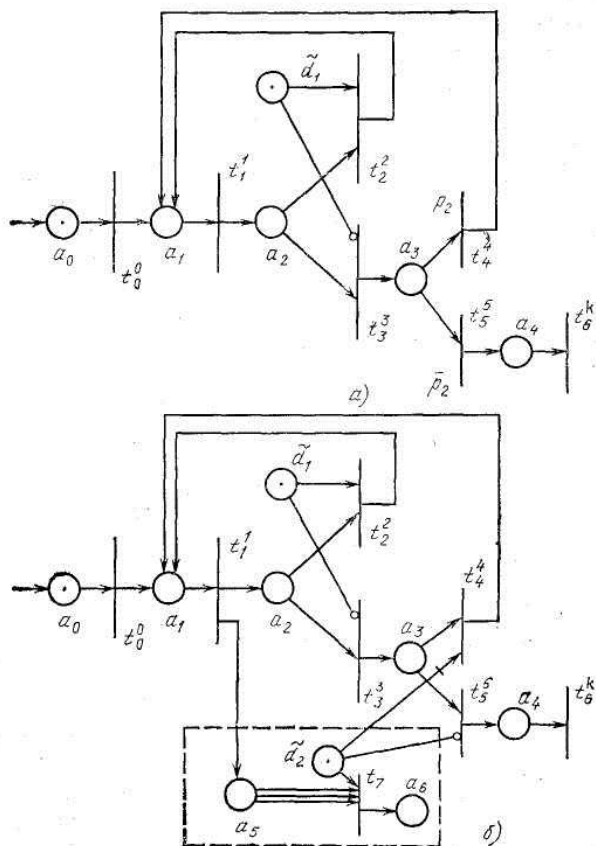


Рис. 5.20.

Состояние ЛР D_2 изменяется процедурой A_1 , что может привести к изменению значения логического условия p_2 , поэтому A_1 ($\{-\}, \{\bar{p}_2\}$).

Кроме того, задано следующее взаимодействие процедур с ЛР:

$A_2(\{p_1\}, \{\bar{p}_1\})$; $A_3(\{\bar{p}_1\} \setminus \{-\})$; $A_4(\{p_1\}, \{-\})$; $A_5(\{\bar{p}_2\}, \{-\})$. Если бы процесс был параллельным, в его описание необходимо было бы ввести позицию состояния ресурса D_2 . Переходы t_4^4 и t_5^5 помечены соответствующими значениями логического условия p_2 , которое приходится считать внешним.

Однако если известен алгоритм функционирования ЛР, то можно представить его в виде отдельной сети Петри и составить ее композицию с сетью Петри управляющего процесса. Пусть, например, ЛР D_2 является счетчиком, состояние которого изменяется каждый раз при выполнении процедуры A_1 . Состояние счетчика сравнивается с константой k , так что при $k=3$ логическое условие p_2 принимает ложное значение (рис. 5.20, б). Теперь позиция d_2 является внутренней, так как ее маркировка полностью определяется состоянием сети Петри, состоящей из двух подсетей. Как видно, здесь потребовалось использовать k -кратную дугу между позицией a_5 и переходом t_7 .

При описании ОУП общего вида необходимо также учитывать то обстоятельство, что одни и те же ресурсы могут запрашиваться разными параллельными подпроцессами. При этом может оказаться, что одни подпроцессы монопольно занимают ЛР D_s , а другие только используют его, т. е. занимают немонопольно. Возникает ситуация, аналогичная задаче о чтении-записи с ограниченным числом процессов чтения, которое в нашем случае не может быть больше числа параллельных подпроцессов. Любой подпроцесс записи должен исключать все другие подпроцессы чтения и записи, тогда как несколько процессов чтения могут выполняться одновременно.

Для моделирования такой ситуации позиция d_s в начальной маркировке должна содержать δ меток, если δ — максимальное число подпроцессов, немонопольно владеющих ресурсом D_s . Позиция d_s является входной и выходной для δ переходов сети Петри этих подпроцессов. Переход сети Петри подпроцесса, который занимает логический ресурс D_s монопольно, соединяется с позицией d_s дугой кратности δ . При этом не нарушается критерий взаимного исключения процессов и допускается одновременная проверка значения логического условия p_s , несколькими параллельными подпроцессами. Ясно, что для этого соответствующие переходы сетей Петри соединяются с позицией $\bar{d}_s$ либо сдерживающими, либо неизменяющими дугами.

На рис. 5.21 показан рассмотренный выше процесс.

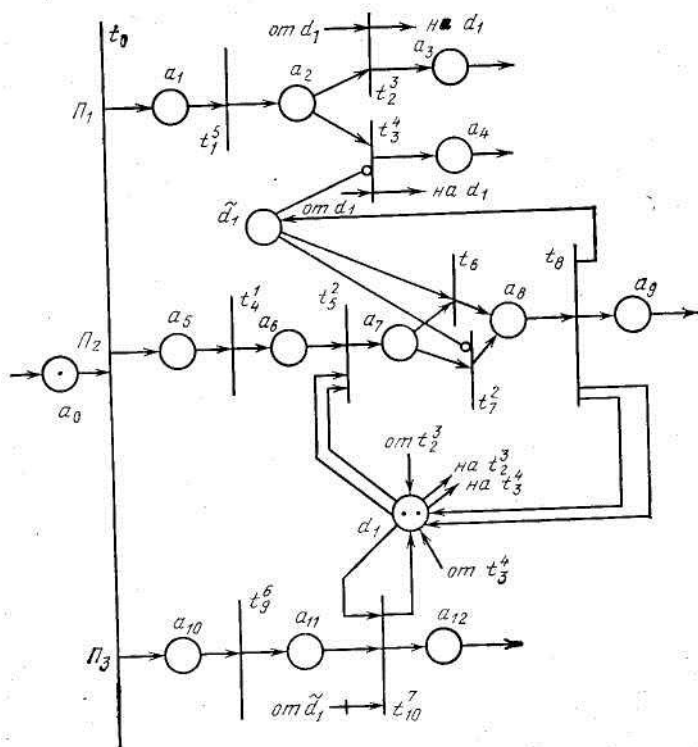


Рис. 5.21.

Здесь подпроцессы P_1 и P_2 немонопольно владеют логическим ресурсом D_1 при выполнении процедур A_3 или A_4 и A_7 , в то же время выполнение процедуры A_2 подпроцесса P_2 не зависит от значения логического условия p_1 , но присваивает ему единичное значение, т. е. $A_3(\{p_1\}, \{-\})$; $A_4(\{\bar{p}_1\}, \{-\})$; $A_7(\{p_1\}, \{-\})$; $A_2(\{-\}, \{p_1\})$.

Наличие двух меток в начальной маркировке позиции d_1 и двухкратных дуг, соединяющих эту позицию с переходом t_7^2 , обеспечивает требуемое взаимодействие параллельных подпроцессов. Действительно, допустима одновременная реализация переходов t_2^3 и t_9^7 , так как каждый из них удаляет по одной метке из позиции d_1 . Реализация перехода t_7^2 удаляет обе метки из этой позиции, запрещая использование логического ресурса D_1 другими подпроцессами. Маркировка позиции $\bar{d}_1$ не изменяется при реализации переходов t_2^3 ,

t_3^4 и t_9^7 , но после реализации t_7^2 эта позиция получает метку — единичное значение логического условия p_1 .

На рисунке показан частный, но довольно распространенный случай разветвленного процесса с состоянием ожидания. При наличии метки в позициях a_{10} и d_1 подпроцесс P_3 не может развиваться (изменяться), если логическое условие p_1 имеет нулевое значение; он будет находиться в этом состоянии до тех пор, пока значение p_1 не изменится. Это частный случай процесса с циклом.

Таким образом, граф обобщенной сети Петри, предназначенной для описания неавтономного управляющего процесса общего вида, содержит длительные и примитивные переходы, основные и ресурсные внутренние позиции, основные, неизменяющие и сдерживающие дуги требуемой кратности. Каждому длительному переходу сопоставлена процедура КП. Если выполнение процедуры A_i зависит от значений внешних ЛУ, то переход t'_d помечается соответствующим предикатом. Реализация длительного перехода занимает некоторое время, в течение которого отсутствуют метки в его входных и выходных позициях. Примитивные переходы, к которым относятся переходы распараллеливания и соединения, реализуются мгновенно, они предназначены для задания структуры процесса. Полное состояние автономного ОУП определяется маркировкой основных (a_i) и внутренних ресурсных позиций сети Петри ($c_j, d_s, \bar{d}_s$). Дуги графа сети Петри определяют последовательность выполнения процедур процесса и их взаимодействие с функциональными и логическими ресурсами.

Такая обобщенная сеть Петри обладает свойствами временных сетей с переходами, помеченными предикатами и операциями, и дугами разных типов. Ее отличительной особенностью является то, что в описание процесса вводятся используемые им ресурсы и учитывается влияние процедур процесса на состояния ресурсов. Это дает возможность осуществить более полный анализ процесса и получить правильный ОУП с учетом ресурсов.

5.2.3.4. Получение правильного облачного управляющего процесса

5.2.3.4.1. Граф достижимых маркировок сети Петри

Формализованное описание ОУП на языке сетей Петри прежде всего используется для *анализа процесса, выявления его свойств, обнаружения и устранения недопустимых состояний*. К числу таких

состояний, наиболее характерных для параллельных процессов с разделяемыми ресурсами, относятся тупиковые состояния, в которых процесс прекращает развитие, не достигнув своего конечного состояния. Причины возникновения тупикового состояния могут быть различны — это и некорректно составленная структура параллельно-последовательного процесса, и неправильное распределение ФР, и недопустимое взаимодействие процедур процесса с ЛР. Поскольку возникновение тупиковых состояний в ОУП приводит к преждевременной остановке технологического процесса, выявление и устранение тупиков является одной из основных задач начального этапа формирования рекомендаций.

Анализ ОУП основан на анализе и выявлении свойств задающей его сети Петри. Для анализа сетей Петри используют различные методы: *построение дерева достижимости, составление матричных уравнений, получение различных инвариантов для позиций и переходов*. Каждый из этих методов имеет свои достоинства и ограничения. Наиболее наглядным и удобным для пояснения является метод, основанный на применении дерева достижимости.

При выполнении сети Петри возникают две взаимно связанные последовательности — маркировок и реализуемых при этом переходов. Дерево достижимости представляет множество достижимости сети Петри, можно называть его также *графом достижимых состояний* сети Петри. Построение такого графа вытекает непосредственно из правил выполнения сети Петри. Начальная вершина графа соответствует начальной маркировке M_0 . Если маркировка M_1 непосредственно достижима из M_0 при реализации перехода t_i , то вершина графа M_0 соединяется с вершиной, соответствующей маркировке M_1 , дугой, помеченной переходом t_i . Затем рассматриваются все маркировки, достижимые из M_1 , и т. д. Поскольку при одной и той же маркировке может быть активизировано несколько альтернативных переходов, то образуется целое дерево достижимости.

В процессе построения такого графа достижимых маркировок может возникнуть маркировка, уже порожденная на предыдущем шаге, тогда в графе образуется цикл. Кроме того, в графе могут возникать маркировки, различающиеся только числом меток в одной и той же позиции. Предлагается возможное бесконечное число меток в маркировках такого типа представить с помощью специального символа ω , описывается алгоритм построения дерева достижимости, доказывается, что этот алгоритм всегда заканчивает свою работу и что дерево достижимости сети Петри конечно.

Очевидно, наличие символа ω в графе достижимых маркировок свидетельствует о том, что число различных маркировок может быть бесконечным, несмотря на конечное число вершин графа. Сеть Петри, для которой получен такой граф, является неограниченной. Если сеть Петри ограничена и символ ω отсутствует в графе ее достижимых состояний, то эта сеть задает систему с конечным числом состояний. Именно к таким системам относится рассматриваемая нами УС, поэтому сеть Петри, описывающая ОУП, реализуемый в УС, должна иметь конечное число состояний.

Граф достижимых маркировок сети Петри N будем обозначать G_N . В вершине графа G_N , соответствующей маркировке $\mathbf{M}_i$, будем выписывать те позиции сети Петри, которые содержат метки при этой маркировке, и те переходы, которые при этом активизированы. Если активизированные переходы входят в параллельные участки процесса, т. е. могут быть реализованы одновременно, в вершине графа G_N они связываются знаком конъюнкции. Активизированные переходы, входящие в альтернативные ветви, соединяются знаком дизъюнкции. Если позиция a_μ в маркировке $\mathbf{M}_i$ имеет более одной метки, число этих меток указывается в виде верхнего индекса этой позиции.

Поскольку каждая маркировка сети Петри определяет состояние рассматриваемой системы, можно считать, что вершина $\mathbf{M}_i$ графа достижимых маркировок сети Петри соответствует состоянию S_i процесса, задаваемого этой сетью. Для каждого процесса задается начальное и конечное состояния; если процесс является циклическим, то выделяется состояние, после которого процесс повторяется. В общем случае может быть несколько начальных и конечных состояний. Начальное состояние соответствует начальной вершине графа G_N , конечное — его терминальной вершине. Состояние процесса, не являющееся конечным, но не допускающее дальнейшего его развития, является *тупиковым*. Вершина графа G_N , соответствующая тупиковому состоянию, характеризуется тем, что при наличии меток в основных внутренних позициях сети Петри нет ни одного активизированного перехода.

5.2.3.4.2. Влияние структуры облачного управляющего процесса на наличие тупиковых состояний

Рассмотрим различные случаи возникновения тупиковых состояний. На рис. 5.22 приведен граф достижимых маркировок простого параллельно-последовательного процесса, заданного сетью Петри рис. 5.19.

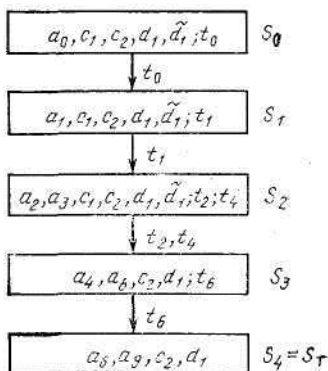


Рис. 5.22.

Для простоты на графе переходы будем выписывать без верхних индексов. Вначале построим граф G_N в предположении, что время реализации всех переходов одинаково, поэтому одновременно активизированные переходы реализуются также одновременно. При этом процесс переходит из состояния S_2 сразу в состояние S_3 . Как видно из рис. 5.22, при заданной начальной маркировке, когда ФР c_1 и c_1 свободны и логическое условие $p_1 = 1$, процесс, не достигнув конечного состояния, соответствующего реализации перехода t_d^k , попадает в состояние S_4 , которое является тупиковым, так как ни один из переходов не является активным и дальнейшее развитие процесса невозможно. Таким образом, при начальном истинном значении логического условия p_1 процесс попадает в тупиковое состояние.

Аналогичный граф достижимости можно построить для начального ложного значения логического условия p_1 когда позиция $\bar{a}_1$ в начальной маркировке не содержит метку. При этом вместо перехода t_2 будет активизирован переход t_3 . Чтобы не составлять отдельные графы для различных допустимых начальных маркировок, можно построить один полный граф, предусматривающий все возможные пути развития процесса. Очевидно, при этом позиции ЛР в вершинах графа не выписываются. Такой граф для сети рис. 5.19 представлен на рис. 5.23; как видно, левая ветвь графа соответствует $p_1=1$ и полностью совпадает с графом рис. 5.22.

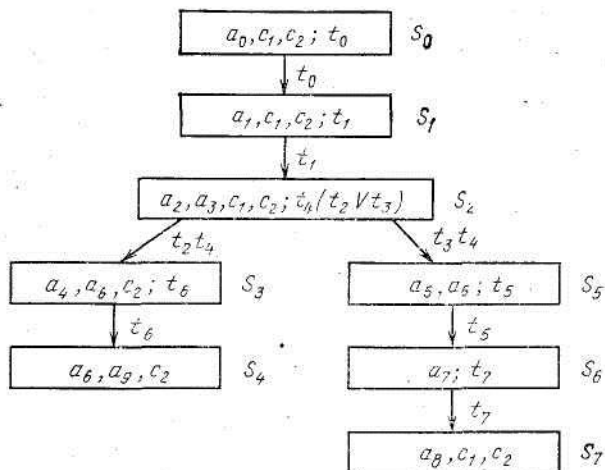


Рис. 5.23.

При $p_1 = 0$ реализуется другая последовательность переходов, но процесс, не достигнув конечного состояния, вновь попадает в тупиковое состояние S_7 . Этот граф также построен в предположении, что реализация активизированных переходов завершается одновременно. Такой граф можно назвать *графом статических состояний* процесса. В статическом состоянии ни один из переходов сети Петри не находится в стадии реализации.

Очевидно, можно построить граф, содержащий не только статические, но и все промежуточные состояния, которые возникают из-за различной скорости реализации переходов. На рис. 5.24 приведен граф, задающий полное множество достижимых состояний рассматриваемого процесса, число которых, как видно, значительно больше числа статических состояний.

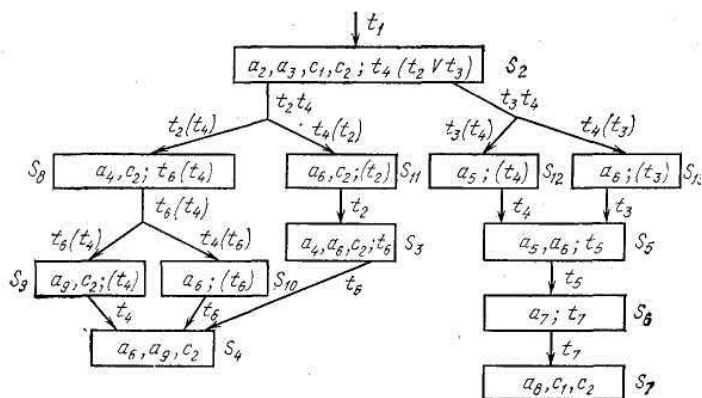


Рис. 5.24.

В таком динамическом графе исходящие из вершин дуги графа помечаются переходами, которые переходят в стадию реализации, входящие дуги помечаются переходом, закончившим реализацию. Переходы, реализация которых продолжается, записываются на этой дуге в скобках. В результате кроме статических состояний в графе появляются неустойчивые состояния; переходы, находящиеся при этом в стадии реализации, также выписываются в скобках в отличие от переходов, которые активизируются в данном неустойчивом состоянии. Следует иметь в виду, что с началом реализации перехода удаляются метки из его входных позиций, поэтому, например, в состоянии S_8 отсутствует метка в позиции a_3 и еще не появилась метка в позиции a_6 , так как реализация перехода t_4 не закончена.

Из рис. 5.24 видно, что независимо от начального значения логического условия p_1 и относительной скорости реализации переходов, т. е. длительности соответствующих процедур, процесс, заданный сетью Петри рис. 5.19, попадает в одно из двух тупиковых состояний. Причиной этого является недопустимая структура процесса. Хотя в общем случае трудно сформулировать требования к правильной структуре процесса, некоторые обязательные условия можно указать. Необходимо, чтобы альтернативные ветви процесса имели альтернативное соединение в пределах того параллельного участка, в котором они начались. Если распараллеливание процесса происходит в одной из альтернативных ветвей, то соединение этих параллельных подпроцессов должно быть также в пределах этой ветви. Нетрудно видеть, что сеть Петри рис. 5.19 не удовлетворяет этим

требованиям. Возникновение таких неприятных явлений, как ловушка, зависание, обычно также связано с нарушением указанных условий. Тупики, вызванные неправильной структурой процесса, могут быть устранены только соответствующим ее изменением.

Другой причиной недостижимости конечного состояния может быть возникновение цикла в графе достижимых маркировок. Такая ситуация свойственна не только параллельному, но и последовательному разветвленному процессу при недопустимом взаимодействии процедур процесса с ЛР.

На рис. 5.25, а приведен граф достижимых маркировок сети Петри рис. 5.18 для фиксированной начальной маркировки позиций $\bar{d}_1$ и $\bar{d}_2$.

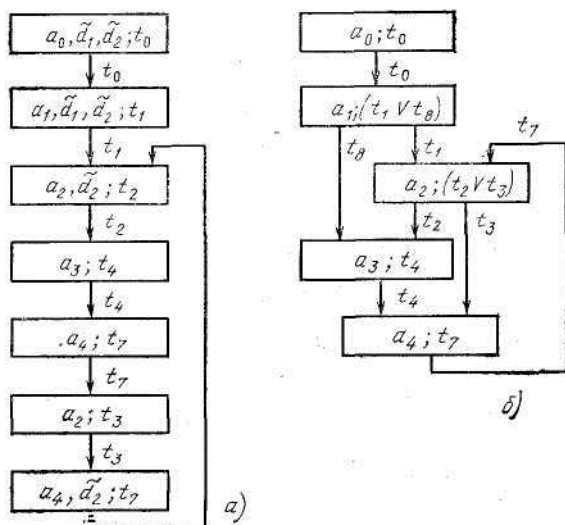


Рис. 5.25.

Как видно, при выполнении процесса возникает бесконечный цикл, так что заключительное состояние, связанное с реализацией перехода t_6^k , оказывается недостижимым. Полный граф достижимости (рис. 5.25, б) показывает, что это состояние не достигается ни при одной из возможных начальных маркировок позиций ЛР. Таким образом, возникновение бесконечного цикла является свойством данного процесса и объясняется недопустимым взаимодействием процедур процесса с логическим ресурсом D_1 . В этом случае также требуется преобразование процесса.

Следует иметь в виду, что для более полного и точного анализа ОУП необходимо использовать всю имеющуюся в распоряжении специалиста информацию, так как в противном случае могут быть сделаны неверные выводы о ходе развития процесса. На рис. 5.26, а приведен граф G_N сети Петри рис. 5.20, а для истинного начального значения внешнего логического условия p_2 . Этот граф содержит цикл, свидетельствующий о том, что если значение p_2 не изменится, процесс не достигнет конечного состояния. Поскольку здесь D_2 является внешним ЛР, неизвестно, произойдет ли такое изменение и будет ли реализован переход t_6^k .

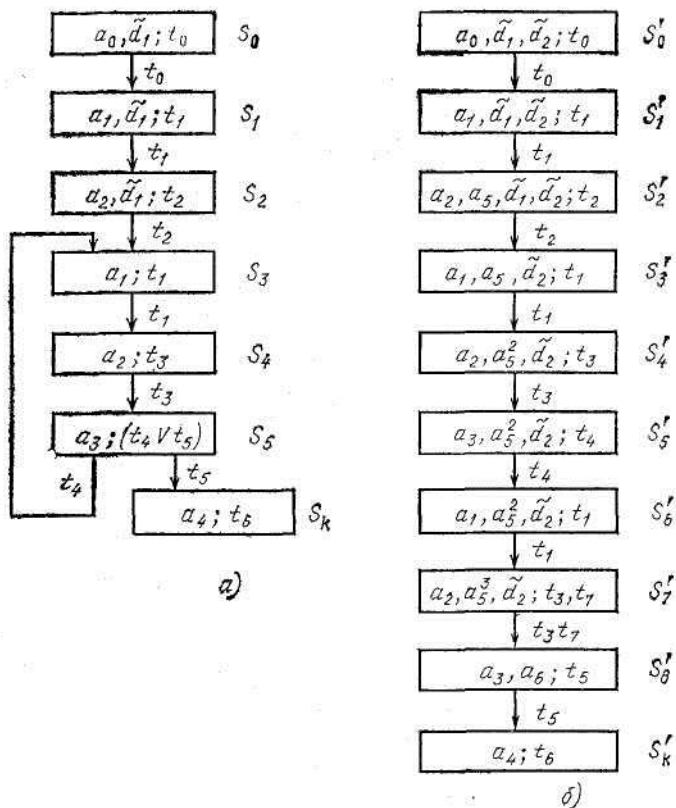


Рис. 5.26.

В сети Петри рис. 5.20, б содержится информация о функционировании логического ресурса D_2 и его взаимодействии с ОУП. Граф достижимости этой сети (рис. 5.26, б) показывает, что

никаких циклов при выполнении процесса не возникает и конечное состояние достижимо.

5.2.3.4.3. Тупиковые состояния, вызываемые разделением функциональных ресурсов

Значительная часть тупиковых состояний возникает из-за использования одних и тех же ФР процедурами параллельных подпроцессов. Рассмотрим более подробно такую ситуацию и способы ее устранения на примере сети Петри рис. 5.16, которая задает два циклических асинхронных процесса P_1 и P_2 , разделяющих ФР C_1 и C_2 . В начальной маркировке сети Петри содержатся метки в позициях c_1 и c_2 , так как оба ресурса свободны, и в позициях a_1 , a_7 , поскольку оба процесса находятся в состоянии готовности. Для указания на асинхронность процессов введены внешние входные позиции b_1 и b_2 для переходов t_1 и t_7 соответственно, так что при появлении метки в позиции b_1 (b_2) начинается выполнение процесса P_1 (P_2). Если метки появляются одновременно в позициях b_1 и b_2 , то оба процесса начинают выполняться параллельно.

Построим граф достижимых маркировок сети Петри и для наглядности изобразим его в виде решетки, горизонтальные пути которой соответствуют функционированию только подсети N_1 , т. е. развитию процесса P_1 , а вертикальные — подсети N_2 (рис. 5.27).

Вершины графа, определяющие состояние сети N , обозначим двойным индексом, первый из которых соответствует состояниям подсети N_1 а второй — подсети N_2 .

Начальное состояние S_{00} является состоянием готовности процессов P_1 и P_2 , каждый из которых начинает выполняться при его активизации, что отмечается появлением метки во внешней входной позиции сети Петри. Следует отметить, что маркировка внешних позиций не зависит от самого процесса. Эти позиции могут заполняться метками и освобождаться от них в любой момент, делая процесс активным либо пассивным. Важно только, чтобы метка во внешней входной позиции b_f перехода t_i сохранялась в течение времени, необходимого для реализации этого перехода. В отличие от внутренних входных позиций перехода t_i , метки из которых удаляются после его реализации, метка из позиции b_f не удаляется.

Активизация и выполнение процесса Π_1 (Π_2) вызывают последовательность состояний $S_{10}, S_{20}, \dots, S_{50}$ ($S_{01}, S_{02}, \dots, S_{05}$). Одновременное выполнение двух процессов соответствует остальным вершинам графа достижимых маркировок. Как видно из рис. 5.27, сеть Петри рис. 5.16 имеет 28 достижимых состояний. Нетрудно заметить, что не все эти состояния являются равноценными. В некоторых состояниях активизировано и может быть реализовано два перехода, что означает параллельное выполнение двух процессов. Другие состояния, например S_{13} , допускают развитие обоих процессов, но одновременно — только одного из них. В других состояниях активизирован только один переход, что говорит о возможности выполнения лишь одного процесса и блокировке другого (S_{41}, S_{14}). Имеется состояние (S_{33}), в котором, несмотря на наличие меток во внутренних позициях сети Петри, ни один из переходов не активизирован, т. е. ни один из процессов не может выполняться. Наличие такого множества различных состояний в графе достижимых маркировок сети Петри приводит к необходимости осуществить их классификацию, введя некоторые определения.

Состоянием блокировки S_6 назовем состояние, в котором при наличии метки во внутренней позиции a_{ij} , являющейся входной для перехода t_i , этот переход не активизирован, т. е. число процессов, которые могут развиваться, меньше числа активных процессов. Такая блокировка процесса происходит из-за занятости необходимых ему ресурсов другими процессами, после выполнения которых ресурсы освобождаются и заблокированный процесс может выполняться.

Состоянием взаимной блокировки $S_{в.6}$ назовем состояние, в котором заблокировано несколько активных процессов и они не могут быть разблокированы, несмотря на развитие других процессов. Таким образом, если в состоянии S_6 развивающиеся процессы освобождают ресурсы, необходимые для выполнения заблокированного процесса, то в состоянии $S_{в.6}$ все ресурсы, запрашиваемые заблокированным процессом, заняты процессами этого же множества, так что развитие других процессов не выводит систему из состояния взаимной блокировки. Если в множество заблокированных процессов входят все рассматриваемые процессы, система попадает в состояние полной взаимной блокировки $S_{п.в.6}$, когда ни один из процессов не может развиваться. Очевидно, состояния $S_{п.в.6}$ и $S_{в.6}$ одинаково недопустимы; будем называть их тупиковыми (S_t).

Состояние сети Петри, из которого все пути графа достижимых маркировок ведут в состояние S_t , назовем предтупиковым $S_{п-т}$. Состояние S_t и все предшествующие ему состояния $S_{п-т}$ объединим в одно множество Q_3 , которое назовем множеством запрещенных

состояний. Если в графе достижимых маркировок сети Петри имеется ребро, соединяющее вершины S_u и S_v такие, что $S_v \in Q_3$, а $S_u \in Q_3$, то состояние S_u назовем опасным ($S_{оп}$). Все опасные состояния образуют множество опасных состояний $Q_{оп}$. Остальные состояния являются безопасными.

Состоянием конфликта ($S_{кн}$) назовем состояние, в котором запрещена одновременная реализация хотя бы одной пары активизированных переходов.

Как видно из рис. 5.27 состояние S_{33} является тупиковым, $Q_3 = \{S_{22}, S_{32}, S_{23}, S_{33}\}$; $Q_{оп} = \{S_{11}, S_{21}, S_{31}, S_{12}, S_{13}\}$.

Состояния S_{13} и S_{31} являются состояниями конфликта, а $S_{41}, S_{53}, S_{14}, S_{35}$ — состояниями блокировки.

Поскольку в графе достижимых маркировок допустимы все пути, то ясно, что при наличии опасных состояний сеть Петри может попасть в тупиковое состояние. Для устранения такой возможности необходимо осуществить преобразование графа, исключив из него все ребра, соединяющие опасные и запрещенные состояния. При этом все опасные состояния станут безопасными и запрещенные состояния будут недостижимы. Такое преобразование графа достижимых маркировок в граф, содержащий только безопасные состояния, должно найти отражение в сети Петри.

Часть пути графа достижимых маркировок, содержащую только опасные состояния, назовем *опасным отрезком пути*. Опасные отрезки, содержащие опасные состояния одного и того же множества $Q_{оп}$, обязательно имеют одну общую величину, которая также соответствует опасному состоянию и может быть названа *корнем опасных отрезков* $S_{к.оп}$ (в крайнем случае это начальная вершина графа). Для того чтобы это состояние стало безопасным, необходимо превратить его в состояние конфликта, запретив одновременную реализацию активизированных переходов, которая переводит сеть в запрещенное состояние. С этой целью введем в сеть Петри дополнительную блокирующую позицию a_6 , являющуюся общей входной позицией этих переходов. Позиция a_6 в начальной маркировке может содержать несколько меток, число которых определяется допустимым числом параллельно выполняемых опасных отрезков. Таким образом, разрешение конфликта в состоянии $S_{к.оп}$ соответствует выбору процессов, выполнение которых будет продолжаться, тогда как остальные процессы блокируются из-за отсутствия меток в позиции a_6 . Эти процессы необходимо разблокировать сразу же, как только устранится опасность возникновения тупика. Для этого позиция a_6 должна быть выходной позицией всех переходов, которыми отмечены ребра графа, соответствующие выходу из опасных отрезков. Тогда

после реализации таких переходов вновь появляются метки в позиции a_6 , что делает возможным выполнение ранее заблокированных процессов.

Таким образом, предлагаемый способ устранения тупиковых состояний состоит во введении своевременной принудительной блокировки процессов. При этом определяется минимальное число процессов, которые необходимо заблокировать, и минимальное число состояний, в которых эта блокировка должна сохраняться.

Для графа достижимых маркировок рассматриваемого примера состояние S_{11} является корнем двух опасных отрезков S_{21} , S_{31} и S_{12} , S_{13} . Это состояние переводится в состояние конфликта введением в сеть Петри рис. 5.16 позиции a_6 , являющейся входной для переходов t_2 и t_8 . Тогда после реализации перехода t_2 (t_8), что соответствует развитию процесса Π_1 (Π_2), удаляется метка из позиции a_6 и процесс Π_2 (Π_1) блокируется. Как видно из графа рис. 5.27 состояние S_{31} (S_{13}) является последним состоянием выбранного опасного отрезка, поэтому при реализации перехода t_4 (t_{10}) блокировка может быть снята. Для этого позиция a_6 должна быть выходной позицией переходов t_4 и t_{10} (рис. 5.28).

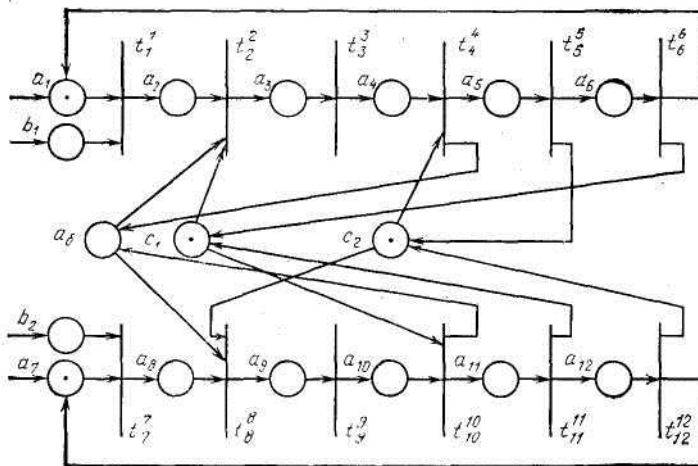


Рис. 5.28.

Теперь состояния S_{12} , S_{13} , S_{21} , S_{31} , которые в исходном графе были опасными, становятся безопасными состояниями блокировки, а

состояние S_{11} —состоянием конфликта. Ясно, что при этом запрещенные состояния S_{22} , S_{32} , S_{23} , S_{33} станут недостижимы.

Для сложной системы взаимодействующих процессов, при которой имеется несколько процессов и они неоднократно обращаются к одним и тем же ресурсам, граф достижимых маркировок может содержать не одно, а несколько множеств запрещенных состояний. Тогда для каждого множества определяются множество опасных состояний, множество опасных отрезков путей графа и корень этих опасных отрезков. Затем в сеть Петри вводится блокирующая позиция так, чтобы опасное состояние, соответствующее корню, перевести в безопасное состояние конфликта. В каждом из конфликтующих опасных отрезков определяется последнее опасное состояние и переходы, которые отмечают ребра, переводящие эти состояния в безопасные, становятся входными переходами позиции a_6 . В сети Петри, которая содержит необходимое число таких блокирующих позиций, не будут возникать тупиковые состояния, связанные с использованием общих ФР процедурами параллельных подпроцессов.

Таким образом, применение графа достижимых маркировок сети Петри позволяет обнаружить тупиковые состояния, выявить причины их возникновения и принять меры к их устранению. Для этого либо изменяется структура процесса и его взаимодействие с ЛР, либо вводится блокировка, запрещающая одновременное развитие некоторых участков параллельных процессов, которые называют иногда критическими. По графу достижимых маркировок можно также обнаружить непроизводительные циклы и недостижимые состояния процесса. Переходы сети Петри, соответствующие таким состояниям, должны быть исключены, если они являются избыточными, либо сделаны достижимыми путем изменения структуры процесса.

Очевидно, все преобразования структуры ОУП могут производиться по согласованию со специалистом по УС, так как иногда тупиковые состояния, например, вводятся специально для сигнализации о неисправностях в системе и т. д. Кроме того, после всех преобразований ОУП должен соответствовать управляемому технологическому процессу.

В результате должен быть получен *правильный ОУП, т. е. процесс, который не содержит непроизводительных бесконечных циклов, тупиковых состояний и все существенные состояния которого достижимы.*

5.3. Облачные управляющие алгоритмы и языки их описания

5.3.1. Последовательный облачный управляющий алгоритм и его свойства

Алгоритм функционирования управляющего средства— это точное предписание о порядке выполнения этим средством процессом операций в ходе функционирования управляющей системы. Хотя понятие алгоритма широко используется в различных областях науки и техники, строгого его определения до сих пор не существует. В ряде работ сформулированы следующие основные свойства, которыми должен обладать любой алгоритм: *детерминированность алгоритма, т. е. наличие точного предписания, не оставляющего места произволу; массовость алгоритма, т. е. приложимость его к изменяющимся в известных пределах исходным данным; результативность алгоритма, т. е. получение результата за конечное число шагов выполнения алгоритма при надлежащих исходных данных.*

Кроме такого содержательного описания понятия алгоритма существуют различные его уточнения, основанные на машине Тьюринга, стекмашине, пушдаун-машине, баллон-машине и др., которые являются абстрактными моделями современных ЭВМ. Мы не будем останавливаться на этих и других уточнениях понятия алгоритма, которые в основном используются в математике для доказательства несуществования алгоритмов, а рассмотрим некоторые формализованные языки их описания. К таким языкам относится уже рассмотренный язык сетей Петри, поскольку он по своим описательным возможностям приближается к машинам Тьюринга и, следовательно, может применяться для представления любых алгоритмов. Однако проверку некоторых свойств управляющих алгоритмов и их преобразование с целью оптимизации структуры УС целесообразно осуществлять с использованием других языков, к числу которых относится язык логических схем алгоритмов и некоторые его модификации.

5.3.1.1. Логические схемы алгоритмов

Логические схемы алгоритмов (ЛСА) были предложены в 1952—1953 гг. А. А. Ляпуновым для целей программирования. Однако они могут использоваться в качестве языка задания облачных алгоритмов функционирования программных управляющих устройств.

Задача (в том числе и задача управления), которую нужно решить, состоит в поиске алгоритма переработки некоторой исходной информации, т. е. в выборе отдельных операций, или *актов* алгоритма, и поиске порядка их выполнения.

С каждым таким актом (операцией) в ЛСА сопоставляется *оператор*, обозначаемый большими латинскими буквами $A, B, C \dots$ Различные операторы могут обозначаться разными буквами или одной и той же буквой, но с различными индексами: $A_1, A_2, \dots, B_1, B_2 \dots$ Если оператор зависит от параметров, то эти параметры могут ставиться в качестве индексов $A_i, A_{ij}, A_{ijk} \dots$ или в скобках: $A(i), A(ij), A(ijk) \dots$ Операторы с различными параметрами выполняют действия над разными частями исходных или промежуточных данных, т. е. над разными частями перерабатываемой информации.

Например, при умножении квадратных матриц необходимо отдельно умножать каждую строку одной матрицы на каждый столбец другой, при этом умножение различных строк и столбцов производится аналогично. Поэтому для умножения различных строк и столбцов достаточно иметь один оператор, параметры которого укажут, какую строку и какой столбец необходимо в данный момент перемножать, т. е. оператор $A_{(ij)}$ обозначает, что необходимо i -ю строку умножить на j -й столбец.

Последовательность выполнения операторов в ЛСА определяется порядком их записи. Например, ABC означает, что вначале выполняется оператор A , затем B , а затем C . Порядок выполнения в ЛСА операторов может быть строго фиксированным — *линейный алгоритм* — или зависящим от некоторых условий — *разветвленный алгоритм*. В последнем случае в ЛСА применяют *логические условия* (ЛУ), обозначаемые малыми латинскими буквами $p, q, r \dots$ Как и операторы, различные ЛУ обозначаются различными буквами или одной и той же буквой, но с разными индексами.

Логические условия могут зависеть от нескольких переменных. Логические условия, зависящие от значений функции n переменных, будем обозначать через

$$p[f(x_1, x_2, \dots, x_n)].$$

Считается, что логические условия могут принимать только два значения: выполняется проверяемое условие ($p_i=1$) или нет ($p_i=0$). В зависимости от значения в данный момент проверяемого ЛУ определяется дальнейший порядок выполнения операторов и ЛУ. Часто среди логических условий целесообразно выделить такие, которые всегда принимают нулевое (ложное) значение, т. е. *тождественно-ложные логические условия*. Тождественно-ложные логические условия не требуют проверки. Будем обозначать их через

ю. Операторы и ЛУ являются основными, а тождественно-ложные логические условия — вспомогательными членами логической схемы алгоритма.

Каждое ЛУ имеет стрелку. Начало i -й стрелки (обозначается через $\uparrow^i$) стоит справа от логического условия, а ее конец (обозначается $\downarrow^i$) — слева от того члена ЛСА, который должен выполняться, если ЛУ принимает нулевое значение.

Логическими схемами алгоритма называют выражения, составленные из следующих друг за другом операторов и ЛУ, а также расставленных определенным образом нумерованных стрелок. Логическая схема алгоритма есть некоторый способ описания алгоритма решения поставленной задачи. Такое описание не зависит от того, какими средствами задача решается.

Описание алгоритма при помощи логических схем есть первый этап формализации алгоритма. Этому этапу предшествует содержательное описание алгоритма.

Логическая схема алгоритма допускает как формальные, так и содержательные равносильные преобразования.

Рассмотрим порядок выполнения алгоритма по логической схеме этого алгоритма. Пусть, например, из операторов A , B , C и логических условий p_1 и p_2 составлена следующая логическая схема алгоритма:

$$\downarrow^2 A p_1 \uparrow^1 B \downarrow^1 p_2 \uparrow^2 C. \quad (5.5)$$

Логическая схема определяет порядок выполнения операторов в зависимости от значения входящих в нее ЛУ.

Работа алгоритма начинается с того, что выполняется самый левый член схемы. После того как некоторый член схемы выполнится, определяется, какой член схемы должен выполняться следом за ним. Если это был оператор, то следом за ним должен выполняться тот член схемы, который стоит непосредственно справа от него. Если последний выполнившийся член схемы был логическим условием, то возможны два случая: если проверявшееся условие выполнено, то должен выполняться член, находившийся справа; если оно нарушено, должен выполняться тот член, к которому ведет стрелка, начинающаяся после данного условия.

Работа алгоритма оканчивается либо тогда, когда последний из выполняющихся операторов содержит указание о прекращении работы алгоритма, либо тогда, когда на некотором этапе не оказывается такого члена схемы, который должен был бы выполняться.

Таким образом, распределение значений ЛУ в логической схеме определяет порядок выполнения операторов, входящих в эту схему.

Например, в логической схеме (5.5) порядок выполнения операторов в зависимости от значений ЛУ будет следующим:

- 1) если $p_1=0$ и $p_2=0$, то $AA \dots A \dots$, т. е. бесконечно будет выполняться оператор A ;
- 2) если $p_1=0$, а $p_2=1$, то AC , т. е. вначале выполняется оператор A , а затем C и алгоритм заканчивает свою работу;
- 3) если $p_1=1$, а $p_2=0$, то будет бесконечная последовательность чередования операторов A и B : $ABAB \dots AB \dots$;
- 4) если $p_1=1$ и $p_2=1$, то последовательно выполняются все три оператора и алгоритм заканчивает свою работу, т. е. ABC .

Следует заметить, что в процессе работы алгоритма логические условия могут изменяться или из-за того, что изменяются внешние к данной управляющей системе условия или значения этих логических условий изменяются после выполнения того или иного оператора.

Например, в логической схеме алгоритма

$$A \downarrow^1 B p \uparrow^2 C \omega \uparrow^1 \downarrow^2 D \quad (5.6)$$

будут поочередно выполняться операторы B и C до тех пор, пока не изменится значение логического условия p с 1 на 0:

$$\underbrace{ABCBC \dots BC \dots BC}_{p=1} \underbrace{BD}_{p=0}.$$

В качестве примера составления ЛСА рассмотрим алгоритм Евклида, решающий задачу нахождения наибольшего общего делителя для двух заданных натуральных чисел a и b . Решение для любых a и b можно получить путем построения убывающей последовательности чисел, из которых первое является большим из двух данных, второе — меньшим, третье получается как остаток от деления первого на второе, четвертое — как остаток от деления второго на третье и т. д., пока не будет деления без остатка. Делитель в последнем делении и будет искомым результатом.

Поскольку деление может быть сведено к повторному вычитанию, можно предложить следующий список предписаний:

- 1) сравнить числа a и b ($a=b$, $a<b$, $a>b$) и перейти к следующему предписанию;
- 2) если числа равны, то каждое из них дает искомый результат. Процесс вычисления остановить. Если нет, то перейти к следующему предписанию;
- 3) если $a<b$, то поменять числа местами и перейти к следующему предписанию;
- 4) вычесть второе из первого и в качестве исходных взять два числа: вычитаемое и остаток. Перейти к предписанию 1.

Этот перечень предписаний есть словесная формулировка алгоритма, по которой можно составить логическую схему алгоритма, введя следующие операторы и ЛУ:

A — оператор, указывающий, какую пару чисел выбираем;

B — оператор вычитания второго числа y из первого x ;

C — оператор перестановки чисел x и y ;

D — оператор выбора вычитаемого и остатка в качестве первого и второго чисел;

O — оператор остановки и выдачи результата;

p — логическое условие, определяющее равенство двух чисел x и y ; $p=1$, если $x=y$; $p=0$, если $x \neq y$;

q — логическое условие, проверяющее равенство $x > y$;

$q=1$, если $x > y$; $q=0$, если $x \not> y$.

Для введенного таким образом множества операторов и ЛУ логическая схема алгоритма будет иметь вид

$$A\bar{p}\uparrow^1\bar{q}\uparrow^2\uparrow^3C\downarrow^2BD\bar{p}\uparrow^1\bar{q}\uparrow^2\omega\uparrow^3\downarrow^1O. \quad (5.7)$$

Здесь $\bar{p}$ — отрицание p , т. е. если $p=1$, то $\bar{p}=0$ и если $p=0$, то $\bar{p}=1$, аналогично и для q .

В ЛСА часто выделяют оператор A_0 , символизирующий начало выполнения алгоритма, — *оператор начала* и оператор A_k , символизирующий окончание работы алгоритма, — *оператор конца*. Последний оператор часто заменяют точкой.

Кроме ЛСА на практике используются и другие способы описания структуры алгоритма. Наиболее распространенными из них являются *матричные схемы алгоритма* (МСА) и *формулы перехода*, обычно употребляемые в качестве промежуточных языков при преобразовании ЛСА.

5.3.1.2. Матричные схемы алгоритмов и их связь с логическими схемами.

Матричная схема алгоритма — это квадратная матрица (табл. 5.6), каждая строка и каждый столбец которой сопоставлены с оператором.

Таблица 5.6

$$\mathfrak{A} = \begin{matrix} & A_1 & . & . & . & A_k \\ \begin{matrix} A_0 \\ A_1 \\ \vdots \\ A_{k-1} \end{matrix} & \begin{bmatrix} \alpha_{0,1} & . & . & . & \alpha_{0k} \\ \alpha_{1,1} & . & . & . & \alpha_{1k} \\ \vdots & & & & \vdots \\ \alpha_{k-1,1} & . & . & . & \alpha_{k-1k} \end{bmatrix} \end{matrix}$$

Элементом a_{ij} матрицы является логическая функция логических условий. При этом оператор A_j , сопоставленный с j -м столбцом матрицы, выполняется после оператора A_i , сопоставленного с i -й строкой, если логическая функция

$$a_{ij} = a_{ij}(p_1 \dots, p_m) = 1.$$

Логические функции МСА обладают следующими двумя свойствами, определяющими условия *непротиворечивости* и *полноты* алгоритма соответственно:

$$1) a_{ij}a_{il} = 0, i \neq l;$$

$$2) \bigvee_{j=1}^k a_{ij} = 1,$$

иными словами, произведение двух различных функций одной и той же строки МСА всегда равно 0. Это первое условие нужно для того, чтобы после оператора A_i могло выполняться не более одного оператора. Из второго условия следует, что после оператора A_i всегда должен выполняться хотя бы один оператор. Таким образом, после оператора A_i всегда выполняется только один оператор.

От ЛСА легко перейти к МСА. Рассмотрим такой переход на примере ЛСА вида (5.7). Вначале заполним строку МСА, сопоставленную с оператором A . Для этого найдем все функции α_{A_i} . Если $\bar{p}=1$ и $\bar{q}=1$, то после A необходимо выполнить оператор C . Следовательно, $\alpha_{AC} = \bar{p}\bar{q}$. Если $\bar{p}=1$, а $\bar{q}=0$, то после A необходимо выполнить оператор B . Таким образом, $\alpha_{AB} = \bar{p}q$. Аналогично при $\bar{p}=0$ получим $\alpha_{AO} = p$. Элемент α_{AD} равен нулю, так как ни при каких наборах значений ЛУ после оператора A не может непосредственно выполняться оператор D .

После оператора C всегда выполняется оператор B , т. е. $\alpha_{CB} = 1$. Аналогично, образуя и другие логические функции, получим МСА, соответствующую ЛСА (5.7) (табл. 5.7).

Таблица 5.7

	C	B	D	O
A	$\bar{p}\bar{q}$	$\bar{p}q$		p
C		1		
B			1	
D	$\bar{p}\bar{q}$	$\bar{p}q$		p

Существует ряд способов перехода и от МСА к ЛСА. Рассмотрим наиболее простой, основанный на использовании формул перехода. Формула перехода записывается следующим образом.

Выбирается оператор (например, C) в МСА (табл. 5.7). Если после оператора C непосредственно следует только один оператор (в нашем случае оператор B), то этот факт записывается в виде $C \cdot B$. Если после оператора (например, A в МСА табл. 5.7) может выполняться один из нескольких операторов в зависимости от значений логических условий, то в формуле перехода после полустрелки записывается первый в строке ненулевой элемент α_{ij} , который отмечается оператором A_j . Затем к такому члену с помощью знака дизъюнкции присоединяется следующий в этой строке ненулевой элемент α_{il} , отмеченный оператором A_l и т. д.

Так, в нашем случае для оператора A получим следующую формулу перехода:

$$A \rightarrow \bar{p} \bar{q} C \vee \bar{p} q B \vee p O.$$

Образовав такие формулы перехода для всех операторов, получим систему формул перехода. Например, по МСА (табл. 5.7) получим следующую систему формул перехода:

$$\begin{cases} A \rightarrow \bar{p} \bar{q} C \vee \bar{p} q B \vee p O. \\ C \rightarrow B; \\ B \rightarrow D; \\ D \rightarrow \bar{p} \bar{q} C \vee \bar{p} q B \vee p O. \end{cases}$$

От системы формул перехода можно перейти к ЛСА. Для этого необходимо прежде выписать оператор начала A_0 . В нашем случае это оператор A . После него выписываются ЛУ (снабженные стрелками), соответствующие первому слева члену формулы перехода A , т. е. получим

$$A \bar{p} \uparrow \bar{q} \uparrow .$$

Затем выписывается оператор, отмечающий заданную функцию, т. е.

$$A \bar{p} \uparrow \bar{q} \uparrow C.$$

Когда выписан оператор, необходимо обратиться к формуле этого оператора и повторить эту процедуру:

$$A \bar{p} \uparrow \bar{q} \uparrow C B.$$

Затем получим последовательно:

$$A \bar{p} \uparrow \bar{q} \uparrow C B D; \quad A \bar{p} \uparrow \bar{q} \uparrow \downarrow ' C B D \bar{p} \uparrow \bar{q} \uparrow \omega \uparrow ' .$$

В последнем выражении введено тождественно-ложное ЛУ ω со стрелкой. Это сделано для того, чтобы не повторять в выражении дважды оператор C .

Когда последовательно уже нельзя выписать ни одного члена ЛСА (как это имеет место в нашем случае), необходимо обратиться к первому слева в последнем выражении ЛУ; пронумеровать стрелку и выписать члены, следующие за оператором A при $\bar{p}=0$, и т. д.

В результате находим

$$\begin{aligned} & A\bar{p} \uparrow^2 q \uparrow \downarrow^1 CBD\bar{p} \uparrow \bar{q} \uparrow \omega \uparrow^1 \downarrow^2 O; \\ & A\bar{p} \uparrow^2 \bar{q} \uparrow^3 \downarrow^1 C \downarrow^3 BD\bar{p} \uparrow \bar{q} \uparrow \omega \uparrow^1 \downarrow^2 O; \\ & A\bar{p} \uparrow^2 \bar{q} \uparrow^3 \downarrow^1 C \downarrow^3 BD\bar{p} \uparrow^2 \bar{q} \uparrow \omega \uparrow^1 \downarrow^2 O; \\ & A\bar{p} \uparrow^2 \bar{q} \uparrow^3 \downarrow^1 C \downarrow^3 BD\bar{p} \uparrow^2 \bar{q} \uparrow^3 \omega \uparrow^1 \downarrow^2 O. \end{aligned} \quad (5.8)$$

В последнем выражении (5.8) выписаны все операторы МСА (табл. 5.7) и все стрелки пронумерованы. Это выражение и является ЛСА. При этом легко заметить, что ЛСА (5.8) с точностью до обозначения стрелок повторяет ЛСА (5.7).

Таким образом, формулы переходов использованы в качестве промежуточного языка при переходе от МСА к ЛСА. Правда, легко заметить, что переход от МСА к ЛСА можно осуществить и без формул перехода.

Однако, как далее будет видно, формулы перехода эффективно применяются при переходе от МСА к ЛСА, если при этом ставится задача получения минимальной ЛСА, т. е. ЛСА с минимальным числом членов.

В ряде случаев для задания алгоритма функционирования могут быть использованы *граф-схемы алгоритмов*, которые дают более наглядное представление об алгоритме.

Граф-схема, соответствующая ЛСА (5.4), имеет вид схемы, изображенной на рис. 5.29.

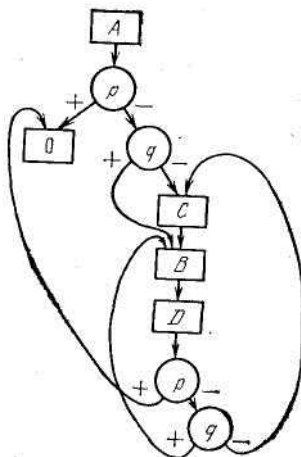


Рис. 5.29.

Из рисунка и ЛСА (5.8) легко видеть их взаимную связь, причем на граф-схеме операторы указаны прямоугольниками, а логические условия — кружочками. Начальный оператор (оператор A) не имеет входных стрелок, конечный (оператор O) — выходных. Единичное значение ЛУ на граф-схеме соответствует стрелке, отмеченной знаком $+$, а нулевое — стрелке со знаком $-$.

5.3.2. Преобразование логических схем алгоритмов

Процесс составления управляющего алгоритма является в настоящее время одним из самых неформализованных этапов построения УС. Несмотря на попытки автоматизировать этот процесс на основе диалога ЭВМ со специалистом, получаемая при этом ЛСА, задающая алгоритм функционирования УС, может содержать избыточное число членов. В то же время минимальность ЛСА является одним из критериев оптимальности структуры УС. В связи с этим возникает задача преобразования ЛСА с целью сокращения числа ее членов.

5.3.2.1. Минимизация числа логических условий

Рассмотрим метод, основанный на применении формул перехода и предполагающий отсутствие повторяющихся операторов в ЛСА. По

имеющейся ЛСА или МСА составляется система формул перехода, затем в каждой формуле перехода находятся переменные, которые входят во все члены формулы. Заметим, что всегда можно найти хотя бы одну такую переменную. Если это переменная p_i , то формула перехода приведена по переменной p_i .

Например, возьмем следующую систему формул перехода из п. 5.3.1:

$$\begin{cases} A \rightarrow \bar{p} \bar{q} C \vee \bar{p} q B \vee p O; \\ C \rightarrow B; \\ B \rightarrow D; \\ D \rightarrow \bar{p} \bar{q} C \vee \bar{p} q B \vee p O, \end{cases}$$

полученную по МСА (табл. 5.7), равносильной ЛСА (5.7). Легко видеть, что в этой системе формула перехода для оператора A приведена по переменной p .

Среди переменных, по которым приведены формулы перехода, выбирают общие для всех или большинства формул. В оставшихся формулах выбирают другие общие переменные и т. д.

В нашем случае выберем переменную p , так как по ней приведены формулы переходов для операторов A и D ; в остальных формулах переменных нет. Одну из выбранных переменных выносят за скобки в тех формулах перехода, которые приведены по этой переменной. Для рассматриваемого примера получим

$$\begin{cases} A \rightarrow \bar{p} \bar{q} C \vee \bar{p} q B \vee p O; \\ C \rightarrow B; \\ B \rightarrow D; \\ D \rightarrow \bar{p} \bar{q} C \vee \bar{p} q B \vee p O, \end{cases}$$

Если в скобках получены выражения типа $(\bar{q} C \vee q B)$, т. е. такие, в которых уже нельзя вынести за скобки ни одной переменной, то такие скобки называют *элементарными*. Все формулы перехода преобразовывают так, чтобы процесс вынесения за скобки привел к элементарным скобкам.

В результате этого получим систему скобочных формул перехода. В ней выделяют общие выражения. Например, в полученной выше системе скобочных формул имеется общее выражение

$\bar{p} (\bar{q} C \vee q B) \vee p O$ в формулах перехода для операторов A и D .

Из всех вхождений в систему одного и того же выражения оставляют только одно (пусть в первой из таких формул перехода и первой слева в одной формуле перехода). Остальные заменяют тождественно-ложными логическими условиями $\omega \uparrow^i$, конец стрелки от которых помещают перед оставшимся выражением.

Таким образом, в рассматриваемом случае получим следующую преобразованную систему скобочных формул перехода:

$$\begin{cases} A \rightarrow \downarrow^1 \bar{p} (\bar{q}C \vee qB) \vee pO; \\ C \rightarrow B; \\ B \rightarrow D; \\ D \rightarrow \omega \uparrow^1. \end{cases}$$

После того как из системы скобочных формул перехода исключены все повторяющиеся выражения, можно переходить к ЛСА, для чего необходимо выполнить следующие действия. Выписывается оператор начала, после которого—логическое условие, первое слева в формуле перехода этого оператора, затем второе условие и т. д. до тех пор, пока не будет выписан следующий оператор.

Например, из формулы перехода для оператора A выпишем следующее выражение:

$$A \downarrow^1 \bar{p} \uparrow \bar{q} \uparrow C.$$

При этом, если перед выписываемым оператором или ЛУ стоит стрелка, она также выписывается с сохранением номера. Затем, как и ранее, обращаемся к формуле перехода оператора C , т. е.

$$A \downarrow^1 \bar{p} \uparrow \bar{q} \uparrow \dot{C} B.$$

Далее последовательно получим:

$$A \downarrow^1 \bar{p} \uparrow \bar{q} \uparrow C B D; \quad A \downarrow^1 \bar{p} \uparrow \bar{q} \uparrow C B D \omega \uparrow^1.$$

Если этот процесс обрывается, необходимо пронумеровать первую слева безындексную стрелку в последнем выражении, обратиться к соответствующей формуле перехода и аналогично рассмотренному выше продолжить построение ЛСА. Таким образом, получим следующую последовательность выражений, иллюстрирующую процесс построения ЛСА:

$$\begin{aligned} & A \downarrow^1 \bar{p} \uparrow^2 \bar{q} \uparrow C B D \omega \uparrow^1 \downarrow^2 O; \\ & A \downarrow^1 \bar{p} \uparrow^2 \bar{q} \uparrow^3 C \downarrow^3 B D \omega \uparrow^1 \downarrow^2 O. \end{aligned} \quad (5.9)$$

Выражение (5.9) является ЛСА. При этом полученная таким образом ЛСА равносильна исходной ЛСА, по которой была построена МСА, т.

е. при подстановке вместо ЛУ какого-либо набора их значений как в исходную, так и в получаемую этим способом ЛСА получаем по ним одну и ту же последовательность выполнения операторов. Следовательно, ЛСА (5.9) равносильна ЛСА (5.7), хотя в них содержится различное число членов, причем ЛСА (5.9) содержит меньшее число членов, чем ЛСА (5.7).

Таким образом можно осуществлять минимизацию ЛСА. Для этого от исходной ЛСА необходимо перейти к МСА, затем к формулам перехода, преобразовать их в скобочные, от которых перейти к минимальной ЛСА. Правда, необходимо отметить, что для построения минимальной ЛСА в ряде случаев необходимо осуществлять перебор вариантов вынесения переменных за скобки, объединение одинаковых выражений и т. п.

Заметим, что при минимизации ЛСА можно и не переходить к МСА, а систему формул перехода получать непосредственно по исходной ЛСА.

Рассмотрим пример упрощения ЛСА (5.10):

$$\mathfrak{A} = p_1 \uparrow^1 p_2 \uparrow^2 \downarrow^5 A_1 p_3 \uparrow^3 p_2 \uparrow^4 p_1 \uparrow^3 \omega \uparrow^5 \downarrow^1 p_2 \uparrow^5 \downarrow^6 \omega \uparrow^6 \downarrow^2 A_2 \downarrow^6 A_3 p_2 \uparrow^6 \omega \uparrow^3 \downarrow^4 p_1 \uparrow^6 \omega \uparrow^2 \downarrow^3 A_5, \quad (5.10)$$

По ЛСА (5.10) составляем МСА (табл. 5.8).

Таблица 5.8

$$\mathfrak{A} = \begin{matrix} & A_1 & A_2 & A_3 & A_K \\ \begin{matrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{matrix} & \left[\begin{array}{cccc} p_1 p_2 \vee \bar{p}_1 \bar{p}_2 & p_1 \bar{p}_2 & \bar{p}_1 p_2 & \\ p_1 p_2 p_3 & p_1 \bar{p}_2 p_3 & \bar{p}_1 \bar{p}_2 p_3 & \bar{p}_1 p_2 p_3 \vee \bar{p}_3 \\ & & 1 & \\ & & \bar{p}_2 & p_2 \end{array} \right] \end{matrix}$$

По МСА (табл. 5.8) получим систему уравнений перехода и преобразуем ее в систему скобочных формул перехода:

$$\begin{aligned} A_0 &\rightarrow p_1 p_2 A_1 \vee \bar{p}_1 \bar{p}_2 A_1 \vee p_1 \bar{p}_2 A_2 \vee \bar{p}_1 p_2 A_3 = p_1 (p_2 A_1 \vee \bar{p}_2 A_2) \vee \bar{p}_1 (\bar{p}_2 A_1 \vee p_2 A_3); \\ A_1 &\rightarrow p_1 p_2 p_3 A_1 \vee p_1 \bar{p}_2 p_3 A_2 \vee \bar{p}_1 \bar{p}_2 p_3 A_3 \vee \bar{p}_1 p_2 p_3 A_K \vee \bar{p}_3 A_K = \\ &= p_3 (p_1 p_2 A_1 \vee p_1 \bar{p}_2 A_2 \vee \bar{p}_1 \bar{p}_2 A_3 \vee \bar{p}_1 p_2 A_K) \vee \bar{p}_3 A_K = \\ &= p_3 (p_1 (p_2 A_1 \vee \bar{p}_2 A_2) \vee \bar{p}_1 (\bar{p}_2 A_3 \vee p_2 A_K) \vee \bar{p}_3 A_K); \\ A_2 &\rightarrow A_3; \\ A_3 &\rightarrow \bar{p}_2 A_3 \vee p_2 A_K. \end{aligned}$$

выявив в формулах перехода этой системы общие выражения, преобразуем ее в следующую систему:

$$\begin{aligned} A_0 &\rightarrow p_1 \downarrow^\alpha (p_2 A_1 \vee \bar{p}_2 A_2) \vee \bar{p}_1 (\bar{p}_2 A_1 \vee p_2 A_3); \\ A_1 &\rightarrow p_3 (p_1 \omega \uparrow^\alpha \vee \bar{p}_1 \downarrow^\beta (\bar{p}_2 A_3 \vee p_2 A_4)) \vee \bar{p}_3 A_5; \\ A_2 &\rightarrow A_3; \\ A_3 &\rightarrow \omega \uparrow^\beta. \end{aligned}$$

По последней системе составим следующую ЛСА:

$$\mathfrak{A} = p_1 \uparrow^3 \downarrow^5 p_2 \uparrow^1 \downarrow^4 A_1 p_3 \uparrow^7 p_1 \uparrow^6 \omega \uparrow^5 \downarrow^1 A_2 \downarrow^2 A_3 \downarrow^6 \bar{p}_2 \uparrow^7 \omega \uparrow^2 \downarrow^3 \bar{p}_2 \uparrow^2 \omega \uparrow^4 \downarrow^7 A_5, \quad (5.11)$$

Легко видеть, что ЛСА (5.11) содержит десять основных членов вместо двенадцати в равносильной ей ЛСА (5.10).

5.3.2.2. Учет распределения сдвигов при минимизации ЛСА

В процессе выполнения алгоритма функционирования отдельные операторы могут изменять значения ЛУ. Например, имеется *ОФБ*, (сопоставленный с оператором A_i ЛСА), осуществляющий прибавление единицы к числу l . Логический *ФБ_j* проверяет значение числа l и при $l \geq n$ вырабатывает значение логического условия $p_j = 1$; при $l < n$ логическое условие $p_j = 0$, т. е. *ЛФБ_j* реализует логическое условие p_j ($l \geq n$) ЛСА. Следовательно, оператор A_i может изменить значение логического условия p_j ($l \geq n$).

Известно, что логическое условие p_j входит в *распределение сдвигов оператора A_i* т. е. $A_i \rightarrow \{p_j\}$, если оператор A_i может изменить значение p_j .

Таким образом, с каждым оператором A_i ЛСА может быть сопоставлено некоторое множество логических условий $\{p_{i1}, \dots, p_{i\zeta}\}$ из множества $\{p_1, \dots, p_n\}$, т. е. $A_i \rightarrow p_{i1}, \dots, p_{i\zeta}$. Такое соответствие и называется *распределением сдвигов*.

Если с каждым оператором ЛСА сопоставлено полное множество ЛУ, т. е. $\zeta = n$, то *распределение сдвигов универсальное*; если с каждым оператором ЛСА сопоставлено пустое подмножество ЛУ, то *распределение сдвигов пустое*.

Заметим, что, если из условий работы УС нельзя или трудно выявить для оператора A_i логические условия, значения которых он может изменить, необходимо принимать универсальное распределение сдвигов.

Выше изложен способ минимизации ЛСА без учета влияния изменения значений ЛУ операторами ЛСА, т. е. этот способ применим в том случае, когда имеется универсальное распределение сдвигов. Учет конкретного распределения сдвигов позволяет в ряде случаев произвести дальнейшее упрощение ЛСА за счет сокращения числа

вхождений ЛУ или числа проверок их значений в процессе выполнения ЛСА.

Рассмотрим несколько примеров, иллюстрирующих данный способ упрощения.

Пусть для ЛСА (5.12) задано следующее распределение сдвигов:

$$\begin{aligned} A_0 &= \{p_1, p_2, p_3\}; A_1 = \{p_2, p_3\}; A_3 = \{\text{---}\}; \\ A_k &= \{p_1, p_2, p_3\}; \\ \mathfrak{A} &= p_1 \uparrow A_1 p_1 \uparrow p_2 \uparrow A_2 p_1 \uparrow p_2 \uparrow p_3 \uparrow A_3 \downarrow A_k. \end{aligned} \quad (5.12)$$

Заметим, что для оператора A_3 задано пустое распределение сдвигов (черточка в скобках), а для операторов A_0 и A_k — универсальное.

Для упрощения ЛСА с учетом распределения сдвигов удобно использовать МСА, поэтому от ЛСА (5.12) перейдем к МСА (табл. 5.9).

Таблица 5.9

$$\mathfrak{A} = \begin{matrix} & A_1 & A_2 & A_3 & A_k \\ \begin{matrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{matrix} & \left[\begin{array}{cccc} p_1 & & & \bar{p}_1 \\ & p_1 \bar{p}_2 & & p_1 \bar{p}_2 \vee \bar{p}_1 \\ & & p_1 p_2 p_3 & p_1 p_2 \bar{p}_3 \vee p_1 \bar{p}_2 \vee \bar{p}_1 \\ & & & 1 \end{array} \right] \end{matrix}$$

Из МСА (табл. 5.9) видно, что оператор A_1 может выполняться лишь после оператора A_0 при $p_1=1$. Вместе с тем логическое условие p_1 не входит в распределение сдвигов оператора A_1 . Значит, оператор A_1 не может изменить значение логического условия p_1 поэтому после выполнения A_1 логическое условие по-прежнему будет иметь единичное значение. В связи с этим после выполнения оператора A_1 не нужно проверять p_1 и можно в строке Л[МСА (табл. 5.9) переменную p_1 заменить ее значением, т. е. получить следующую МСА (табл. 5.10).

Таблица 5.10

$$\mathfrak{A} = \begin{matrix} & A_1 & A_2 & A_3 & A_k \\ \begin{matrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{matrix} & \left[\begin{array}{cccc} p_1 & & & \bar{p}_1 \\ & p_2 & & \bar{p}_2 \\ & & p_1 p_2 p_3 & p_1 p_2 \bar{p}_3 \vee p_1 \bar{p}_2 \vee \bar{p}_1 \\ & & & 1 \end{array} \right] \end{matrix}$$

Из этой МСА видно, что оператор A_3 может быть выполнен только после A_2 , если $p_1 = 1$, $p_2=1$ и $p_3=1$. Однако ни p_1 , ни p_2 в распределение сдвигов оператора A_2 не входят. Следовательно, значения логических условий p_1 и p_2 будут такими, какими они были до выполнения оператора A_2 .

Из табл. 5.10 видно, что для выполнения оператора A_2 необходимо иметь значение $p_2= 1$. Поэтому, как и ранее, в строке A_2 МСА (табл. 5.10) переменную p_2 можно заменить единицей, а ее отрицание — нулем. Вместе с тем из табл. 5.10 не следует, что оператор A_2 , выполнится лишь при каком-то одном значении p_1 . В связи с этим необходимо рассмотреть более подробно условия выполнения оператора A_2 и его предшественников. Оператор A_2 выполняется только после оператора A_1 , который, как и A_2 , не изменяет значения p_1 , поэтому рассмотрим условия выполнения оператора A_1 . Из МСА видно, что оператор A_1 может выполняться при $p_1=1$. Так как ни A_1 , ни A_2 не изменяют значение p_1 , то после выполнения оператора A_2 логическое условие p_1 по-прежнему будет иметь единичное значение. Следовательно, в строке A_2 МСА переменную p_1 можно заменить единицей, а ее отрицание — нулем. В результате получим МСА (табл. 5.11).

Таблица 5.11

$$\mathfrak{A} = \begin{matrix} & A_1 & A_2 & A_3 & A_K \\ \begin{matrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{matrix} & \left[\begin{matrix} p_1 & & & \bar{p}_1 \\ & p_2 & & \bar{p}_2 \\ & & p_3 & \bar{p}_3 \\ & & & 1 \end{matrix} \right] \end{matrix}$$

Заметим, что условие выполнения A_2 при $p_1=1$ следует и из МСА (табл. 5.9), в которой еще не произведено упрощение с учетом распределения сдвигов, тогда как из табл. 5.10 этого не видно. Поэтому желательно не заменять переменные их значениями до окончания упрощения ЛСА.

По полученной МСА (табл. 5.11) легко получить систему формул перехода, а затем и упрощенную, ЛСА (5.13), имеющую только семь членов вместо десяти в ЛСА (5.12), составленной без учета распределения сдвигов:

$$\begin{aligned} A_0 &\rightarrow p_1 A_1 \vee \bar{p}_1 A_K; \\ A_1 &\rightarrow p_2 A_2 \vee \bar{p}_2 A_K; \\ A_2 &\rightarrow p_3 A_3 \vee \bar{p}_3 A_K; \\ A_3 &\rightarrow A_K \\ \mathfrak{A} &= p_1 \uparrow^1 A_1 p_2 \uparrow^1 A_2 p_3 \uparrow^1 A_3 \downarrow^1 A_K. \end{aligned} \quad (5.13)$$

Процесс выявления ЛУ, которые могут быть заменены их значениями, можно формализовать. Для этого по первоначальной МСА (в рассматриваемом примере по табл. 5.9) образуют функцию f_{A_j} в виде дизъюнкции всех элементов столбца A_j :

$$f_{A_j} = \bigvee_{i=0}^{k-1} \alpha_{ij}. \quad (5.14)$$

В нашем случае

$$f_{A_1} = p_1; \quad f_{A_2} = p_1 p_2; \quad f_{A_3} = p_1 p_2 p_3;$$

$$f_{A_4} = p_1; \quad f_{A_5} = p_1 \bar{p}_2 \vee \bar{p}_1 \vee p_1 p_2 \bar{p}_3 \vee p_1 \bar{p}_2 \vee \bar{p}_1 \vee 1 = 1$$

Среди всех этих функций выявляют так называемые *сокращенные особенные функции* вида

$$f(p_1, \dots, p_n) = p_{i_1}^{\sigma_1}, \dots, p_{i_m}^{\sigma_m},$$

где

$$m \leq n, \quad p_{i_\zeta}^{\sigma_\zeta} = \begin{cases} p_{i_\zeta}, & \text{если } \sigma_\zeta = 1; \\ \bar{p}_{i_\zeta}, & \text{если } \sigma_\zeta = 0; \end{cases}$$

f' — функция, не зависящая от переменных $p_{i_1}, \dots, p_{i_m}$

В нашем случае сокращенными особенными функциями будут f_{A_1} ,

f_{A_2} и f_{A_3} . Из этих функций получим $\varphi_{A_i} = p_{i_1}, \dots, p_{i_m}$.

Из последнего выражения исключим логические условия, входящие в распределение сдвигов оператора A_i и получим функции

$$\varphi_{A_i}^* \geq \varphi_{A_i}; \quad \varphi_{A_1}^* = p_1; \quad \varphi_{A_2}^* = p_1 p_2; \quad \varphi_{A_3}^* = p_1 p_2 p_3.$$

Из этих функций видно, что в строке A_1 матрицы (5.9) переменную p_1 можно заменить на единицу, а переменную $\bar{p}_1$ — на нуль; в строке A_2

переменные p_1 и p_2 — на единицу, а переменные $\bar{p}_1$ и $\bar{p}_2$ — на нуль.

В результате такой замены получим МСА (табл. 5.11).

Если в какой-либо строке МСА стоит единица, изложенный способ может не привести к сокращению числа ЛУ, тогда как такое сокращение возможно. Для этого все единицы в матрице необходимо заменить *максимумами логических функций*.

Пусть задана МСА (табл. 5.12) и следующее распределение сдвигов: $A_0 - \{p_1, p_2\}$; $A_1 - \{p_2\}$; $A_2 - \{p_2\}$; $A_3 - \{p_2\}$; $A_4 - \{p_1, p_2\}$.

Таблица 5.12

$$\mathfrak{A} = \begin{array}{c} A_0 \\ A_1 \\ A_2 \\ A_3 \end{array} \begin{array}{cccc} A_1 & A_2 & A_3 & A_K \\ \left[\begin{array}{cccc} p_1 p_2 & & \bar{p}_1 & p_1 \bar{p}_2 \\ & 1 & & \\ & & \bar{p}_1 & p_1 \\ & & p_1 & \bar{p}_1 \end{array} \right] \end{array}$$

Образует вначале функции

$$f_{A_1} = p_1 p_2; f_{A_2} = 1 \vee p_1 = 1; f_{A_3} = \bar{p}_1; f_{A_K} = p_1 \bar{p}_2 \vee p_1 \vee \bar{p}_1 = 1,$$

а затем $\varphi_{A_1}' = p_1 p_2; \varphi_{A_3} = \bar{p}_1$ и функции

$$\varphi_{A_1}^* = p_1 p_2; \varphi_{A_3}^* = \bar{p}_1.$$

В строке A_3 переменную p_1 можно заменить на нуль, а переменную $\bar{p}_1$ — на единицу. В результате получим МСА (табл. 5.13).

Таблица 5.13

$$\mathfrak{A} = \begin{array}{c} A_0 \\ A_1 \\ A_2 \\ A_3 \end{array} \begin{array}{cccc} A_1 & A_2 & A_3 & A_K \\ \left[\begin{array}{cccc} p_1 p_2 & & \bar{p}_1 & p_1 \bar{p}_2 \\ & 1 & & \\ & & \bar{p}_1 & p_1 \\ & & & 1 \end{array} \right] \end{array}$$

Однако если умножить единичный элемент $a_{1,2}$ из табл. 5.12 на $\varphi_{A_1}^*$, то найдем $f_{A_2} = 1$ ($p_1 p_2 \vee p_1$)= p_1 . Это означает, что перед выполнением оператора A_2 логическое условие p_1 всегда имеет единичное значение, хотя для выполнения оператора A_2 после оператора A_1 его проверять нет необходимости.

Образовав теперь $\varphi_{A_2} = p_1$, видим, что в строке A_2 (табл. 5.12) переменную p_1 можно заменить на единицу, а $\bar{p}_1$ — на нуль. В результате получим следующую МСА (табл. 5.14):

Таблица 5.14

$$\mathfrak{A} = \begin{array}{c} A_0 \\ A_1 \\ A_2 \\ A_3 \end{array} \begin{array}{cccc} A_1 & A_2 & A_3 & A_K \\ \left[\begin{array}{cccc} p_1 p_2 & & \bar{p}_1 & p_1 \bar{p}_2 \\ & 1 & & \\ & & & 1 \\ & & & 1 \end{array} \right] \end{array}$$

Как видно, последняя МСА проще МСА табл. 5.13, полученной без умножения единичных элементов МСА на функции $\varphi_{A_1}^*$.

По МСА табл. 5.13 получим ЛСА

$$\mathfrak{A} = p_1 \uparrow^1 p_2 \uparrow^2 A_1 A_2 \bar{p}_1 \uparrow^2 \downarrow^1 A_3 \downarrow^2 A_k, \quad (5.15)$$

а по МСА табл. 5.14 следующую ЛСА:

$$\mathfrak{A} = p_1 \uparrow^1 p_2 \uparrow^2 A_1 A_2 \downarrow^1 A_3 \downarrow^2 A_k. \quad (5.16)$$

Как видно, ЛСА (5.16) проще ЛСА (5.15).

5.3.3. Объединение ЛСА

При задании условий работы управляющего средства в виде совокупности частных ЛСА, очевидно, в каждую из таких ЛСА могут входить одни и те же операторы и ЛУ. Поэтому когда вместо каждого символа частной ЛСА будут подставлены соответствующие выражения из операторов и ЛУ, ЛСА, описывающая общий алгоритм работы средства, может оказаться не минимальной. При этом в такой ЛСА повторяются не только ЛУ, но и операторы. Однако описанными выше способами такое повторение операторов не может быть исключено.

Для минимизации ЛСА с повторяющимися операторами и ЛУ может быть использован метод объединения ЛСА. Вначале этот метод рассмотрим для случая попарного объединения ЛСА. Произведем объединение двух частных ЛСА (5.17) и (5.18), описывающих два режима работы УС:

$$\mathfrak{A}_I = \downarrow^2 F_I p_1 \uparrow^1 p_3 \uparrow^2 \downarrow^4 F_J p_4 \uparrow^3 p_5 \uparrow^4 p_2 \uparrow^4 A_1 A_2 \omega \uparrow^2 \downarrow^3 A_3 \omega \uparrow^2 \downarrow^1; \quad (5.17)$$

$$\mathfrak{A}_{II} = \downarrow^1 F_J p_1 \uparrow^2 p_6 \uparrow^1 \downarrow^4 F_I p_1 \uparrow^3 p_7 \uparrow^4 p_2 \uparrow^4 A_1 A_2 \omega \uparrow^1 \downarrow^3 A_3 \omega \uparrow^1 \downarrow^2. \quad (5.18)$$

Общий алгоритм функционирования УС имеет вид

$$\mathfrak{A} = \downarrow^2 r \uparrow^1 \mathfrak{A}_I \omega \uparrow^2 \downarrow^1 \mathfrak{A}_{II} \omega \uparrow^2. \quad (5.19)$$

В зависимости от значения параметра r , задаваемого извне, можно настраивать УС на выполнение A_1 или A_{11} . Если подставить в ЛСА (5.19) выражения для A_1 , A_{11} , то получим ЛСА с 21 членом (как и ранее, здесь не учитываем число тождественно-ложных условий).

Процесс построения объединенной ЛСА, в которой операторы не повторяются, состоит в следующем. По МСА двух частных ЛСА составляется МСА объединенной ЛСА, каждый элемент которой

$$\alpha_{ij} = \bigvee_{l=1,2} \alpha_{ij}^l \beta_i^l, \quad (5.20)$$

где β_i^l — определяющая функция, равная функции $f(r_1 \dots r_n)$, принимающей единичное значение на наборе значений переменных

$r_1 \dots \dots, r_n$, который сопоставлен с ЛСА A_i . Если оператор A_i не входит в ЛСА A_i , то в функцию β_i^I соответствующий набор значений

$r_1 \dots \dots, r_n$ войдет в качестве условного.

Если число наборов значений переменных $r_1 \dots \dots, r_n$ больше числа объединяемых ЛСА, то в определяющую функцию могут входить конъюнкции, которые принимают единичные значения на неиспользуемых наборах.

В нашем случае для всех операторов ЛСА A_i $\beta_i^I=r$, а для всех операторов ЛСА A_{II} $\beta_i^{II}=\bar{r}$, так как в A_I и A_{II} входят все операторы.

Составим предварительно МСА, равносильные ЛСА (5.17) и (5.18), в виде табл. 5.15 и 5.16:

Таблица 5.15

	A_1	A_2	A_3	F_i	F_j	A_K
A_0				1		
A_1		1				
A_2				1		
A_3				1		
F_i				$p_1 \bar{p}_3$	$p_1 p_3$	$\bar{p}_1$
F_j	$p_2 p_4 p_5$		$\bar{p}_4$		$\bar{p}_2 p_4 p_5 \vee p_4 \bar{p}_5$	

Таблица 5.16

	A_1	A_2	A_3	F_i	F_j	A_K
A_0					1	
A_1		1				
A_2					1	
A_3					1	
F_i	$p_1 p_2 p_7$		$\bar{p}_1$	$p_1 \bar{p}_2 p_7 \vee p_1 \bar{p}_7$		
F_j				$p_4 p_6$	$p_4 \bar{p}_6$	$\bar{p}_4$

После этого по указанному выше правилу получим следующую МСА, равносильную объединенной ЛСА (табл. 5.17):

Таблица 5.17

$$\mathfrak{A} = \begin{array}{c|cccccc} & A_1 & A_2 & A_3 & F_i & & F_j \\ \hline A_0 & & & & r & & \bar{r} \\ A_1 & & r\sqrt{r}=1 & & & & \\ A_2 & & & & r & & \bar{r} \\ A_3 & & & & r & & \bar{r} \\ F_i & p_1 p_2 p_7 \bar{r} \sqrt{p_1 r} & \bar{p}_1 r & p_1 \bar{p}_3 r \sqrt{p_1 \bar{p}_2 p_7 r} \sqrt{p_1 \bar{p}_7 r} & & & p_1 p_3 r \\ F_j & p_2 p_4 p_5 r \sqrt{p_4 r} & \bar{p}_4 r & p_4 p_6 r & & & \bar{p}_2 p_4 p_5 r \sqrt{p_4 \bar{p}_5 r} \sqrt{p_4 \bar{p}_6 r} \end{array}$$

От МСА (табл. 5.17) перейдем к системе формул перехода, после чего преобразуем ее в систему скобочных формул перехода, где объединим общие выражения:

$$\left\{ \begin{array}{l} A_0 \rightarrow \downarrow {}^{\alpha} r F_i \vee \bar{r} F_j; \\ A_1 \rightarrow A_2; \\ A_2 \rightarrow r F_i \vee \bar{r} F_j = \omega \uparrow {}^{\alpha}; \\ A_3 \rightarrow r F_i \vee \bar{r} F_j = \omega \uparrow {}^{\alpha}; \\ F_i \rightarrow p_1 p_2 p_7 \bar{r} A_1 \vee \bar{p}_1 r A_1 \vee \bar{p}_1 \bar{r} A_3 \vee p_1 \bar{p}_3 r F_i \vee \\ \vee p_1 \bar{p}_2 p_7 \bar{r} F_i \vee p_1 \bar{p}_7 \bar{r} F_i \vee p_1 p_3 r F_j = p_1 (r (p_3 F_j \vee \bar{p}_3 F_i) \vee \\ \vee \bar{r} (p_7 (p_2 A_1 \vee \bar{p}_2 F_i) \vee \bar{p}_7 F_i)) \vee \bar{p}_1 (r A_1 \vee \bar{r} A_3) = \\ = p_1 (r (p_3 F_j \vee \bar{p}_3 F_i) \vee \bar{r} (p_7 \downarrow {}^{\beta} (p_2 A_1 \vee \bar{p}_2 F_i) \vee p_7 F_i)) \vee \\ \vee \bar{p}_1 (r A_1 \vee \bar{r} A_3); \\ F_j \rightarrow p_2 p_4 p_5 r A_1 \vee \bar{p}_4 \bar{r} A_1 \vee \bar{p}_4 r A_3 \vee p_4 p_6 \bar{r} F_i \vee \\ \vee \bar{p}_2 p_4 p_5 r F_j \vee p_4 \bar{p}_5 r F_j \vee p_4 \bar{p}_6 \bar{r} F_j = \\ = r (p_4 (p_5 (p_2 A_1 \vee \bar{p}_2 F_j) \vee \bar{p}_5 F_j) \vee \bar{p}_4 A_3) \vee \bar{r} (p_4 (p_6 F_i \vee \\ \vee \bar{p}_6 F_j) \vee \bar{p}_4 A_1) = r (p_4 (p_5 \omega \uparrow {}^{\beta} \vee \bar{p}_5 F_j) \vee \\ \vee \bar{p}_4 A_3) \vee \bar{r} (p_4 (p_6 F_i \vee \bar{p}_6 F_j) \vee \bar{p}_4 A_1). \end{array} \right.$$

Теперь нетрудно получить и объединенную ЛСА:

$$\mathfrak{A} = \downarrow {}^1 r \uparrow {}^2 \downarrow {}^6 F_i p_1 \uparrow {}^3 r \uparrow {}^4 p_3 \uparrow {}^5 \downarrow {}^2 F_j r \uparrow {}^7 p_4 \uparrow {}^8 p_5 \uparrow {}^2 \omega \uparrow {}^9 \downarrow {}^3 r \uparrow {}^8 \downarrow {}^5 A_1 A_2 \omega \uparrow {}^1 \downarrow {}^4 p_7 \uparrow {}^6 \downarrow {}^9 p_2 \uparrow {}^6 \omega \uparrow {}^5 \downarrow {}^7 p_4 \uparrow {}^5 p_6 \uparrow {}^2 \omega \uparrow {}^6 \downarrow {}^8 A_3 \omega \uparrow {}^1, \quad (5.21)$$

в которой имеется 17 членов вместо 21 в ЛСА (5.19).

Если в ЛСА (5.21) подставить значение $r=1$, то можно легко убедиться, что получим ЛСА, равносильную ЛСА (5.17), а при подстановке $r=0$ получим ЛСА, равносильную ЛСА (5.18).

Очевидно, при объединении L ЛСА вместо одной переменной потребуется n переменных $r_1 \dots, r_n$, $2^n \geq L$, так как каждой из L ЛСА сопоставляется своя, отличная от других конъюнкция

$$R_i = r_1^{\beta_i^1}, \dots, r_n^{\beta_i^n},$$

называемая *определяющей конъюнкцией*.

Таким образом, объединенной ЛСА **A** называют ЛСА, удовлетворяющую следующим условиям:

- 1) если оператор A_i входит хотя бы в одну из частных ЛСА A_j , то он обязательно входит в ЛСА **A**, причем только один раз (если в каждой из частных ЛСА нет повторений одинаковых операторов);
- 2) при подстановке набора значений переменных $r_1 \dots, r_n$, на котором $R_j=1$, ЛСА **A** превращается в ЛСА, равносильную частной ЛСА A_j .

Предполагая наличие только неповторяющихся операторов в каждой из частных ЛСА, получим, что каждый оператор в объединенную ЛСА входит только один раз. Однако в различных частных ЛСА после одного и того же оператора могут выполняться различные члены ЛСА. Поэтому каждый раз при выполнении оператора объединенной ЛСА необходимо знать, в какой из частных ЛСА он в данный момент выполняется.

Проверка значений переменных $r_1 \dots, r_n$, осуществляемая после выполнения оператора, входящего в различные частные ЛСА, позволяет выбрать тот член объединенной ЛСА, который должен выполняться в частной ЛСА, соответствующей данному набору значений переменных $r_1 \dots, r_n$.

Нетрудно понять, что если оператор входит только в одну из L ЛСА, то

$$\beta_i^t = R_i \bigvee \frac{R_{j_1}}{0} \bigvee \dots \bigvee \frac{R_{i_{2n}}}{0} = 1.$$

При объединении L ЛСА определяющая функция может быть недоопределенной, поэтому из (5.20) следует, что элементы матрицы объединенной ЛСА могут быть недоопределены.

Получаемая при объединении L ЛСА матрица с такими недоопределенными элементами является недетерминированной МСА $\hat{A}$.

Легко понять, что недетерминированная МСА переводится в МСА при подстановке вместо переменных $r_1 \dots, r_n$ в β_i^l набора значений $\sigma_1^l, \dots, \sigma_n^l$, на котором

$$R_i = r_1^{\sigma_1^l}, \dots, r_n^{\sigma_n^l} = 1.$$

Если с каждой из L объединяемых ЛСА сопоставлена лишь одна определяющая конъюнкция, а $L < 2^n$, то $2^n - L$ конъюнкций переменных $r_1 \dots, r_n$ останутся неиспользуемыми. Для того чтобы объединенная ЛСА обладала свойством детерминированности, эти конъюнкции должны быть переведены в используемые. Например, их можно сопоставить с одной из ЛСА или потребовать, чтобы на соответствующих наборах значений переменных $r_1 \dots, r_n$ выполнялась бы «пустая» ЛСА вида $A = A_k$. Однако такое доопределение приводит в большинстве случаев к неоптимальной ЛСА. Оказывается более целесообразным составить неиспользуемые ЛСА, которые могут быть и неравносильны ни одной из используемых частных ЛСА. Построение таких неиспользуемых ЛСА целесообразно производить не сразу, а в процессе получения объединенной ЛСА, т. е. здесь задача в каком-то смысле аналогична задаче синтеза контактной схемы или схемы из функциональных элементов по не полностью определенным булевым функциям.

Таким образом, как и при объединении двух ЛСА, в данном случае по частным ЛСА составляются МСА. Затем выявляются определяющие функции β_i^l , которые в данном случае могут быть и недоопределенными. После этого в соответствии с (5.20) построим недетерминированную МСА. По недетерминированной МСА получим недоопределенные формулы перехода.

В процессе вынесения за скобки переменных, как и в случае учета неиспользуемых наборов значений логических условий, недоопределенные формулы перехода переводим в определенные скобочные формулы перехода, причем доопределение формул перехода следует делать так, чтобы обеспечить максимальное удовлетворение требований к порядку вынесения переменных за скобки и выявлению общих выражений, при выполнении которых может быть построена ЛСА с минимальным (или близким к минимальному) числом членов. После этого, как и ранее, переходим к объединенной ЛСА.

Рассмотрим пример объединения трех следующих ЛСА:

$$\mathfrak{A}_1 = \downarrow^2 p_1 \uparrow^1 A_1 p_2 \uparrow^2 A_2 \downarrow^3 A_3 p_3 \uparrow^3 A_4 \downarrow^1 A_k; \quad (5.22)$$

$$\mathfrak{A}_2 = p_2 \uparrow^1 A_2 \downarrow^2 A_3 p_3 \uparrow^2 A_4 p_1 \uparrow^3 \downarrow^1 A_1 \downarrow^3 A_k; \quad (5.23)$$

$$\mathfrak{A}_3 = A_1 p_2 \uparrow^1 A_3 A_4 \downarrow^1 A_k. \quad (5.24)$$

Требуется построить объединенную ЛСА; так как $L=3$, число дополнительных переменных равно двум (дополнительные логические переменные r_1 и r_2). Очевидно, при этом одна определяющая конъюнкция будет неиспользуемой.

Если не осуществлять объединение заданных ЛСА, а получить общий алгоритм функционирования в соответствии с ЛСА, например, вида

$$\mathfrak{A} = r_1 \uparrow^1 \mathfrak{A}_1 \omega \uparrow^2 \downarrow^1 r_2 \uparrow^3 \mathfrak{A}_2 \omega \uparrow^2 \downarrow^3 \mathfrak{A}_3 \downarrow^2 A_K, \quad (5.25)$$

то общее число членов будет равно 21.

Для объединения ЛСА (5.21) — (5.24) перейдем к МСА (табл. 5.18—5.20).

Таблица 5.18

	A_1	A_2	A_3	A_4	A_K
A_0	$\overline{p_1}$				$\overline{p_1}$
A_1	$p_1 \overline{p_2}$	p_2			$\overline{p_1 p_2}$
A_2			1		
A_3			$\overline{p_3}$	p_3	
A_4					1

Таблица 5.19

	A_1	A_2	A_3	A_4	A_K
A_0	$\overline{p_2}$	p_2			
A_1				1	
A_2			1		
A_3			$\overline{p_3}$	p_3	
A_4	$\overline{p_1}$				$\overline{p_1}$

Таблица 5.20

	A_1	A_3	A_4	A_K
A_0	1			
A_1		p_2		$\overline{p_2}$
A_3			1	
A_4				1

Теперь выберем определяющие конъюнкции R_1 — R_3 . Соседние определяющие конъюнкции должны быть приписаны тем парам ЛСА, для которых число совпадающих элементов МСА является максимальным. При этом упростится наибольшее число элементов объединенной МСА.

В соответствии с этим условием приписывания определяющих конъюнкций подсчитываем число одинаковых элементов различных пар МСА. При этом будем подсчитывать число совпадающих элементов между каждой из всех четырех МСА.

Так как элементы неиспользуемой МСА не определены, то при сравнении с ней число совпадающих элементов берется равным числу элементов используемой МСА, не равных нулю.

Кроме того, заметим, что если из сравнения двух используемых МСА выяснится, что оператор A_i входит лишь в МСА A_j , а в МСА A_l не содержится, то в число совпадающих элементов МСА A_j и A_l входят все ненулевые элементы строки A_l МСА A_j .

Для наглядности связи между МСА изобразим в виде графа, каждому ребру которого припишем цифру, указывающую число одинаковых элементов у МСА, приписанных вершинам, соединенным этим ребром. Для нашего примера такой отмеченный граф изображен на рис. 5.30, где A_n — вершина, соответствующая неиспользуемой МСА.

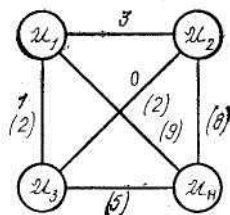


Рис. 5.30.

Тем парам МСА, которым соответствует наибольшее число совпадающих элементов, должны быть приписаны соседние определяющие конъюнкции. Надо заметить, что такой способ выявления определяющих конъюнкций не гарантирует построения объединенной ЛСА с минимальным числом членов. Однако он позволяет получить вполне приемлемые решения. Кроме того, надо иметь в виду, что наибольшее упрощение может быть получено в том случае, когда соседние определяющие конъюнкции приписываются тем парам МСА, у которых одинаковыми являются достаточно сложные элементы. Поэтому целесообразно в первую очередь приписывать таким парам МСА соседние определяющие конъюнкции, если даже у них число одинаковых элементов меньше, чем у других пар МСА.

В рассматриваемом примере таких сложных одинаковых элементов нет, поэтому выявление определяющих конъюнкций осуществлено в соответствии, с графом, показанным на рис. 5.30, т. е.

$$R_1=r_1r_2; R_2=r_1\bar{r}_2; R_3=\bar{r}_1r_2; R_4=\bar{r}_1\bar{r}_2,$$

где R_4 —неиспользуемая определяющая конъюнкция. Теперь получим определяющие функции β_i^j :

$$\beta_0^1=\beta_1^1=\beta_3^1=\beta_4^1=r_1r_2\bigvee\frac{\bar{r}_1\bar{r}_2}{0}=r_1r_2;$$

$$\beta_2^1=r_1r_2\bigvee\frac{\bar{r}_1\bar{r}_2}{0}\bigvee\frac{\bar{r}_1r_2}{0}=\frac{r_1}{1}r_2;$$

$$\beta_0^2=\beta_1^2=\beta_3^2=\beta_4^2=r_1\bar{r}_2\bigvee\frac{\bar{r}_1\bar{r}_2}{0}=\frac{r_1}{1}\bar{r}_2;$$

$$\beta_2^2=r_1r_2\bigvee\frac{\bar{r}_1\bar{r}_2}{0}\bigvee\frac{\bar{r}_1r_2}{0}=\frac{r_1}{1}r_2;$$

$$\beta_0^3=\beta_1^3=\beta_3^3=\beta_4^3=\bar{r}_1r_2\bigvee\frac{\bar{r}_1\bar{r}_2}{0}=\bar{r}_1\frac{r_2}{1}.$$

Строим недетерминированную МСА (табл. 5.21):

Таблица 5.21
A_к

	A ₁	A ₂	A ₃	A ₄	
A ₀	$r_1 r_2 p_1 \sqrt{\frac{r_1}{1} \bar{r}_2 \bar{p}_2} \sqrt{\frac{r_1}{1} \bar{r}_2 p_2}$	$\frac{r_1}{1} \bar{r}_2 p_2$			$r_1 r_2 p_1$
	$\sqrt{\frac{r_2}{1} \bar{r}_1}$				
A ₁	$r_1 r_2 p_1 \bar{p}_2$	$r_1 r_2 p_2$	$\bar{r}_1 \frac{r_2}{1} \bar{p}_2$		$r_1 r_2 p_1 \bar{p}_2 \sqrt{\frac{r_1}{1} \bar{r}_2 \bar{p}_2} \sqrt{\frac{r_1}{1} \bar{r}_2 \bar{p}_2}$
$\hat{A}_2 = A_2$			$\frac{r_1}{1} \bar{r}_2 \sqrt{\frac{r_2}{1} \bar{r}_1}$		
			$\sqrt{\frac{r_1}{1} \bar{r}_2} = 1$		
A ₃			$r_1 r_2 p_3 \sqrt{\frac{r_1}{1} \bar{r}_2} \times$	$r_1 r_2 p_3 \sqrt{\frac{r_1}{1} \bar{r}_2} \times$	
			$\times \bar{r}_2 \bar{p}_3 =$	$\times \bar{r}_2 \bar{p}_3 \sqrt{\frac{r_1}{1} \bar{r}_2} \times$	
			$= r_1 \bar{p}_3$	$\times \frac{r_2}{1} =$	
				$= r_1 p_3 \sqrt{\frac{r_1}{1} \bar{r}_2}$	
A ₄	$\frac{r_1}{1} \bar{r}_2 p_1$				$r_1 r_2 \sqrt{\frac{r_1}{1} \bar{r}_2 \bar{p}_1} \sqrt{\frac{r_1}{1} \bar{r}_2 \bar{p}_1}$
					$\sqrt{\frac{r_1}{1} \bar{r}_2} = r_2 \sqrt{\frac{r_1}{1} \bar{r}_2 \bar{p}_1}$
					$\sqrt{\frac{r_1}{1} \bar{r}_2 \bar{p}_1}$

От недетерминированной МСА переходим к недоопределенным формулам перехода, которые затем преобразовываем:

$$\begin{aligned}
 A_0 &\rightarrow r_1 r_2 p_1 A_1 \vee \frac{r_1}{1} \bar{r}_2 \bar{p}_2 A_1 \vee \bar{r}_1 \frac{r_2}{1} A_1 \vee \frac{r_1}{1} \bar{r}_2 p_2 A_2 \vee r_1 r_2 \bar{p}_1 A_K = \\
 &= r_1 (r_2 \downarrow^1 (p_1 A_1 \vee \bar{p}_1 A_K) \vee \bar{r}_2 (p_2 A_2 \vee \bar{p}_2 A_1)) \vee \bar{r}_1 A_1; \\
 A_1 &\rightarrow r_1 r_2 p_1 \bar{p}_2 A_1 \vee r_1 r_2 p_2 A_2 \vee \bar{r}_1 \frac{r_2}{1} p_2 A_3 \vee r_1 r_2 \bar{p}_1 \bar{p}_2 A_K \vee \frac{r_1}{1} \bar{r}_2 A_K \vee \\
 &\vee \bar{r}_1 \frac{r_2}{1} \bar{p}_2 A_K = r_1 (r_2 (p_2 A_1 \vee \bar{p}_2 (p_1 A_2 \vee \bar{p}_1 A_K) \vee \bar{r}_2 A_K)) \vee \bar{r}_1 (p_2 A_3 \vee \bar{p}_2 A_K) = \\
 &= r_1 (r_2 (p_2 A_2 \vee \bar{p}_2 \omega \uparrow^1) \vee \bar{r}_2 A_K) \vee \bar{r}_1 (p_2 A_3 \vee \bar{p}_2 A_K); \\
 &A_2 \rightarrow A_3; \\
 A_3 &\rightarrow r_1 \bar{p}_3 A_3 \vee r_1 p_3 A_4 \vee \bar{r}_1 A_4 = r_1 (\bar{p}_3 A_3 \vee p_3 A_4) \vee \bar{r}_1 A_4; \\
 A_4 &\rightarrow \frac{r_1}{1} \bar{r}_2 p_1 A_1 \vee r_2 A_K \vee \frac{r_1}{1} \bar{r}_2 \bar{p}_1 A_K = r_2 A_K \vee \bar{r}_2 (p_1 A_1 \vee \bar{p}_1 A_K) = \\
 &= r_2 A_K \vee \bar{r}_2 \omega \uparrow^1.
 \end{aligned}$$

После этого составляем объединенную ЛСА:

$$\begin{aligned}
 \mathfrak{A} &= r_1 \uparrow^1 r_2 \uparrow^2 \downarrow^6 p_1 \uparrow^7 \downarrow^1 A_1 r_1 \uparrow^4 r_2 \uparrow^7 p_2 \uparrow^6 \downarrow^3 A_2 \downarrow^5 A_3 r_1 \uparrow^8 p_3 \uparrow^5 \downarrow^8 A_4 r_2 \uparrow^6 \omega \uparrow^7 \downarrow^2 p_2 \uparrow^1 \omega \uparrow^3 \\
 &\downarrow^4 p_2 \uparrow^7 \omega \uparrow^5 \downarrow^7 A_K,
 \end{aligned} \tag{5.26}$$

в которой содержится 16 членов вместо 21 в ЛСА (5.25).

В рассмотренном примере при определенных значениях дополнительных ЛУ выполняется одна из частных ЛСА, после чего процесс заканчивается.

Значения дополнительных логических условий r_1 и r_2 при этом должны быть заданы или каким-либо образом выбраны. Однако возможны случаи, когда частные ЛСА выполняются одна за другой и после выполнения всех ЛСА процесс повторяется. Тогда общий алгоритм функционирования УС запишем в виде

$$\mathfrak{A} = \downarrow^1 \mathfrak{A}_1 \mathfrak{A}_2 \dots \mathfrak{A}_L \omega \uparrow^1, \tag{5.27}$$

при этом изменение значений дополнительных логических условий должно производиться в процессе выполнения общей ЛСА, а для формирования соответствующих их значений могут быть введены новые операторы

$$F_{\mathfrak{A}_1}, F_{\mathfrak{A}_2}, \dots, F_{\mathfrak{A}_L},$$

каждый из которых является одновременно оператором начала соответствующей ЛСА.

Таким образом, в каждой из частных ЛСА вместо оператора A_0 будет оператор F_{A_i} , который вместе с тем будет оператором конца ЛСА A_{i-1} . Теперь

$$\mathfrak{A} = \downarrow^1 F_{\mathfrak{A}_1} \mathfrak{A}_1 F_{\mathfrak{A}_2} \mathfrak{A}_2 \dots F_{\mathfrak{A}_L} \mathfrak{A}_L \omega \uparrow^1. \quad (5.28)$$

В ЛСА (5.28) порядок выполнения частных ЛСА единственный, однако может быть задан и некоторый алгоритм над частными ЛСА, так что допускается несколько различных последовательностей выполнений частных ЛСА. Тогда в общей ЛСА имеются внешние ЛУ, значения которых определяют эти последовательности. При объединении ЛСА эти условия входят в МСА объединенной ЛСА. Хотя объединение частных ЛСА не всегда приводит к сокращению числа членов, но в большинстве практических случаев объединенная ЛСА содержит значительно меньшее число членов, чем общая ЛСА, в которой частные ЛСА не объединены.

5.3.4. Описание параллельных облачных управляющих алгоритмов

Стремление к повышению производительности УС и его многоблочная структура часто приводят к необходимости распараллеливания управляющего алгоритма. При этом языки задания алгоритма функционирования УС должен обладать средствами для описания параллелизма, которых, очевидно, не имеет язык ЛСА.

Предложено расширение языка ЛСА — язык ЛСАII, который позволяет описывать алгоритмы функционирования с параллельными участками. Запись на языке ЛСАII может иметь, например, следующий вид:

$$\mathfrak{A} = A_0 p_1 \uparrow^1 A_1 \downarrow^1 \left[\begin{array}{l} \rightarrow A_2 A_3 \rightarrow \left[\begin{array}{l} A_7 p_3 \uparrow^3 A_8 \downarrow^3 \\ A_9 p_4 \uparrow^4 A_{10} \downarrow^4 \end{array} \right] \\ \rightarrow p_2 \uparrow^2 A_4 \downarrow^2 \\ \rightarrow A_5 A_6 \end{array} \right] \rightarrow p_5 \uparrow^5 A_{11} \downarrow^5 \rightarrow A_{12} A_k. \quad (5.29)$$

Такая запись означает, что после выполнения оператора A_1 или при ложном значении логического условия p_1 начинается одновременное выполнение трех участков ЛСА. Затем после выполнения операторов A_4 и A_6 два участка соединяются, а после выполнения оператора A_3 алгоритм опять распараллеливается и т. д.

Этот язык удобен своей наглядностью, однако он недостаточно формализован, что ограничивает его использование для выполнения различных преобразований алгоритма функционирования. Кроме того, являясь фактически сочетанием логических и граф-схем алгоритмов, язык ЛСАП может привести к громоздкому описанию сложных алгоритмов.

Теперь рассмотрим другое расширение языка ЛСА — язык *параллельных логических схем алгоритмов*, в котором с помощью введения в язык ЛСА специальных операторов и дополнительных символов сохранена линейная запись алгоритма с параллельными участками.

Для расширения возможностей языка ЛСА введем *оператор распараллеливания* R . Оператор R подобно оператору A_0 (начало алгоритма) является фиктивным, он лишь символизирует переход к выполнению параллельных частей алгоритма. Поскольку после оператора R начинается выполнение параллельных участков алгоритма, этот оператор в отличие от других операторов ЛСА имеет несколько последователей, которые являются начальными членами параллельных участков. Для перехода к параллельным участкам введем специальные стрелки $\uparrow_j^i \downarrow^i$, которые отличаются от имеющихся в языке ЛСА *стрелок разветвления* $\uparrow \downarrow$ — *стрелки распараллеливания*. Нижний индекс j стрелки распараллеливания определяет число параллельных участков, следующих за данным оператором распараллеливания.

Справа от оператора распараллеливания R будем записывать тождественно-ложное логическое условие ω со стрелкой распараллеливания $\uparrow_j^i (\omega \uparrow_j^i)$. Концы i -й стрелки $\downarrow^i$ записываются перед первыми членами параллельных участков, которые начинаются после данного оператора распараллеливания. Очевидно, что число таких стрелок равно j . Заметим, что нижний индекс стрелки распараллеливания не определяет последовательность выполнения отдельных шагов алгоритма, а играет лишь вспомогательную роль и введен для проверки правильности записи алгоритма. Справа от каждой стрелки $\downarrow^i$ последовательно, как в языке ЛСА, записываются члены каждого параллельного участка.

Таким образом, с помощью оператора и стрелок распараллеливания можно осуществить переход к параллельным участкам алгоритма.

Для соединения параллельных участков введем *оператор соединения* S и *стрелки соединения* $\uparrow^k \downarrow_m^k$. После последнего члена

каждого параллельного участка запишем тождественно-ложное условие ω со стрелкой соединения $\uparrow^k(\omega \uparrow^k)$.

Конец k -й стрелки соединения $\downarrow^k$ записывается перед оператором соединения S , в котором соединяются данные параллельные участки. Нижний индекс стрелки соединения m играет вспомогательную роль, указывая число параллельных участков, которые соединяются в данном операторе соединения, т. е. число стрелок $\uparrow^k$.

Оператор S , как и оператор R , является фиктивным, символизирующим конец выполнения параллельных участков. Оператор соединения S может быть выполнен только после того, как закончится выполнение всех соединяющихся в нем параллельных участков. Справа от оператора S записывается первый член последовательного участка алгоритма, который должен выполняться после окончания предшествующих параллельных участков.

Таким образом, использование операторов и стрелок соединения позволяет осуществлять переход от параллельных участков алгоритма к последовательным.

Логическую схему алгоритма, содержащую кроме обычных операторов, логических условий и стрелок разветвления также операторы и стрелки распараллеливания и соединения, будем называть *параллельной логической схемой алгоритма* (ПЛСА).

Будем рассматривать такие ПЛСА, в которых каждый параллельный участок является замкнутым, т. е. имеет только один вход и один выход. Каждый такой участок можно считать отдельным подалгоритмом, введя в него фиктивные операторы, символизирующие начало и окончание выполнения этого подалгоритма, подобно операторам A_0 и A_k в ЛСА. Для i -го подалгоритма обозначим эти операторы B_0^i и B_k^i соответственно (нумерация самих подалгоритмов не имеет значения). При этом приведенная ранее ЛСАП (5.29) может быть представлена в виде следующей ПЛСА:

$$\begin{aligned} \mathfrak{A} = & A_0 B_0^0 p_1 \uparrow^1 A_1 \downarrow^1 B_0^1 R_1 \omega \uparrow^6 \downarrow^6 B_0^1 A_2 A_3 B_k^1 R_2 \omega \uparrow^2 \downarrow^2 B_0^4 A_7 \\ & p_3 \uparrow^3 A_8 \downarrow^3 B_k^4 \omega \uparrow^8 \downarrow^8 B_0^5 A_9 p_4 \uparrow^4 A_{10} \downarrow^4 B_k^5 \omega \uparrow^8 \downarrow^8 B_0^2 \\ & p_2 \uparrow^2 A_4 \downarrow^2 B_k^2 \omega \uparrow^9 \downarrow^9 B_0^3 A_5 A_6 B_k^3 \omega \uparrow^9 \downarrow^9 S_1 B_0^6 p_5 \\ & \uparrow^5 A_{11} \downarrow^5 B_k^6 \omega \uparrow^8 \downarrow^8 S_2 A_{12}. \end{aligned} \quad (5.30)$$

Здесь фиктивные операторы B_0 и B_k введены во все подалгоритмы, так что каждый из них не содержит ни операторов, ни стрелок распараллеливания и соединения и описывается своей ЛСА. При этом

ПЛСА может быть записана в более компактной форме с использованием обозначений соответствующих подалгоритмов:

$$\mathfrak{A} = \mathfrak{A}_0 R_1 \omega \uparrow_3^6 \uparrow^8 \mathfrak{A}_1 R_2 \omega \uparrow_2^7 \downarrow^7 \mathfrak{A}_4 \omega \uparrow^8 \downarrow^7 \mathfrak{A}_5 \omega \uparrow^8 \downarrow^6 \mathfrak{A}_2 \omega \uparrow^9 \downarrow^6 \mathfrak{A}_3 \omega \uparrow^9 \downarrow_2^9 S_1 \mathfrak{A}_6 \omega \uparrow^8 \downarrow_3^8 S_2 A_{12}. \quad (5.31)$$

Для каждого из подалгоритмов можно составить МСА и проверить такие важные свойства, как полноту и непротиворечивость. По аналогии с МСА может быть введена и параллельная матричная схема алгоритма (ПМСА). Параллельная МСА также является квадратной матрицей, строки и столбцы которой соответствуют операторам ПЛСА. Все элементы ПМСА имеют тот же смысл, что и в МСА, за исключением строк операторов распараллеливания и столбцов операторов соединения. Строка оператора R содержит несколько единичных элементов, число которых равно нижнему индексу стрелки распараллеливания этого оператора в ПЛСА. Очевидно, что формула перехода оператора распараллеливания отличается от формул перехода других операторов. Иначе чем для других столбцов ПМСА необходимо трактовать наличие нескольких единичных элементов в столбце оператора соединения. Они определяют всех предшественников, которые должны быть выполнены для перехода к этому оператору соединения. Для других операторов необходимо выполнение лишь одного из возможных предшественников.

Такие же особенности, вытекающие из свойств параллельного алгоритма, характерны и для параллельной граф-схемы алгоритма (ПГСА). Вершина графа, соответствующая оператору R , имеет несколько последователей в отличие от всех других операторов. В вершине, соответствующей оператору S , параллельно соединяется несколько путей графа в отличие от альтернативного соединения в других его вершинах. На рис. 5.31 приведен пример ПГСА для ПЛСА (5.31), подалгоритм A_4 представлен в терминах операторов и ЛУ в соответствии с ПЛСА (5.30).

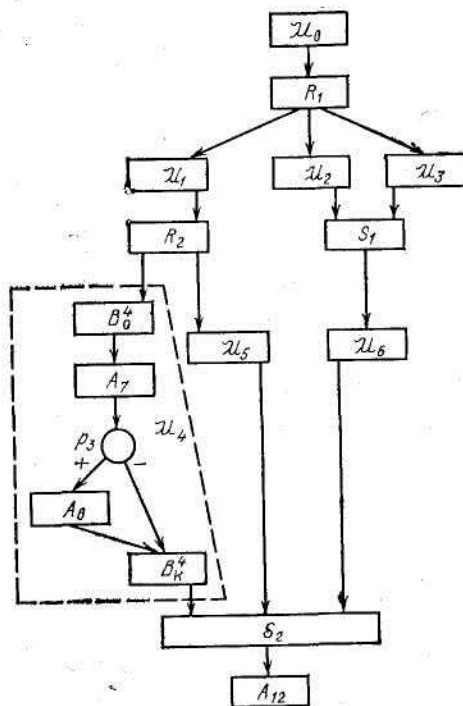


Рис. 5.31.

Здесь в операторе B_k^4 альтернативное соединение ветвей алгоритма, тогда как в S_1 и S_2 — параллельное.

Ясно, что для сложного параллельного управляющего алгоритма ПМСА и ПГСА оказываются очень громоздкими, что затрудняет их использование для проверки и преобразования алгоритма.

5.3.5. Переход от правильного облачного управляющего процесса к облачному управляющему алгоритму

В предыдущих разделах показано, что ряд задач по преобразованию и упрощению облачного управляющего алгоритма удобно решать с использованием языка ЛСА. В то же время управляющий процесс, реализуемый в УС, с учетом функциональных и ЛР, необходимых для выполнения процедур процесса, задается на языке сетей Петри. Очевидно, для использования различных языков на

разных этапах разработки УС необходимо установить связь между ЛСА, ПЛСА и сетями Петри.

Осуществим вначале переход от ЛСА к сети Петри, описывающей тот же последовательный процесс. Заметим, что возможны различные интерпретации ЛСА в терминах языка сетей Петри. Рассмотрим одну из них, основанную на использовании МСА.

Первой строке A_0 МСА сопоставим переход t_0^0 сети Петри, входной позицией которого является позиция a_0 . Появление метки в позиции a_0 , вызывающее реализацию перехода t_0^0 , соответствует началу выполнения алгоритма. После реализации перехода t_0^0 появляется метка в его выходной позиции, которая является входной для перехода t_1^l , если в строке оператора A_0 МСА заполнена только одна клетка $\alpha_{0,j}$ (очевидно, она содержит 1). Если оператор A_0 имеет несколько последователей, например A_l, A_s, A_q , т. е. в МСА заполнены клетки $\alpha_{0,l}, \alpha_{0,s}, \alpha_{0,q}$, выходная позиция перехода t_0^0 является позицией альтернативного разветвления. Она будет общей входной позицией переходов t_1^l, t_2^s, t_3^q , которые помечаются конъюнкциями, записанными в соответствующих клетках МСА. Обратившись затем к строке оператора A_l и определив его последователей, можно таким же образом построить фрагмент сети Петри для перехода t_1^l . Процесс построения сети Петри по МСА заканчивается после того, как будут рассмотрены все строки матрицы и введены все переходы t_j^k , число которых должно быть равно числу заполненных клеток столбца A_k МСА. Ясно, что эти переходы не имеют выходных позиций. Если для ЛСА (МСА) было задано распределение сдвигов ее операторов, эта информация должна быть сохранена и для сети Петри введением в нее позиций ЛР. На рис. 5.32 показан граф сети Петри, построенный по МСА (табл. 5.12) в соответствии с приведенными правилами (без учета распределения сдвигов).

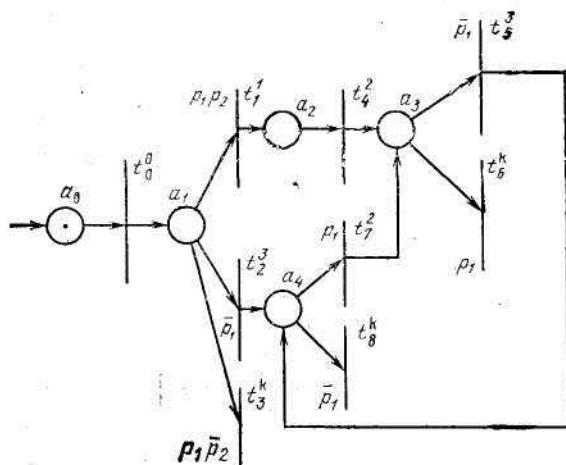


Рис. 5.32.

Как видно, число переходов получаемой сети Петри на единицу больше числа заполненных клеток МСА, а число позиций равно числу различных операторов, так что позиция a_i получает метку после реализации перехода t_j^{l-1} . Переходы-последователи позиции a_i определяются формулой перехода оператора A_{i-1} , т. е. строкой A_{i-1} МСА. Если оператор A_{i-1} имеет несколько предшественников, позиция a_i является позицией альтернативного соединения.

Аналогичным образом можно осуществить переход от ПЛСА к сети Петри, сопоставив с каждым оператором R переход распараллеливания, число выходных позиций которого равно числу единиц в строке ПМСА этого оператора, а с оператором S переход соединения с числом входных позиций, равным числу единиц в столбце ПМСА этого оператора. Для каждого подалгоритма ПЛСА сеть Петри строится так же, как для ЛСА. Очевидно, при этом сеть Петри может содержать избыточное число переходов и позиций из-за большого числа введенных в ПЛСА фиктивных операторов начала и окончания подалгоритмов. Поскольку эти операторы были введены специально, чтобы иметь одинаковые правила перехода от ЛСА и ПЛСА к сети Петри, соответствующие им переходы и позиции могут быть затем исключены из сети.

Рассмотрим способ перехода от сети Петри, описывающей правильный ОУП, к ПЛСА или ЛСА, если процесс является последовательным. Сравнение языков сетей Петри и ПЛСА

показывает, что существенным отличием сетей Петри является их способность описывать конфликтные состояния. Благодаря этому сетью Петри можно задать гораздо больше последовательностей выполнения процедур процесса, чем ПЛСА, которая описывает параллелизм, но не обладает средствами для задания конфликтов. Поэтому при переходе от сети Петри к ПЛСА необходимо либо разрешить конфликты путем проверок значений некоторых дополнительно введенных переменных, либо удалить позиции конфликтов, к которым относятся позиции ресурсов и блокирующие позиции. Во втором случае для сохранения всех ограничений на возможность одновременного выполнения процедур процесса, которые задаются этими позициями, приходится осуществлять преобразование сети Петри, выбирая лишь допустимую последовательность реализации переходов. С этой целью можно использовать граф достижимых маркировок сети Петри и совместить преобразование процесса с устранением тупиковых состояний.

На рис. 5.33 приведен граф статических состояний сети Петри рис. 5.15.

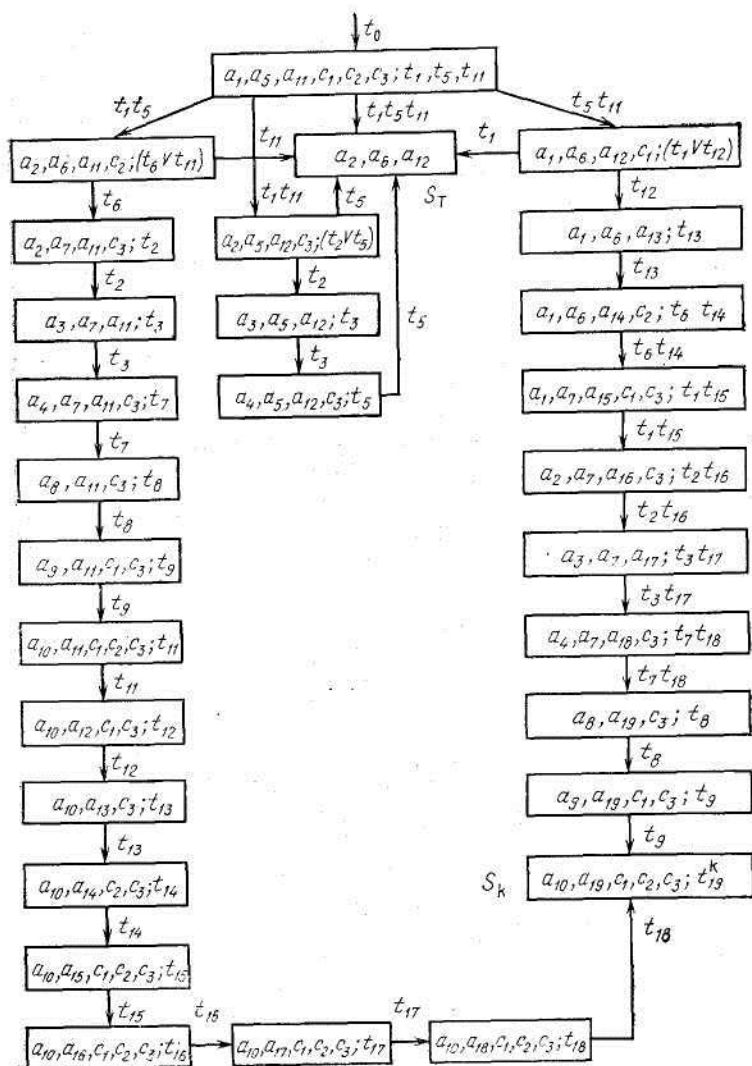


Рис. 5.33.

Как видно, этот граф содержит тупиковое состояние S_T , в которое процесс попадает при одновременном развитии трех параллельных подпроцессов. Для того чтобы исключить возможность возникновения тупикового состояния и получить правильный процесс, необходимо

Как видно, в граф сети Петри введены дополнительные переходы распараллеливания и соединения и новые позиции, которые запрещают одновременное выполнение процедур, использующих одни и те же ФР.

От полученной таким образом сети Петри можно перейти к описанию этого процесса на языке ПЛСА. Для этого с каждым переходом t_j^i , т. е. с процедурой A_i процесса, сопоставим оператор A_i ПЛСА, с каждым переходом распараллеливания — оператор R , а с переходом соединения — оператор S . Если в сети Петри не были выделены переходы, соответствующие началу и окончанию процесса, операторы A_0 и A_k могут быть введены в ПЛСА дополнительно.

Для сети Петри рис. 5.34 получим следующую ПЛСА:

$$\mathfrak{A} = A_0 R_1 \omega \downarrow_2^1 \downarrow^1 A_2 \omega \uparrow_2^2 \downarrow^1 A_5 A_9 A_{10} R_2 \omega \uparrow_2^3 \downarrow^3 \omega \uparrow_2^4 \downarrow^4 A_{11} R_3 \omega \uparrow_2^4 \downarrow^4 A_5 \omega \uparrow_2^5 \downarrow^4 A_{12} A_{13} A_{14} A_{15} \omega \uparrow_2^6 \downarrow^2 S_1 A_3 \omega \uparrow_2^5 \downarrow^5 S_2 A_1 A_6 A_4 A_7 \omega \uparrow_2^6 \downarrow^6 S_3 A_k.$$

Каждый параллельный участок ПЛСА можно оформить в виде отдельного подалгоритма, введя операторы B_0 и B_k .

Рассмотрим особенности перехода от сети Петри разветвленного процесса к ПЛСА. Если этот процесс является последовательным, то он описывается ЛСА, содержащей только операторы и ЛУ. Переходу t_j^i сопоставляется оператор ЛСА A_i . Если выходной позицией перехода t_j^i является позиция альтернативного разветвления, то в ЛСА после A_i вводятся ЛУ, наборы значений которых разрешают конфликт переходов — последователей этой позиции. При этом не различаются внутренние и внешние ЛУ, а следующие за ними стрелки разветвления проставляются в соответствии с заданной сетью Петри последовательностью выполнения процедур. Информация о влиянии процедур на значения ЛУ задается в виде распределения сдвигов операторов ЛСА. В результате исключаются все позиции d_s и $\bar{d}_s$, причем последние заменяются логическим условием p_s .

Сеть Петри рис. 5.20 может быть описана следующей ЛСА:

$$\mathfrak{A} = A_0 \downarrow^2 A_1 p_1 \uparrow^1 A_2 \omega \uparrow^2 \downarrow^1 A_3 p_2 \uparrow^3 A_4 \omega \uparrow^2 \downarrow^3 A_5 A_k$$

и распределением сдвигов: $A_2 - \{p_1\}$, $A_1 - \{p_2\}$ (остальные операторы имеют пустое распределение сдвигов). Ясно, что при этом вновь происходит потеря информации, содержащейся в исходной сети Петри. Из рис. 5.20 видно, какие значения принимают логические условия p_1 и p_2 после выполнения процедур A_2 и A_1 . Распределение сдвигов задает лишь возможность изменения операторами ЛСА значений ЛУ.

Аналогичным образом можно перейти к ПЛСА от сети Петри, задающей разветвленный параллельный процесс. Очевидно, получаемая ПЛСА эквивалентна первоначальной сети Петри в том смысле, что она описывает некоторое подмножество всех возможных последовательностей выполнения процедур КП и может рассматриваться как одна из его допустимых реализаций.

Следует подчеркнуть, что прежде чем переходить к ПЛСА от сети Петри, необходимо перевести задаваемый ею процесс в правильный, так как существующие методы анализа ПЛСА не позволяют обнаруживать и устранять тупиковые состояния. Например, по сети Петри рис. 5.19 можно составить следующую ПЛСА:

$$\mathfrak{A} = A_0 R_1 \omega \uparrow_2^1 \downarrow^1 p_1 \uparrow^2 A_1 A_5 \omega \uparrow^3 \downarrow^2 A_2 \omega \uparrow^4 \downarrow^1 A_3 \omega \uparrow^4 \downarrow^2 S_1 \\ A_4 \omega \uparrow^3 \downarrow^2 S_2 A_k.$$

Поскольку ФР C_2 используется процедурами разных альтернативных ветвей, позиция c_2 не накладывает дополнительных ограничений на последовательность выполнения процедур. По этой ПЛСА уже трудно выявить наличие тупикового состояния в данном процессе, тогда как это сразу выявляется по графу достижимости (см. рис. 5.22).

В то же время проверку полноты и непротиворечивости удобно делать по ПЛСА, составляя МСА для отдельных подалгоритмов. Для объединения частных алгоритмов, их преобразования и упрощения с учетом распределения сдвигов также используются ЛСА и МСА. Таким образом, *при проектировании УС целесообразно применять различные языки, имея средства перехода от одного языка к другому.*

Необходимо иметь в виду, что граф статических состояний сети Петри дает неполную информацию о возможных последовательностях выполнения процедур процесса, поэтому рассмотрение полного графа с учетом длительности реализации переходов может привести к получению другой ПЛСА. Заметим, что для процессов, не зависящих от скорости, результаты анализа по графу статических состояний и по полному графу совпадают, однако в общем случае необходимо использовать полный граф, несмотря на его сложность.

Проиллюстрируем это на примере сети Петри рис. 5.35, описывающей параллельный разветвленный процесс с начальным значением внутреннего логического условия p_2 равным нулю.

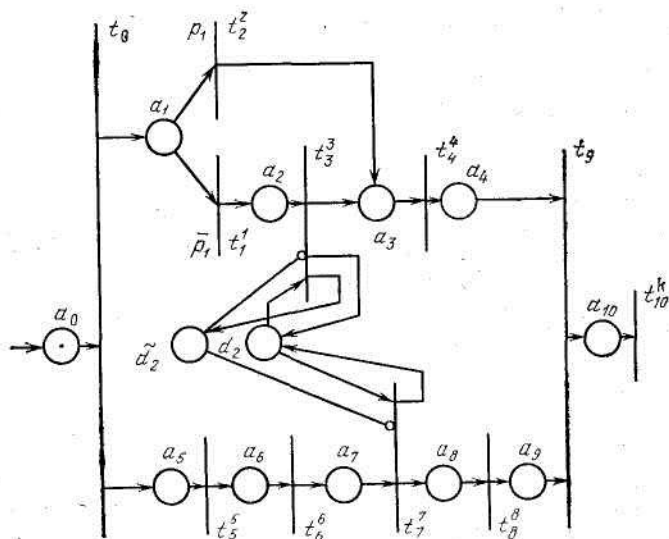


Рис. 5.35.

Все процедуры процесса используют собственные ФР, которые поэтому в описание процесса не включены. На рис. 5.36 приведена часть полного графа достижимых маркировок, соответствующая развитию процесса при ложном значении внешнего логического условия p_1 .

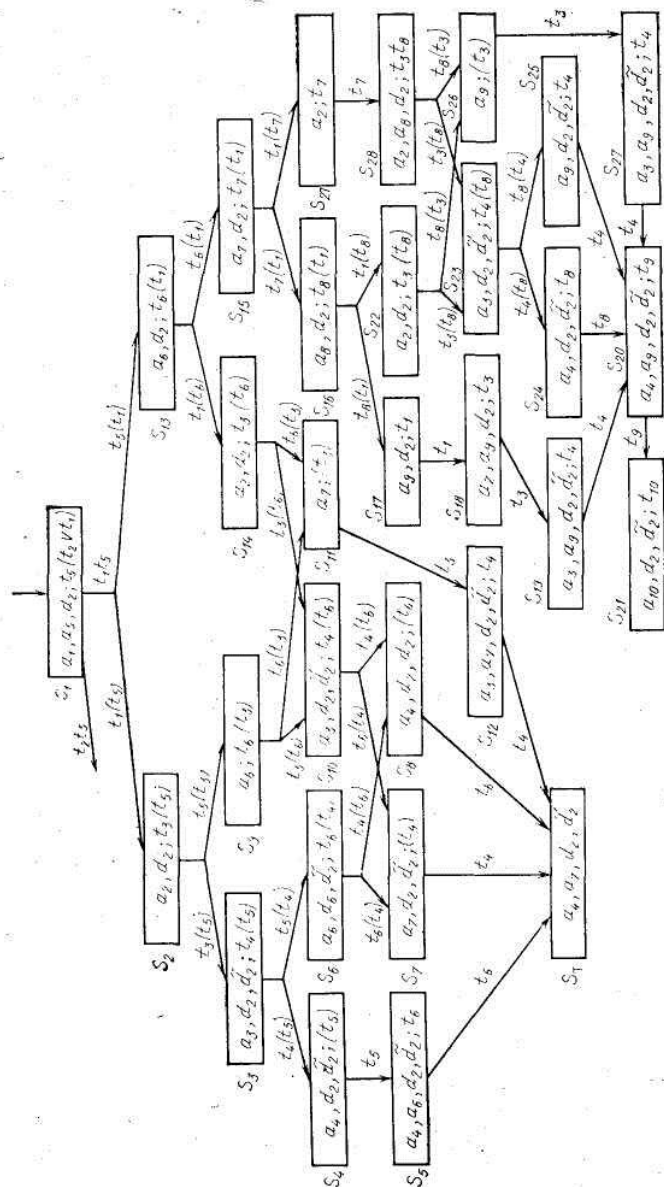


Рис. 5.36.

Как видно, в зависимости от длительности реализации отдельных переходов процесс либо успешно завершается, переходя в заключительное состояние S_{21} , либо попадает в тупиковое состояние S_T . Для предотвращения развития процесса, приводящего в тупик, нужно обеспечить переход его в состояние S_{15} , после которого все пути, имеющие одинаковую длину, ведут в конечное состояние. Преобразованная в соответствии с этим требованием сеть Петри приведена на рис. 5.37, по которой получена следующая ПЛСА:

$$\mathcal{A} = A_0 R_1 \omega \downarrow_2^1 \bar{\downarrow}_1^1 p_1 \uparrow^2 \omega \downarrow_2^3 \bar{\downarrow}_1^1 A_5 A_6 A_7 R_2 \omega \downarrow_2^4 \bar{\downarrow}_2^4 \omega \downarrow_2^3 \bar{\downarrow}_2^3 S_1$$

$$A_3 \downarrow^2 A_4 \omega \downarrow_2^5 \bar{\downarrow}_2^5 A_8 \omega \downarrow_2^5 \bar{\downarrow}_2^5 S_2 A_K.$$

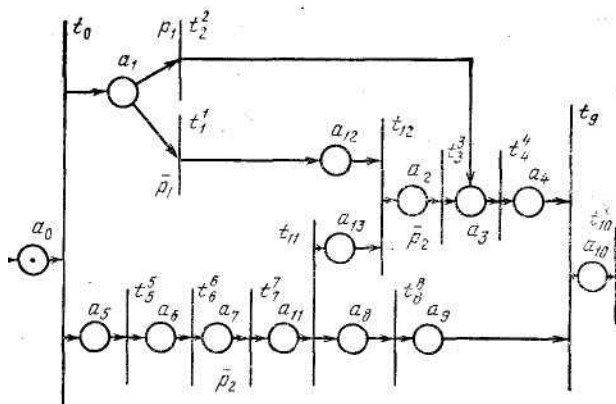


Рис. 5.37.

Если использовать граф статических состояний сети Петри рис. 5.35, то можно сделать неверный вывод о том, что при $p_1=0$ процесс всегда попадает в тупиковое состояние. В действительности, как видно из рис. 5.36, это происходит только при определенном соотношении длительностей реализации переходов t_5 , t_6 и t_1 . Когда для описания процесса используется временная сеть Петри и с каждым переходом t_j сопоставляется длительность его реализации Δ_j , можно проверить, выполняются ли необходимые временные соотношения. Так, в нашем примере, если $\Delta_1 > \Delta_5 + \Delta_6$, то тупиковое состояние недостижимо и никаких преобразований исходной сети Петри не требуется. Поскольку в нашем случае длительность реализации перехода не известна, осуществляется преобразование сети Петри с целью принудительного выполнения условия, устраняющего тупиковое состояние.

Таким образом, для *перехода от сети Петри к ПЛСА* необходимо *построить полный граф достижимых маркировок, выбрать в нем кратчайший путь, приводящий процесс в конечное состояние, и в соответствии с ним осуществить преобразование сети Петри, исключив все позиции конфликтов*. По полученной сети Петри *составляют ПЛСА*, а по исходной — *распределение сдвигов*. После этого проверяют ПЛСА на полноту и непротиворечивость с целью получения ПЛСА, задающей правильный облачный управляющий алгоритм, которая затем используется для построения управляющего средства.

Конечно, не исключается возможность синтеза УС непосредственно по сети Петри. Это наиболее целесообразно при рассмотрении множества асинхронных взаимодействующих процессов, когда возникающие тупики устраняются, например, с применением механизма семафоров и предполагается программная реализация управляющего средства.

5.4. Реализация облачного управляющего алгоритма

5.4.1. Принципы реализации параллельного облачного управляющего алгоритма

При реализации алгоритма функционирования УС в зависимости от его сложности, используемой элементной базы, особенностей ОУ и т. п. могут быть использованы различные принципы реализации, из которых рассмотрим четыре основных. При этом для наглядности принципы реализации параллельного облачного управляющего алгоритма рассмотрим на примере алгоритма, представленного в виде ПГСА (рис. 5.38).

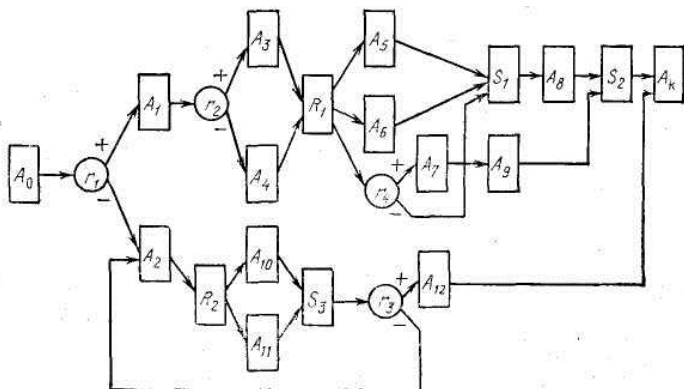


Рис. 5.38.

Заметим, что ПГСА вместо отдельных команд управления (операторов A_i и логических условий r_j), представляющих собой элементарные частные управляющие алгоритмы, могут быть представлены и достаточно сложные частные управляющие алгоритмы A_i и A_j (управляющие подалгоритмы), определяющие функционирование УС на отдельных этапах или режимах его работы. При этом аналогами оператора A_i и логического условия r_j будут подалгоритм A_i с одним выходом (выходным портом) и подалгоритм A_j с двумя выходами (выходными портами). Число входов (входных портов) как в подалгоритме A_i с одним выходным портом, так и в подалгоритме A_j с двумя выходными портами может быть любым. В ПГСА, описывающей порядок выполнения подалгоритмов (т. е. структуру ПГСА), по-прежнему будем использовать операторы A_0 , A_k , R_l и S_m .

Первый принцип. Наиболее наглядным и простым для понимания принципом реализации управляющего алгоритма является такой, при котором каждый оператор A_i и логическое условие, r_j (частные управляющие алгоритмы A_i и A_j) реализуются отдельными программами УС. При этом будет получена совокупность операций, структура которой будет повторять структуру ПГСА. Таким образом, в нашем случае в совокупности операций, не считая входные и выходные операции, соответствующие операторам A_0 и A_k , будет использовано 19 операций, включая операции, предназначенные для распараллеливания и соединения параллельных подалгоритмов. Сложность каждой операции такой совокупности зависит от функций, которые сопоставляются с соответствующим оператором и ЛУ. В случае применения в ПГСА частных управляющих алгоритмов каждая

из операций совокупности может оказаться достаточно сложной, из-за чего, в свою очередь, потребуется ее декомпозиция на совокупность более простых операций, т. е. образуется двухуровневая иерархия совокупностей операций.

Таким образом, одноуровневая или двухуровневая (может быть и многоуровневая) совокупностей операций представляет собой *распределенный принцип построения УС*.

Второй принцип. Альтернативным распределенному принципу реализации управляющего алгоритма может служить *централизованный принцип*, когда весь параллельный алгоритм реализуется одним УС, в каждом такте работы которого одновременно выполняется один или несколько операторов и (или) логических условий.

В том случае, когда в ПГСА вместо операторов и ЛУ указаны частные управляющие алгоритмы, вместо такта говорят о цикле (периоде) работы УС, в течение которого для реализации соответствующего частного управляющего алгоритма может потребоваться несколько, может быть и довольно большое число тактов работы УС.

При данном принципе реализация последовательного управляющего алгоритма не вызывает никаких трудностей. Однако при реализации параллельного управляющего процесса алгоритма в одном УС возникает ряд трудностей, связанных с выполнением параллельных ветвей алгоритма (параллельных подалгоритмов) в одном такте (цикле, периоде) работы УС.

Рассмотрим четыре возможных варианта сочетаний параллельно выполняемых членов (частных управляющих алгоритмов) ПГСА:

1. В параллельных ветвях ПГСА имеются только операторы (параллельные подалгоритмы с одним выходным портом). Для простоты, здесь далее будем рассматривать ПГСА с двумя параллельными ветвями.

Пусть в одной ветви имеется n операторов $A_{i1}, \dots, A_{in}$, а в другой — m операторов $A_{j1}, \dots, A_{jm}$, $m \neq n$, и для определенности $m < n$ (рис. 5.39).

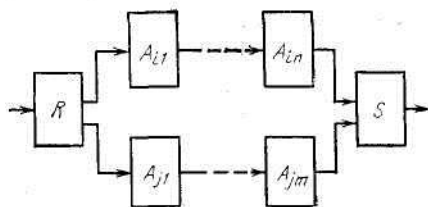


Рис. 5.39.

Тогда могут быть образованы m объединенных операторов $A_{ij1}, \dots, A_{ijm}$ и $n-m$ операторов $A_{im+1}, \dots, A_{in}$, каждый из которых может реализоваться в отдельном такте работы УС.

2. В одной ветви ПГСА имеются только операторы, а во второй — хотя бы одно ЛУ. При этом ЛУ может находиться в начале, середине или конце ветви.

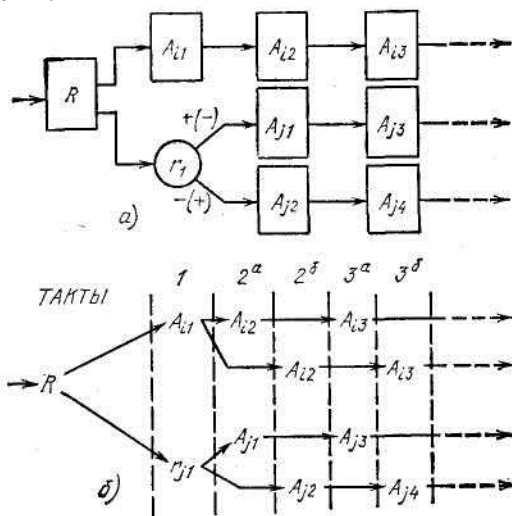


Рис. 5.40.

Рассмотрим эти три случая:

а) логическое условие находится в начале ветви (рис. 5.40, а). Очевидно, в одном и том же такте (например, такте 1) можно объединить оператор A_{i1} с логическими условиями r_{j1} (рис. 5.40, б). В связи с тем что после ЛУ r_{j1} происходит разветвление ПГСА, операторы A_{j1} и A_{j2} , находящиеся в альтернативных ветвях, не могут быть выполнены одновременно в одном и том же такте. Пусть оператор A_{j1} выполняется в такте 2^a при $r_{j1} = 1$ (стрелка со знаком + на рис. 5.40, а), а оператор A_{j2} — в такте 2^b , следующим за тактом 1 при $r_{j1} = 0$ (стрелка со знаком минус). Вместе с тем оператор A_{i2} выполняется независимо от значения ЛУ r_{j1} находящегося в параллельной ветви, т. е. оператор A_{i2} должен выполняться как при $r_{j1} = 1$, так и при $r_{j1} = 0$. Поэтому оператор A_{i2} должен быть совмещен и с оператором A_{j1} (такт 2^a) и с оператором A_{j2} (такт 2^b). Аналогично последующие операторы $A_{i3}, \dots, A_{in}$ также должны совмещаться с операторами как одной, так и другой альтернативной ветвью второй параллельной ветви консультационного алгоритма;

б) логическое условие находится в середине ветви. До этого ЛУ совмещение операторов различных параллельных ветвей происходит так же, как это было описано в варианте 1, а начиная с этого логического условия — как в случае для варианта 2^а;

в) логическое условие находится в конце ветви.

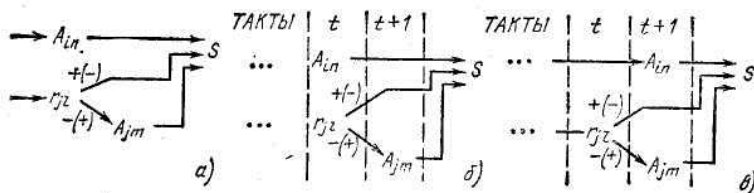


Рис. 5.41.

В этом случае (рис. 5.41, а) после логического условия r_{jl} в одной из альтернативных ветвей после $r_{jl}=0$ (знак —) или $r_{jl}=1$ (знак +) и только в одной из них перед оператором соединения S должен быть хотя бы один оператор (в нашем случае оператор A_{jm}). Тогда могут быть два варианта совмещения: в такте t совмещаются оператор A_{in} и логическое условие r_{jl} (рис. 5.41, б), а в такте $t+1$ выполняется только оператор A_{jm} , или в такте t выполняется только логическое условие r_{jl} , а в такте $t+1$ совмещаются операторы A_{in} и A_{jm} .

Заметим, что операторы S и R , как и операторы A_0 и A_k , совмещаются между собой и с другими операторами, как правило, не могут.

3. В каждой из параллельных ветвей имеются логические условия. Если они находятся в разных местах параллельных ветвей ПГСА и совмещение этих ЛУ невозможно, то условия совмещения операторов и логических условий параллельных ветвей в этом случае будут такими же, как и во втором варианте.

Рассмотрим особенности совмещения ЛУ, входящих в разные параллельные ветви. Как и в предыдущем варианте сочетания параллельных ветвей (вариант 2), рассмотрим три случая (ЛУ находятся в начале, середине и конце параллельных ветвей ПГСА):

а) логические условия находятся в начале параллельных ветвей ПГСА (рис. 5.42, а).

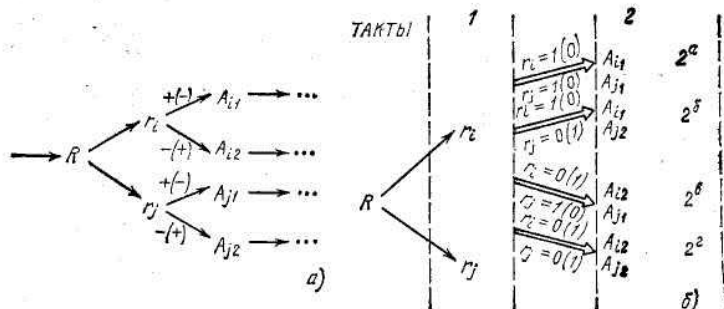


Рис. 5.42.

Если их совместить в одном такте 1 (рис. 5.42, б), то при двух ЛУ возможно четыре сочетания значений ЛУ, а следовательно, четыре возможных разновидности такта 2; такт 2^a , когда должны одновременно выполняться операторы A_{i1} и A_{j1} ; такт 2^b — операторы A_{i1} и A_{j2} ; такт 2^c — операторы A_{i2} и A_{j1} , такт 2^d — операторы A_{i2} и A_{j2} . После такта 2^5 ($\xi = a, б, в, г$) в последующих тактах совмещаются те операторы, которые являются последователями операторов, совмещенных в такте 2^5 , ($\xi = a, б, в, г$);

б) в том случае, когда логические условия находятся в середине параллельных ветвей ПГСА, то, как и в случае для варианта 2б, совмещение операторов различных параллельных ветвей происходит аналогично тому, как это было описано в варианте 1а, начиная с этих ЛУ — как в случае для варианта 3а;

в) логические условия находятся в конце ветвей ПГСА. Как и в случае для варианта 2в в одной из альтернативных ветвей в каждой параллельной ветви за r_i, r_j должен следовать хотя бы один оператор (рис. 5.43, а).

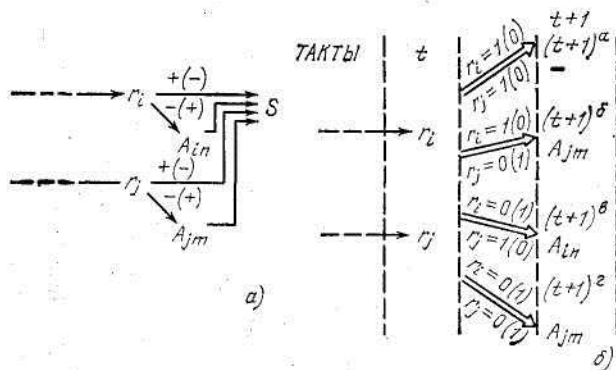


Рис. 5.43.

Тогда очевидно, что при двух ЛУ могут быть четыре сочетания их значений, а поэтому четыре разновидности такта $t+1$, как это указано на рис. 5.43, б.

4. Хотя бы в одной из параллельных ветвей ПГСА имеется цикл. Например, в одной из двух параллельных ветвей имеется цикл, как это показано на рис. 5.44, а.

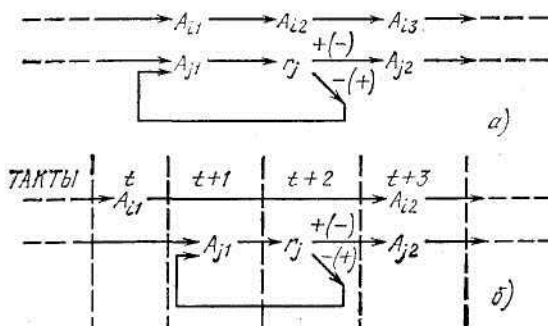


Рис. 5.44.

В связи с тем, что заранее не известно, сколько тактов работы УС займет цикл, операторы и ЛУ, входящие в цикл, не могут быть совмещены с операторами и ЛУ параллельной ветви. Поэтому для членов (операторов, ЛУ) цикла должны быть отведены отдельные такты (рис. 5.44, б), число повторений которых, например, в нашем случае зависит от того, когда ЛУ сменит свое значение с 0 (или 1) на 1 (или 0).

Второй принцип реализации управляющего алгоритма рассмотрим на примере.

Пример 1. Пусть управляющий алгоритм задан в виде ПГСА, изображенной на рис. 5.38. Требуется распределить операторы и ЛУ по тактам работы УС при втором принципе реализации управляющего алгоритма. При этом для последовательно выполняемых членов ПГСА, т. е. операторов и ЛУ, входящих в одну и ту же ветвь ПГСА, будем отводить отдельный такт работы УС.

При этих условиях в первых двух тактах последовательно выполняются оператор A_0 и логическое условие r_1 (рис. 5.45, а).

После ЛУ r_1 происходит разветвление алгоритма: выполняется ветвь ПГСА, начиная с оператора A_1 (такт 3^a), или ветвь ПГСА, начиная с оператора A_2 (такт. 3^b , рис. 5.45,б). Аналогично алгоритм разветвляется на две альтернативные ветви после ЛУ r_2 и r_3 . При этом если $r_2=1$, то будет выполнен оператор A_3 (такт 5^a , рис. 5.45,а), а при $r_2=0$ — оператор A_4 (такт 5^b , рис. 5.45,в. При $r_3=1$ реализуется оператор A_1 (такт 17, рис. 5.45,б), а при $r_3=0$ — оператор A_2 (такт 3^b , рис. 5.45,г).

Таким образом, из-за разветвления алгоритма на альтернативные ветви при проверке логических условий r_1 , r_2 и r_3 может быть получено пять вариантов последовательности выполнения операторов и ЛУ управляющего алгоритма. При этом начиная с такта 6 процессы выполнения операторов и ЛУ при $r_1 = 1$, $r_2=1$ (рис. 5.45,а) и при $r_1=1$, $r_2=0$ (рис. 5.45,б) совпадают. Поэтому на рис. 5.45,в после такта 6 при $r_4=1$ такты 7, 8^a , 9, 10, 11, 12 (рис. 5.45,а), а также при $r_4 = 0$ такты 7, 8^b , 18, 19, 12 (рис. 5.45,д) не указаны.

В связи с тем что при $r_4=0$ соединение параллельных подалгоритмов осуществляется оператором S_1 , а не оператором S_2 , при $r_4=0$ оператору S_1 отведен новый такт 18, где он реализует несколько отличные функции от функций такта 9 — соединение не двух, а трех параллельных ветвей управляющего алгоритма.

В том случае, когда вместо операторов в ПГСА используются подалгоритмы, их объединение осуществляется на уровне операторов и логических условий, входящих в эти подалгоритмы по приведенным выше правилам.

Третий принцип. Промежуточным принципом реализации куправляющего алгоритма между рассмотренными выше может служить псевдопараллельная реализация в виде *режима разделения времени*.

Режим (система) разделения времени возник в начале шестидесятых годов XX в. для организации на одной ЭВМ обслуживания многих пользователей. Система разделения времени дала возможность программисту постоянно контролировать ход решения задачи на ЭВМ и вмешиваться в процесс ее решения без заметного по сравнению с пакетной обработкой снижения использования ЭВМ. В связи с этим режим разделения времени позволил эффективно применять ЭВМ в качестве управляющих машин, работающих в масштабе реального времени, когда создаваемыми ЭВМ задержками при управлении можно пренебречь.

Режим разделения времени может использоваться не только для обслуживания многих пользователей на однопроцессорной ЭВМ или управлять многими блоками объекта управления в масштабе реального времени, но и реализовать параллельный управляющий алгоритм.

При реализации параллельного управляющего алгоритма в режиме разделения времени процесс выполнения операторов и ЛУ параллельных ветвей ПГСА разбивается на *периоды*. При этом в каждом периоде для каждой параллельной ветви отводится один *квант времени*, в течение которого УС реализует команды, соответствующие операторам или ЛУ, только одной ветви ПГСА. Так, вначале в первом кванте времени выполняются команды первой параллельной ветви. Затем во втором кванте времени — команды первой ветви прерываются и начинаются команды второй параллельной ветви и т. д. до тех пор, пока не закончится данный период. Во втором периоде данный процесс повторится, т. е. в первом кванте времени возобновится выполнение команд первой параллельной ветви и т. д.

В простейшем случае, когда в параллельных ветвях имеется по одной команде (оператору или ЛУ), то период соответствует такту работы УС, а в качестве кванта времени может рассматриваться *микротакт*. При этом режим разделения времени вырождается в обычный процесс последовательного (по-тактного) выполнения команд параллельных ветвей управляющего алгоритма.

Принцип псевдопараллельной реализации параллельного управляющего алгоритма в режиме разделения времени для такого простейшего случая рассмотрим на примере.

Пример 2. Пусть задан параллельный управляющий алгоритм, представленный на рис. 5.38 в виде ПГСА. Временная диаграмма работы программной УС, реализующей данный управляющий алгоритм, представлена на рис. 5.46.

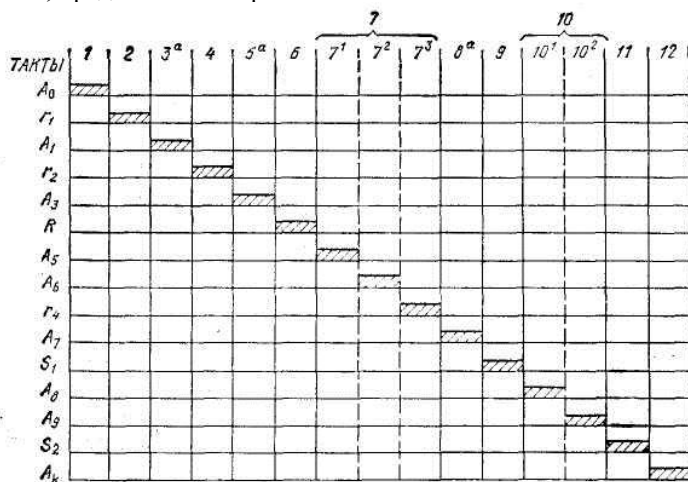


Рис. 5.46.

При этом, как и ранее, предполагается, что в одном такте работы САК выполняется один оператор или одно ЛУ одной ветви ПГСА, а различные последовательности операторов при различных наборах значений логических условий r_1, r_2, r_3 и r_4 представим, как и в примере 1, различными временными диаграммами. Для примера на рис. 5.46 изображена временная диаграмма выполнения операторов и ЛУ для одной ветви ПГСА при $r_1=1, r_2=1$ и $r_4=1$ соответствующей ветви, изображенной на рис. 5.45,а.

Из рис. 5.46 видно, что такт 7, в котором параллельно должны быть выполнены операторы A_5, A_6 и логические условия r_4 , разделен на три кванта времени (микротакта), каждый из которых отводится для одной из трех параллельных ветвей. Аналогично такт 10 разделен на два микротакта — в одном (микротакте 10^1) для оператора A_8 , а в другом (микротакте 10^2) — для оператора A_9 , принадлежащего второй параллельной ветви.

В том случае, когда в такте нет операторов или ЛУ, принадлежащих параллельным ветвям, его деление на микротакты не происходит и вся длительность такта отводится для соответствующего члена ПГСА.

Наиболее рельефно эффективность системы разделения времени при реализации параллельного управляющего алгоритма проявляется в том случае, когда в параллельных ветвях имеется достаточно большое число операторов и ЛУ.

Пример 3. Пусть в режиме разделения времени необходимо организовать выполнение двух подалгоритмов A_1 и A_2 управляющего алгоритма, включающих соответственно 5 и 11 членов (команд) ПГСА $K_i^1, i=1, \dots, 5; K_j^2, j=1, \dots, 11$. При этом для подалгоритма A_1 отводятся нечетные кванты времени в периоде, а для подалгоритма A_2 — четные.

Процесс выполнения команд этих параллельных подалгоритмов в режиме разделения времени виден из рис. 5.47.

Частные
консультационные
алгоритмы

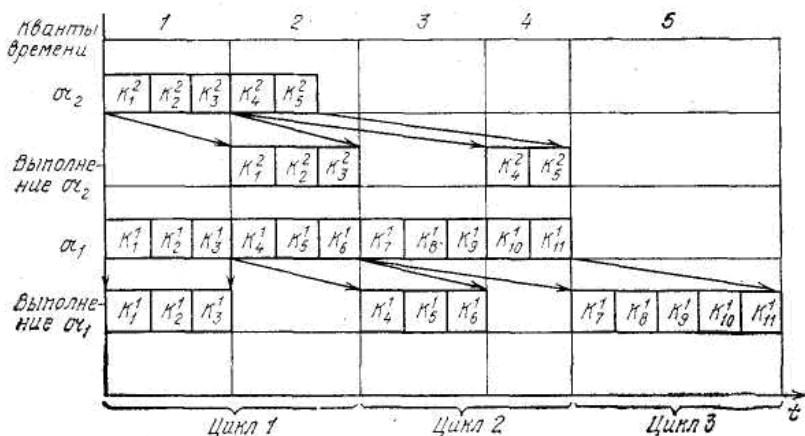


Рис. 5.47.

При этом легко заметить, что когда и в A_1 и в A_2 имеются невыполненные команды, в периоде по очереди отводятся кванты времени и для A_1 (кванты 1, 3), и для A_2 (кванты 2, 4). Когда же заканчиваются команды подалгоритма A_2 , деление периода на кванты уже не осуществляется — САК начинает последовательно выполнять команды только подалгоритма A_1 . При этом квант времени совпадает с периодом, а сам период не ограничен во времени. Фактически все время работы САК отводится для подалгоритма A_1 до тех пор, пока он не прекратится или не встретится новый подалгоритм, который должен выполняться с ним параллельно.

Четвертый принцип. В заключение следует отметить, что система разделения времени в значительной степени опирается на использование режима многопрограммного управления, которое реализуется в децентрализованном УС с активными функциональными блоками (ФБ), среди которых могут встречаться медленно действующие ФБ (МФБ). При применении режима многопрограммного управления, которое может рассматриваться в качестве четвертого принципа реализации параллельного управляющего алгоритма, будем говорить, что УС является многопрограммным.

В связи с тем что многопрограммное управление получает широкое распространение в УС, рассмотрим его более подробно в следующем разделе.

5.4.2. Многопрограммное управление

Программный принцип построения УС, как уже отмечалось, наиболее эффективен в том случае, когда в УС может быть реализовано несколько различных управляющих алгоритмов. Обычно выбирается набор однотипных управляющих задач, для решения которых требуются одинаковые ФБ, и для каждой задачи составляется своя программа (алгоритм). Все эти программы хранятся в центральном блоке управления (ЦБУ) УС (см. рис. 5.48) и в зависимости от поступившего на ЦБУ сигнала R (это в частном случае может быть код операции) в САК выполняется определенный алгоритм.

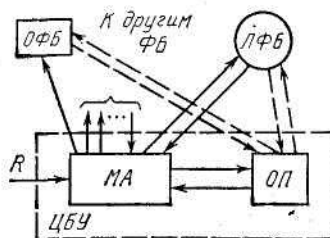


Рис. 5.48.

УС может перейти к другому алгоритму лишь после того, как будет закончен предыдущий. При этом ЦБУ непрерывно следит за работой функциональных блоков, принимающих участие в решении данной задачи. Подав включающий сигнал на ФБ, ЦБУ независимо от длительности работы этого блока находится в состоянии ожидания ответного сигнала от этого блока (при асинхронном режиме) или от тактового генератора (при синхронном режиме). Таким образом, хотя УС, как правило, может решать несколько различных задач или управлять несколькими процессами, одновременно УС решает только одну задачу, т. е. работает только по одному из алгоритмов, хранящихся в ЦБУ. Такой принцип можно было бы назвать однопрограммным.

Однако характерной особенностью УС является наличие в их составе устройств, имеющих разное быстродействие. Часть этих устройств (например, ЦБУ) может обладать весьма высоким быстродействием. Быстродействие же другой части устройств в значительной степени ограничено связанными с их работой процессами механических перемещений в ОУ. Так, в ЭВМ медленно действующими являются операции ввода и вывода информации.

Во всех этих случаях при сохранении однопрограммной работы УС повышение быстродействия отдельных ее узлов (например, ЦБУ) не

приводит к соответствующему увеличению производительности УС из-за простаивания этих быстродействующих узлов в ожидании срабатывания более медленных.

Одним из средств повышения производительности УС является совмещение во времени отдельных шагов управляющего алгоритма. При этом из ЦБУ подается одновременно несколько сигналов на включение параллельно работающих различных ФБ УС, что приводит к сокращению общего времени выполнения алгоритма. Но и в этом случае ЦБУ управляет одновременно выполнением только одного технологического процесса и каждый ФБ УС все время находится под контролем ЦБУ. При этом даже при асинхронном режиме работы УС, обеспечивающем наибольшее быстродействие, ЦБУ непроизводительно простаивает в ожидании ответа об окончании работы медленно действующего функционального блока (МФБ).

Между тем для работы активных ФБ не требуется постоянное воздействие со стороны ЦБУ — достаточно подать лишь кратковременный сигнал на начало работы ФБ. Производительность ЦБУ можно значительно повысить, если на время работы МФБ отключить ЦБУ от управления начатым процессом и использовать его для решения другой задачи. При этом такие МФБ должны быть в значительной степени автономными, чтобы они могли выполнить достаточно сложную обработку информации, не получая от ЦБУ дополнительных данных и сигналов. В ряде случаев такие МФБ могут быть настолько сложными, что сами оказываются УС. Таким образом, возникает некоторая иерархия УС. Однако для УС верхнего уровня все УС нижнего уровня являются просто ФБ, выполняющими определенные акты алгоритма и находящимися в подчинении его ЦБУ.

Подав сигнал на включение МФБ, ЦБУ переходит к решению другой задачи, вырабатывая последовательность управляющих сигналов в соответствии с алгоритмом ее решения, доходит до сигнала, включающего МФБ в этой программе, переходит к решению третьей задачи и т. д. (рис. 5.49).

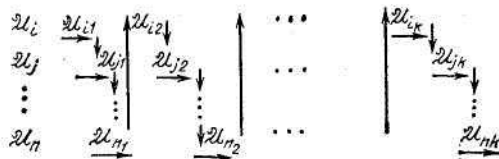


Рис. 5.49.

Таким образом создается возможность выполнять параллельно несколько ветвей одной программы, или выполнять параллельно несколько различных программ, или решать одну и ту же задачу, но

для различных исходных данных одновременно. Такой принцип работы УС находит применение во многих УС и получил название многопрограммного управления.

Очевидно, для того чтобы вернуться к решению прерванной задачи, ЦБУ должен иметь информацию об этапе, на котором она была прервана, об окончании работы того МФБ, который был включен, и др. Вся эта информация хранится в ячейках оперативной памяти.

Выбор емкости памяти и способ ее распределения могут быть выполнены по-разному и представляют самостоятельную задачу, близкую к задачам, решаемым в теории массового обслуживания. Можно считать, что имеется какое-то число N обслуживаемых объектов (это могут быть абоненты, исходные данные для решения задачи и т. д.), от которых в УС поступают заявки на определенный вид обслуживания. Начав обслуживание по определенному алгоритму и дойдя до выполнения первого медленно действующего оператора, ЦБУ переходит к обслуживанию другого объекта и т. д. При этом за каждым из объектов можно закрепить свою ячейку памяти, в отдельных разрядах которой будут отмечаться номер алгоритма, по которому в данный момент обслуживается этот объект, этап алгоритма, выполняемый в данный момент, а также номера промежуточных устройств, занятых на этом этапе обслуживания. Однако если число обслуживаемых объектов велико, то при таком способе распределения памяти УС объем ее может оказаться очень большим. Не от всех объектов могут поступать одновременно заявки на их обслуживание и не все поступившие заявки УС может обслуживать одновременно. Поэтому число ячеек памяти целесообразно выбирать не по общему числу объектов, а по числу объектов, которое УС может обслуживать одновременно. Тогда в каждой ячейке памяти нужно будет записывать еще и номер объекта, за которым она в данный момент закреплена.

Возможен и другой подход. Поскольку УС может обслуживать один и тот же объект в соответствии с различными алгоритмами, то можно для каждого из алгоритмов, хранящихся в ЦБУ, отвести свою ячейку.

Вообще говоря, число одновременно выполняемых операций не может превышать число функциональных блоков в УС, а число алгоритмов, обслуживание которых начал ЦБУ, может быть значительно большим и ограничивается только емкостью памяти. Однако с целью экономии емкости оперативной памяти (ОП) целесообразно число ячеек выбирать не по общему числу алгоритмов, а по числу алгоритмов, в соответствии с которыми одновременно может вестись обслуживание поступающих заявок. Тогда в отдельные разряды ячеек памяти будут

записываться номер заявки, номер алгоритма, по которому она обслуживается, выполняемый этап алгоритма и необходимые номера промежуточных устройств, занятых в этом процессе. Так как заявки могут одновременно обслуживаться по одному и тому же алгоритму, различные ячейки памяти могут соответствовать одному алгоритму. В зависимости от решаемых задач и требований, предъявляемых к УС, может быть выбран тот или иной способ распределения памяти.

Отказавшись от жесткого закрепления ячеек памяти за номерами поступающих заявок или за номерами алгоритмов, приходим к общему случаю, когда каждая свободная ячейка памяти может быть занята любой заявкой и отведена для любого из алгоритмов. При этом хотя и сократится число ячеек по сравнению с описанными крайними случаями, разрядность каждой ячейки увеличится, так как нужно будет запоминать номер заявки, которой она занята, и номер алгоритма, по которому эта заявка в данный момент обслуживается. Число ячеек памяти, которое необходимо при этом выбрать, будет определяться числом заявок и алгоритмов их обслуживания, а также свойствами УС.

Наряду с выбором емкости оперативной памяти ЦБУ возникает задача выбора способа организации его работы при многопрограммном управлении. Начав обслуживание одной заявки по требуемому алгоритму и дойдя до первой медленно действующей операции этого алгоритма, ЦБУ включает соответствующий МФБ и переходит к обслуживанию другой заявки. Возникает естественный вопрос—через какое время ЦБУ должен продолжить обслуживание первой заявки. Это время будет зависеть от того, по какому принципу (дисциплине) организована работа ЦБУ. Рассмотрим два из возможных принципов. При первой дисциплине обслуживания ЦБУ последовательно просматривает состояние всех ячеек оперативной памяти и выполняет для каждой заявки все операции алгоритма (по которому она обслуживается), не требующие включения МФБ. После включения МФБ ЦБУ переходит к считыванию информации со следующей ячейки ОП и т. д. Таким образом, ЦБУ может продолжить обслуживание начатой заявки лишь после того, как будут просмотрены все остальные ячейки памяти и выполнены определенные операции по обслуживанию заявок, для которых они отведены. Если число ячеек ОП будет очень большим, то ЦБУ сможет начать обслуживание большого числа заявок, однако время выполнения каждой заявки увеличится по сравнению с однопрограммным методом управления, так как МФБ успеют закончить свою работу намного раньше, чем ЦБУ обратится вновь к данной ячейке памяти. Если число ячеек слишком мало, то ЦБУ возьмет на обслуживание небольшое число заявок и будет столь часто обращаться к каждой ячейке памяти, что МФБ не будут успевать

заканчивать работу за время одного цикла опроса ЦБУ ячеек памяти. Таким образом, при выборе емкости ОП необходимо учитывать быстродействие ЦБУ, конкретные алгоритмы, которые им реализуются, число и длительность работы МФБ.

Если ЦБУ работает по второй дисциплине обслуживания, он продолжает обслуживание начатой заявки сразу же после поступления сигнала о том, что включенный МФБ закончил свою работу. Для этого ЦБУ, обратившись к очередной ячейке памяти, выполнив все быстродействующие операции и включив нужный МФБ, прежде чем перейти к опросу следующей ячейки памяти, должен опросить все включенные МФБ. Если какой-либо из них закончил работу, ЦБУ продолжает обслуживание той заявки, выполнением которой был занят этот МФБ. При этом для каждого МФБ в специальном регистре должен храниться номер той ячейки памяти, в которой записан номер этого МФБ, чтобы ЦБУ мог продолжить обслуживание заявки по нужному алгоритму и записать в данную ячейку номер следующего включенного МФБ.

Таким образом, каждый раз номер следующей команды, вырабатываемой ЦБУ, будет зависеть от того, какой МФБ подал сигнал об окончании своей работы. Разумеется, если таких МФБ несколько, должна быть организована определенная очередность их опроса.

При такой дисциплине обслуживания ЦБУ может приступить к обслуживанию новой заявки при условии, если ни в одной из уже обслуживаемых заявок не закончилось выполнение длительных операций, т. е. нет сигнала ни от одного из включенных МФБ. При этом число одновременно обслуживаемых заявок может уменьшиться, но время выполнения отдельных заявок может сократиться по сравнению с первой дисциплиной обслуживания. Такой принцип работы ЦБУ нашел применение в некоторых УС.

Следует отметить, что как при первой, так и при второй дисциплинах обслуживания, может возникнуть ситуация, когда при обслуживании i -й заявки потребуются включить j -й МФБ, который уже занят для выполнения другой заявки и еще не закончил свою работу. При этом i -я заявка либо может быть поставлена на ожидание, либо источнику этой заявки может быть дан отказ.

Ранее отмечалось, что при многопрограммном управлении параллельно могут выполняться не только различные алгоритмы, но и разные ветви одного и того же параллельного управляющего алгоритма. Очевидно, при этом возникает задача выявления таких ветвей алгоритма, которые могут выполняться одновременно (распараллеливание алгоритма).

Принцип многопрограммного управления может быть использован не только при наличии в выполняемых алгоритмах длительных операций, но и в тех случаях, когда заявки, которые обслуживает УС, неравноценны по своей значимости и задается некоторая система приоритетов. Тогда при поступлении заявки высокой категории ЦБУ переходит к ее обслуживанию и прерывает обслуживание заявки более низкой категории, записав в ячейку памяти этап, на котором она была прервана. Обслуживание этой заявки будет продолжено после того, как будут выполнены все заявки более высоких категорий. При этом ЦБУ будет работать так же, как и при наличии МФБ, только обслуживание заявки здесь будет прерываться не естественно, в момент включения МФБ, а принудительно, в любой момент, когда поступит заявка более высокой категории.

5.4.3. Преобразование управляющего алгоритма при его реализации в многопрограммном УС

В связи с тем что в многопрограммном УС кроме ЦБУ имеются активные МФБ, в которых параллельно с ЦБУ могут выполняться отдельные или группы операций алгоритма, требуется так преобразовать управляющий алгоритм, чтобы обеспечить ЦБУ в соответствии с управляющей информацией, хранящейся в его ОП, возможность последовательного выполнения различных управляющих алгоритмов или подалгоритмов параллельного управляющего алгоритма.

Управляющий алгоритм представим в виде ЛСА. В дальнейшем нам удобно будет считать, что в ЛСА нет членов, которые могут выполняться одновременно. Очевидно, этого всегда можно добиться, объединив одновременно выполняющие члены в один оператор или в одно логическое условие.

Тогда при обычном программном принципе работы УС, если началось выполнение i -го члена ЛСА (т. е. осуществлена подача сигнала на включение i -го ФБ), нельзя начать выполнение $(i+1)$ -го члена ЛСА до тех пор, пока не будет получена информация об окончании работы его предшественника. При этом управляющий сигнал, вырабатываемый ЦБУ и включающий ФБ $_i$, и осведомительный сигнал, воздействующий на ЦБУ от ФБ $_i$, описываются одним членом ЛСА (A_i или p_i).

Если говорим, что выполнен оператор A_i , то, значит, от ЦБУ на блок ОФБ $_i$ поступил сигнал Z_{A_i} , этот блок закончил свою работу и

от него на ЦБУ поступил сигнал A_i (при синхронном режиме этот сигнал заменяется сигналом от тактового генератора).

При многопрограммном управлении ЦБУ, подав включающий сигнал на МФБ, отключается от него и осведомительный сигнал получает уже от другого ФБ, принадлежащего даже другому алгоритму. При этом управляющий и осведомительный сигналы МФБ, которые обычно описываются одним членом ЛСА, теперь должны быть разделены. И в ЛСА вместо ее членов, соответствующих МФБ, должны быть введены новые члены, характеризующие отдельно управляющие и осведомительные сигналы. Предварительно в каждой ЛСА, реализуемой в ЦБУ, должны быть выделены члены, соответствующие МФБ. Разделение операций на быстрые и медленные является условным, однако во многих практических задачах относительно некоторых операций можно сразу сказать, что для их выполнения потребуется гораздо больше времени, чем для выполнения других операций.

Рассмотрим пример ЛСА, отметив ее члены, соответствующие МФБ, индексом «м»:

$$\mathcal{A}_i = \downarrow^3 p_1 \uparrow^1 A_1 A_{2m} \downarrow^1 p_{2m} \uparrow^2 A_3 \downarrow^2 A_{4m} p_3 \uparrow^3.$$

Управляющий сигнал, поступающий из ЦБУ на i -й операторный МФБ (ОМФБ), будем обозначать C_i , а сигнал на включение i -го логического МФБ (ЛМФБ) — S_i . Сигналы C_i и S_i включают соответствующие МФБ и записывают в определенные разряды ячеек памяти сигналы о том, что эти блоки включены.

Сигнал о выполнении оператора A_{im} , поступающий от i -го ОМФБ в ЦБУ, будем обозначать K_i . Заметим, что i -й логический блок ЛМФБ выдает два сигнала: G_i , если в результате проверки выяснено, что $p_{im}=1$, и H_i , если $p_{im}=0$.

Сигналы K_i , G_i и H_i не могут воздействовать непосредственно на ЦБУ, так как в момент их появления он может работать в соответствии с другим алгоритмом. Эти сигналы записывают в определенные разряды ячейки памяти информацию об окончании работы МФБ. Для того чтобы эта информация могла быть использована ЦБУ, необходимо осуществить проверку этих разрядов ячейки памяти. Поэтому в ЛСА кроме операторов C_i и S_i вводятся логические условия $g_1, g_2, \dots, g_k$, где $k=l+2m$, если l — число ОМФБ и m — число ЛМФБ. Для сигналов от ЛМФБ отводятся два разряда ячейки памяти, при этом можно условиться, что для сигнала G_i ($p_i=1$) отводится всегда младший из этих двух разрядов. Тогда для работы по выписанной ранее ЛСА A_i ячейка памяти будет иметь вид, показанный на рис. 5.50,а.

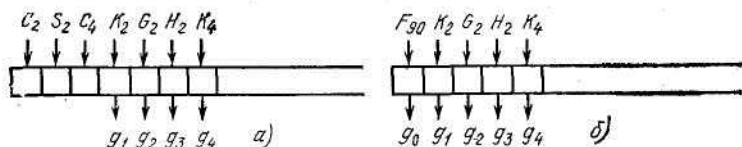


Рис. 5.50.

Однако эта информация является избыточной. Действительно, если включен второй МФБ (т. е. подан сигнал C_2), то может прийти только сигнал K_2 , после чего логическое условие g_1 примет истинное значение. И, наоборот, если $g_1=1$, это значит, что был включен второй МФБ. Для того чтобы значения переменных $g_1, \dots, g_k$ действительно характеризовали этап, на котором было прервано выполнение алгоритма, необходимо, чтобы в каждый момент не более чем одна из этих переменных принимала единичное значение. Поэтому каждый раз после считывания единичного значения переменной g_i разряд ячейки, в который она была записана, должен быть очищен. При этом сигналы о включении МФБ можно не записывать в ячейку памяти. Однако тогда потребуется еще один дополнительный сигнал, свидетельствующий о том, что данная ячейка памяти уже занята и начато выполнение определенного алгоритма. В противном случае, когда в ячейке памяти значения всех переменных $g_1, \dots, g_k$ равны нулю, нельзя будет определить, начата ли работа по данному алгоритму, но включенный МФБ еще не закончил работу, либо данная ячейка памяти еще свободна. Для формирования такого сигнала введем в ЛСА оператор F_{g_0} , который будет осуществлять запись сигнала о занятии данной ячейки в определенный ее разряд.

Содержимое этого разряда будем определять путем проверки логического условия g_0 . Если $g_0=1$, значит, ячейка занята и начато обслуживание поступившей заявки по требуемому алгоритму. Оператор F_{g_0} может выполняться вслед за первым членом ЛСА, соответствующим МФБ. Очевидно, после того как алгоритм полностью выполнен, все разряды ячейки памяти должны быть освобождены. Для этого в ЛСА введен оператор $F_{g_0}^*$.

Тогда ячейка памяти для алгоритма A_i принимает вид ячейки, изображенной на рис. 5.50,б, а ЛСА запишем в виде

$$\begin{aligned} \mathcal{A}_i = & g_0 \uparrow^1 \downarrow^6 \bar{g}_1 \uparrow^2 \bar{g}_2 \uparrow^3 \bar{g}_3 \uparrow^4 \bar{g}_4 \uparrow^5 \omega \uparrow^6 \downarrow^1 p_1 \uparrow^2 A_1 C_2 \\ & F_{g_0} \omega \uparrow^6 \downarrow^2 S_2 \omega \uparrow^6 \downarrow^3 A_3 \downarrow^4 C_4 \omega \uparrow^6 \downarrow^5 p_3 \uparrow^1 F_{g_0}^* . \end{aligned}$$

Конечно, в ячейке памяти могут быть и другие разряды, необходимые для записи некоторых дополнительных данных. Число таких разрядов зависит от конкретно выполняемого алгоритма и здесь не указывается.

Такая запись означает, что ЦБУ по-прежнему одновременно обслуживает только одну заявку, так как после включения МФБ внутреннее состояние МА не меняется и он находится в состоянии ожидания. Здесь только разделены сигналы включения МФБ и сигналы об окончании их работы. При многопрограммном управлении после выполнения оператора, включающего МФБ, ЦБУ переходит к обслуживанию следующей заявки, а выполнение начатого алгоритма заканчивается, т. е.

$$\mathcal{U}_i = g_0 \uparrow^1 \bar{g}_1 \uparrow^2 \bar{g}_2 \uparrow^3 g_3 \uparrow^4 \bar{g}_4 \uparrow^5 \omega \uparrow^6 \downarrow^1 p_1 \uparrow^2 A_1 \\ C_2 F_{g_0} \omega \uparrow^6 \downarrow^2 S_2 \omega \uparrow^6 \downarrow^3 A_3 \downarrow^4 C_4 \omega \uparrow^6 \downarrow^5 p_3 \uparrow^1 F_{g_0}^* \downarrow^6.$$

Рассмотрим теперь работу ЦБУ при многопрограммном управлении. Пусть имеется N источников, от которых на УС поступают заявки на обслуживание. Каждая заявка может быть обслужена по одному из следующих трех алгоритмов:

$$\mathcal{U}_1 = \downarrow^3 p_1 \uparrow^1 A_1 A_{2M} \downarrow^1 p_{2M} \uparrow^2 A_3 \downarrow^2 A_{4M} p_3 \uparrow^3; \\ \mathcal{U}_2 = A_6 \downarrow^2 A_{5M} p_1 \uparrow^1 A_3 p_{2M} \uparrow^2 \downarrow^1 A_1; \\ \mathcal{U}_3 = A_6 A_{2M} p_3 \uparrow^1 p_{2M} \uparrow^2 A_3 \downarrow^2 A_{4M} \downarrow^1.$$

Заметим, что эти алгоритмы можно рассматривать как частные алгоритмы одного параллельного управляющего алгоритма.

Для обеспечения одновременного выполнения нескольких заявок выберем L ячеек памяти и организуем работу ЦБУ следующим образом. Из ЦБУ поступают сигналы на последовательный опрос ячеек временной памяти. Для этого вводится оператор F_j , формирующий номер ячейки ($j=1, 2, \dots, L$). После того как найдена первая свободная ячейка ($g_0=0$), выдается сигнал на определение наличия заявок. Пусть имеется специальный блок F_1 , который выдает номер одного из вызывающих в данный момент источников и номер алгоритма, по которому эта заявка должна быть обслужена. Все эти данные одновременно с сигналом $F_{g_0}^*$ записываются в выбранную ячейку памяти. Затем начинается выполнение требуемого алгоритма до первого его члена, соответствующего МФБ. Подав сигнал на включение этого МФБ, ЦБУ выдает команду F_i и переходит к считыванию информации со следующей ячейки памяти. Если ячейка памяти уже занята ($g_0=1$), то анализируется состояние разрядов, в которых записываются значения переменных $g_1, \dots, g_k$. В зависимости

от этих значений либо выполняются определенные акты алгоритма, номер которого записан в данной ячейке, либо вновь выполняется команда F_j и начинается опрос следующей ячейки памяти.

Для каждого МФБ введем разряды ячейки памяти. Поскольку в различных алгоритмах имеются члены, соответствующие разным МФБ, некоторые разряды будут не заполнены в зависимости от того, для выполнения какого алгоритма занята в данный момент эта ячейка. (Возможно и другое распределение этих разрядов ячейки памяти, о котором скажем ниже). Для определения номера алгоритма, который записан в данной ячейке, введем логические условия $r_1, \dots, r_n$. Для нашего примера, когда имеются три различных алгоритма, достаточно двух логических условий r_1 и r_2 , т. е. для записи номера алгоритма будем использовать двоичный код. Пусть $R_1 = r_1 r_2$; $R_2 = \bar{r}_1 \bar{r}_2$; $R_3 = r_1 \bar{r}_2$.

На рис. 5.51 приведена структура ячеек памяти, занятых обслуживанием алгоритмов A_1 , A_2 и A_3 .

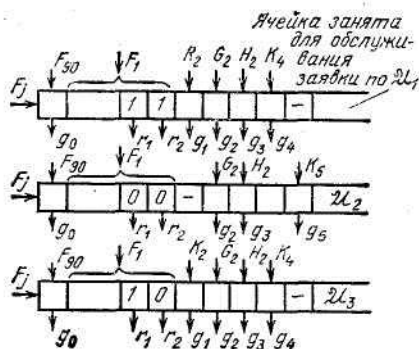


Рис. 5.51.

Ранее отмечали, что возможна такая ситуация, когда для обслуживания вновь поступившей заявки потребуется включить такой ФБ, который уже занят обслуживанием другой заявки. Поэтому в общем случае перед включением ФБ нужно вводить проверку его свободности и в зависимости от результатов этой проверки либо включить ФБ, если он свободен, либо ставить заявку на ожидание или давать отказ источнику заявки, если ФБ занят. Однако предположим, что среди одинаковых ФБ всегда найдется хотя бы один свободный, и не будем вводить специальных ЛУ для проверки состояния ФБ.

Тогда работу ЦБУ при многопрограммном управлении можно описать следующей ЛСА:

$$\begin{aligned} \mathfrak{A} = & \downarrow^4 F_j \bar{g}_0 \uparrow^1 F_1 F_{g_0} r_1 \uparrow^2 r_2 \uparrow^3 \downarrow^{11} p_1 \uparrow^7 A_1 C_2 \omega \uparrow^4 \downarrow^3 A_6 \\ & C_2 \omega \uparrow^4 \downarrow^2 A_6 \downarrow^{17} C_5 \omega \uparrow^4 \downarrow^1 r_1 \uparrow^5 r_2 \uparrow^6 \bar{g}_1 \uparrow^7 \bar{g}_2 \uparrow^8 \bar{g}_3 \uparrow^9 \bar{g}_4 \uparrow^{10} \\ & \omega \uparrow^4 S_2 \omega \uparrow^4 \downarrow^8 A_3 \downarrow^9 C_4 \omega \uparrow^4 \downarrow^{10} p_3 \uparrow^{11} F_{g_0}^* \\ & \omega \uparrow^4 \downarrow^6 \bar{g}_1 \uparrow^{12} \bar{g}_2 \uparrow^{13} \bar{g}_3 \uparrow^{14} \bar{g}_4 \uparrow^{15} \omega \uparrow^4 \downarrow^{12} p_3 \uparrow^{15} S_2 \omega \uparrow^4 \downarrow^{13} \\ & A_3 \downarrow^{14} C_4 \omega \uparrow^4 \downarrow^{15} F_{g_0}^* \omega \uparrow^4 \downarrow^5 \bar{g}_2 \uparrow^{16} \bar{g}_3 \uparrow^{17} g_5 \uparrow^4 p_1 \uparrow^{16} A_3 S_2 \\ & \omega \uparrow^4 \downarrow^{16} A_1 F_{g_0}^* \omega \uparrow^4. \end{aligned}$$

Как видно, в этой ЛСА много повторяющихся операторов и логических условий, поэтому ее целесообразно упростить, объединив одинаковые члены ЛСА, принадлежащие различным ЛСА. Метод объединения ЛСА изложен в 5.3.3.

Таким образом, все особенности, связанные с использованием многопрограммного принципа управления, проявляются в основном на этапе составления общего управляющего алгоритма. После выбора режима работы ЦБУ и способа распределения памяти в этот алгоритм приходится вводить новые операторы и ЛУ, которых не было ни в одном из алгоритмов обслуживания заявок.

5.4.4. Программно-аппаратурная реализация управляющего алгоритма

Управляющий алгоритм УС представляет собой *алгоритмическое описание*. Если по заданному управляющему алгоритму построена структурная схема УС в том или ином элементном базисе, говорят, что алгоритмическое описание переведено в *структурное* описание, представляющее собой структурную модель заданного управляющего алгоритма, т. е. алгоритма функционирования УС.

Альтернативой *структурному моделированию* может служить *программное моделирование*, при котором управляющий алгоритм представляют в виде совокупности (множеств) *команд*, каждая из которых определяет операцию над множеством исходных или промежуточных данных (операндов) и номер следующей команды. В качестве операндов могут рассматриваться как отдельные одно- и многоразрядные числа, хранящиеся в запоминающих устройствах (ЗУ), так и наборы значений входных сигналов (в частности, одного сигнала), которые также могут быть предварительно записаны в ЗУ в виде многоразрядных чисел.

Операции могут быть *одноместными*, т. е. выполняемыми над одним операндом, и *многоместными* (в основном двухместными), выполняемыми над несколькими операндами.

Совокупность команд, однозначно описывающая заданный управляющий алгоритм, называют *программой*, а представление управляющего алгоритма в виде программы — *программным моделированием алгоритма*. Для хранения программы может быть использовано ЗУ.

Следует заметить, что структурой УС, аппаратурно реализующей его алгоритм функционирования, обеспечивается активное выполнение последнего в соответствии с воздействующими на УС входными сигналами. В отличие от такой аппаратурной реализации при структурном моделировании управляющего алгоритма программа, реализующая тот же управляющий алгоритм при программном моделировании, является пассивной, т. е. программа является некоторой записью управляющего алгоритма. Для чтения этой записи и выполнения команд программы при программном моделировании используется универсальное **дискретное устройство** — *процессор*. Процессор обеспечивает выполнение любой, но в один и тот же момент одной операции, имеющейся в программе. При этом заметим, что в связи с успехами микроэлектроники широкое распространение получили *микропроцессоры*.

Таким образом, вместо структурной схемы устройства, аппаратно реализующего весь управляющий алгоритм при его структурном моделировании, при программном моделировании процессор (или микропроцессор), представляющий собой универсальное устройство, настраиваемое на аппаратурную реализацию одновременно только одной операции, обеспечивает выполнение всего управляющего алгоритма путем последовательного выполнения четко разграниченных отдельных операций (актов алгоритма), определяемых программой, хранящейся в ЗУ.

Очевидно, в УС при программном моделировании могут быть реализованы только те управляющие алгоритмы, которые удастся представить в виде программы, команды которой содержат операции, реализуемые в процессоре (микропроцессоре) данного УС. Однако на практике число выполняемых операций в процессоре или микропроцессоре имеется такое, что с их использованием можно описать любой управляющий алгоритм. Такой набор операций называют *функционально полным*. Вместе с тем функционально полные наборы могут заметно отличаться друг от друга составом операций. При этом один набор позволяет описать один класс алгоритмов более компактно, а другой набор — иной класс

алгоритмов. В этом случае говорят, что такие наборы операций являются проблемно-ориентированными, т. е. специализированными для описания для управляющих алгоритмов того или иного класса.

В УС широкое распространение получили *многопроцессорные* и *многомикропроцессорные системы*. Легко понять, что в таких системах одновременно в разных процессорах (микропроцессорах) могут выполняться несколько (по числу процессоров) команд. Поэтому такие системы могут быть эффективно использованы для одновременного выполнения в УС нескольких программ, соответствующих или нескольким управляющим алгоритмам, или нескольким параллельным ветвям одного параллельного управляющего алгоритма.

Структурное (аппаратурное) и программное моделирование являются крайними принципами реализации управляющего алгоритма. Среди промежуточных между ними принципов реализации управляющего алгоритма отметим два наиболее интересных: реализация структуры УС в базисе однородных сред и микропрограммный принцип.

На практике значительно большее распространение получил микропрограммный принцип реализации консультационного алгоритма. При этом следует отметить, что в отличие от программного управления при микропрограммном управлении вместо процессора используется некоторый *блок микропрограммного управления* (БМУ). В отличие от процессора он обрабатывает не считанные из ЗУ операнды, а снимаемые с управляемых объектов сигналы (параметры), на основе которых вырабатывает сигналы управления теми или иными управляемыми объектами. Такие управляющие сигналы называются *микрооперациями*, а их совокупность, выполняемая за один такт работы УС, — *микрокомандами*. Совокупность микрокоманд, обеспечивающая реализацию управляющего алгоритма, называется *микропрограммой*.

Каждая микрокоманда, включающая набор микроопераций и адрес следующей микрокоманды, может, как и команда, храниться в ЗУ (ЗУ микропрограмм) или реализоваться аппаратурно-структурной схемой в том или ином элементном базисе. Заметим при этом, что процессор УС может быть построен так, что выполняемые в нем команды реализуются или аппаратурно, или в блоке микропрограммного управления в соответствии с микропрограммами, определяющими элементарные действия процессора при выполнении той или иной команды. В последнем случае говорят, что УС имеет процессор с микропрограммным управлением. Наиболее часто микропрограммное управление используется в микропроцессорах.

Таким образом, программная и микропрограммная реализации управляющего алгоритма предполагают наличие программы (или микропрограммы) и аппаратных средств в виде процессора или микропроцессора, поэтому фактически такая реализация является программно-аппаратурной. Однако, как правило, ее называют просто программной, а термин программно-аппаратурной реализации используется в том случае, когда отдельные частные алгоритмы реализуются аппаратурно, например, в ФБ, а другие — программно в ЦБУ иерархической или распределенного УС.

6. Технические системы как средства реализации облачных технологий

Наиболее важными средствами реализации облачных технологий (наряду с людьми и реальным окружением) являются технические системы *TS*, выполняющие необходимые воздействия на операнды *Od*, в результате чего осуществляется реализация требуемого облачного технологического процесса. Таким образом, технические системы являются средством реализации облачных технологий.

Техника проявляет себя через изделия, подобно тому как искусство реализуется в художественных произведениях. Количество и многообразие изделий, создаваемых в различных отраслях техники, практически неисчислимо, что в значительной мере обусловливается машинными системами — изделиями машиностроения. Необозримое множество машин, приборов, оружия, транспортных средств, устройств, инструментов (как бы они ни назывались) может служить человеку либо вредить ему, либо вовсе не использоваться в зависимости от того, как человек обращается с ними.

Здесь мы попробуем упорядочить это разнообразие и сформулировать точки зрения, с позиций которых можно различать технические системы (особенно машинные системы), а также установить общие закономерности, справедливые для этого кажущегося необозримым множества технических систем. Удовлетворительного решения этой проблемы, несмотря на то что оно кажется само собой разумеющимся, в науке пока еще не найдено.

6.1. Сущность технической системы

При рассмотрении технической системы нужно прежде всего определить такие ключевые характеристики, как ее *назначение*, *способ*

действия и структура. При этом мы рассмотрим также вопрос о состояниях технической системы.

В причинно-механистическом смысле любая техническая система **подчиняется принципу причинности: каждое событие в технической системе имеет одну или несколько причин и одновременно является причиной ряда других событий. Без причины ничто не происходит.**

В философии принцип причинности, понимаемый как более или менее сильная взаимосвязь событий, является объектом обширной полемики. Однако эта полемика не относится к «машинной теории жизни» и не затрагивает область техники, особенно технических систем.

При более подробном рассмотрении причина распадается на три компонента ;

- обстоятельства, при которых что-то происходит;
- внутренние условия, при которых осуществляется событие;
- повод, который является непосредственной причиной.

«Действие» осуществляется тогда и только тогда, когда существует «причина».

Эти утверждения основываются на опыте людей и бесчисленных наблюдениях природных явлений, которые обобщены и исследованы в физике, химии, биологии и других науках. Понятие причинности сформулировано в результате обобщения этого опыта.

Людам свойственно стремиться к осуществлению желаемых событий и достижению своих целей. С точки зрения причинности определенное действие стремятся совершить именно там и тогда, где и когда в нем есть надобность. Кроме причинности можно говорить также о **целеустремленности**, наличие которой помогает человеку определенным образом формировать свою жизнь. Он использует в своих целях физические, химические, биологические и другие процессы в рамках существующих технико-экономических условий.

Исходя из принципа причинности, человек создает **причинные системы (цепочки действий)** и **собственно причины**, которые должны обеспечить требуемое действие в нужный момент времени. Описанный процесс развития от причин к следствиям для достижения поставленной цели имеет место при проектировании и конструировании технических систем, которые в смысле причинности реализуют систему причин и осуществляют действие. При этом должны соблюдаться условия функционирования системы, в том числе условия реального окружения. После этого философского экскурса в проблему сущности технических систем можно более подробно рассматривать вопрос о их назначении.

Назначение технической системы

Рассматривая вопрос о *назначении* технической системы, вернемся к модели системы преобразований (рис. 3.1). Как видно из этой модели, технические системы должны реализовывать запланированные целенаправленные воздействия на операнды технологического процесса. Технические системы выполняют бесчисленные и разнообразные действия типа фиксировать, двигать, хранить, нагревать, соединять, разделять, уплотнять, управлять и др., служащие для удовлетворения потребностей людей. Подобно функционированию и назначению, следует делать различие между технической и целевой функциями технической системы. Хансен и ряд других авторов техническую функцию системы определяют как способность «преобразовывать входные величины в требуемые выходные величины», тогда как задачи **технической системы правильнее характеризовать целевыми функциями. Целевая функция находится с технической функцией в отношении «цель — средство».** При конструкторских работах это различие замаскировано, особенно для технических систем низких уровней сложности, что зачастую сильно затрудняет понимание этих систем. Рополь при исследовании технических систем использует понятия телеологической (целевой) и дескриптивной (технической) функций. При дальнейшем изложении во избежание недоразумений понятие целевой функции использоваться не будет; «внешние» задачи системы мы будем определять ее назначением. Для обозначения «внутренних» задач системы будет сохранено понятие «техническая функция», так как оно уже достаточно прочно укоренилось в практике. **Назначение технической системы можно представить через систему ее воздействий.** Поэтому назначение технической системы часто называют ее совокупной функцией, что, хотя и подходит для некоторых технических систем, однако, вообще говоря, ошибочно. Позднее мы вернемся к этой проблеме.

Способ действия

В технических системах для реализации их назначения используются известные природные эффекты, например эффект рычага, гравитация, эффект расширения при повышении температуры, эффект электрического тока в проводнике, движущемся в магнитном поле, и другие физические, химические и биологические явления. Взаимодействие составных частей технической системы таково, что реализует внутренний технологический процесс и тем самым создает

требуемое внешнее воздействие. Так, используя зубчатую передачу, изменяют число оборотов и направление вращения. В качестве другого примера можно рассмотреть люминесцентную лампу: электрический ток преобразуется в нити накаливания в тепловую энергию, которая в слое окислов создает поток электронов, вызывающий в свою очередь ультрафиолетовое излучение ртути, трансформируемое в люминесцентном слое в световое излучение, вследствие чего достигается требуемый результат — свет. Такая причинная цепочка с превращением следствий (выходов) в причины (входы) следующих операций характеризует *способ действия* (способ функционирования) технической системы. Внутренние преобразования в технической системе либо описывают внутренними функциями системы и тогда изображают назначение системы в виде ее функциональной структуры, либо характеризует средствами (т. е. исполнительными органами), осуществляющими эти функции, и тогда способ действия системы может быть представлен как ее органоструктура. Функциональные способности каждой функциональной единицы (исполнительного органа) обеспечиваются множеством соответствующих функций, реализуемых органами более низкого уровня сложности. Подчеркнем, что при рассмотрении исполнительного органа как средства осуществления желаемого воздействия используют чисто функциональный подход. Отметим попутно, что ту или иную функцию можно осуществить различными исполнительными органами. При этом выбор исполнительного органа определяет функцию, выполняемую на ближайшем снизу иерархическом уровне. Исполнительные органы можно рассматривать на различных уровнях абстрагирования; тем самым определяется также и *степень абстрактности* соответствующих им функций.

Структура технической системы

Техническая система может быть создана только в том случае, если имеется возможность создать и желаемым образом объединить ее составные части. При этом посредством структуры должны быть реализованы (в возможно более полной мере) определенные свойства, обеспечивающие желаемое функционирование системы. Рассмотрение технических систем с точки зрения структуры приводит к понятиям структурных элементов и групп, которые находятся между собой в определенных геометрических, механических, энергетических и других отношениях. Подобный способ рассмотрения известен и понятен большинству людей, так как почти каждый, по крайней мере однажды, пытался самостоятельно провести ремонт велосипеда,

водопроводного крана или выключателя и при этом был вынужден разобрать ремонтируемый объект. Структура представляет собой как бы «рентгеновский снимок» объекта. В конструкторском деле обычно она характеризуется чертежом и спецификацией (рис. 6.1).

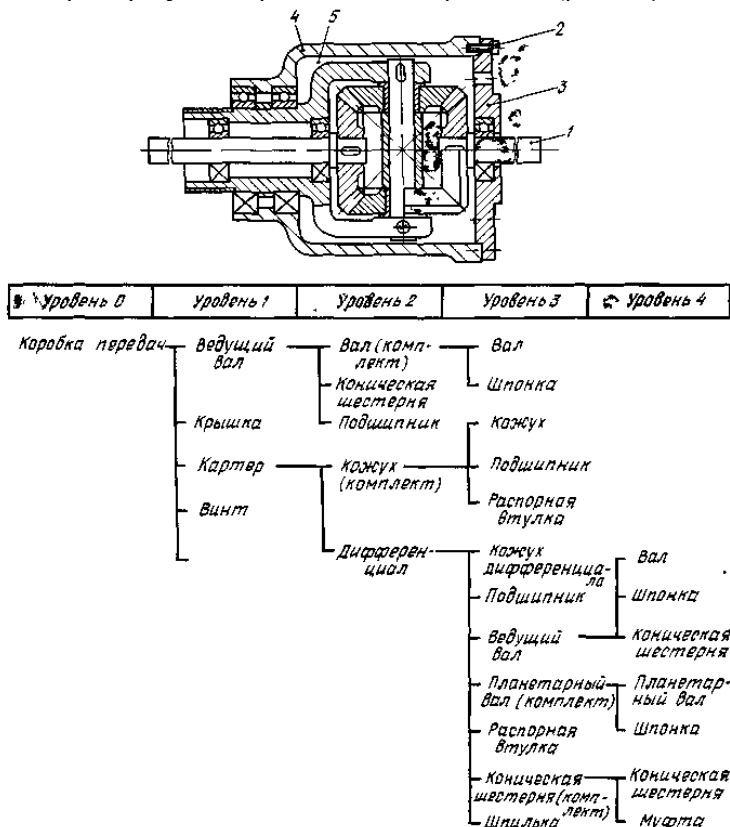


Рис. 6.1. Коробка передач и ее спецификация согласно стандарту ФРГ VSM80.

Структура объекта при этом расчленяется на элементы и группы в зависимости от принятой точки зрения (например, сборки или функционирования). Структурные группы четырех уровней в данном случае определяются требованиями изготовления (технологическая группировка).

Состояния технической системы

В ходе своего развития от возникновения и существования до ликвидации любая техническая система проходит ряд типичных состояний, обусловленных изменением состава системы преобразований. При обсуждении, моделировании или изображении технической системы необходимо сразу же указать соответствующие состояния для выбора правильной точки зрения и определения системы преобразований. В табл. 6. 1 систематизированы некоторые наиболее важные состояния системы с соответствующими пояснениями.

Таблица 6.1. Состояния технических систем

Состояния технической системы	Определение	Характеристика	
		абстрактная a , конкретная k	операнд Od , оператор Ot
1. Планирование	Планирование	a	Od
2. Возникновение	Конструирование, подготовка производства, изготовление	a	Od
3. Разборка	TS как ряд узлов или элементов (при изготовлении и монтаже)	k	Od
4. Сборка	TS как одно целое, в котором проявляются пространственные отношения	k	Od
5. Испытания	Оценка функциональной пригодности	k	Od, Ot
6. Хранение	Законсервированная TS	k	Od
7. Транспортировка	Изменение местоположения	k	Od
8. Использование	Рабочее состояние	k	Ot
9. Простой	Уборка или ремонт	k	Od
10. Ликвидация	Разделка в металлолом	k	Od

Инженер-конструктор должен представлять себе эти состояния, для того чтобы установить, в каком из них находится конструируемая им система. Там, где недостаточно одного воображения, полезны эксперименты на модели.

6. 2. Модель технической системы

Теперь целесообразно формализовать рассуждения раздела 3 с целью получения общей модели, связывающей модели системы преобразований (рис. 3. 1) и технологического процесса (рис.3.3.) и отражающей сущность технической системы (рис. 6.2).

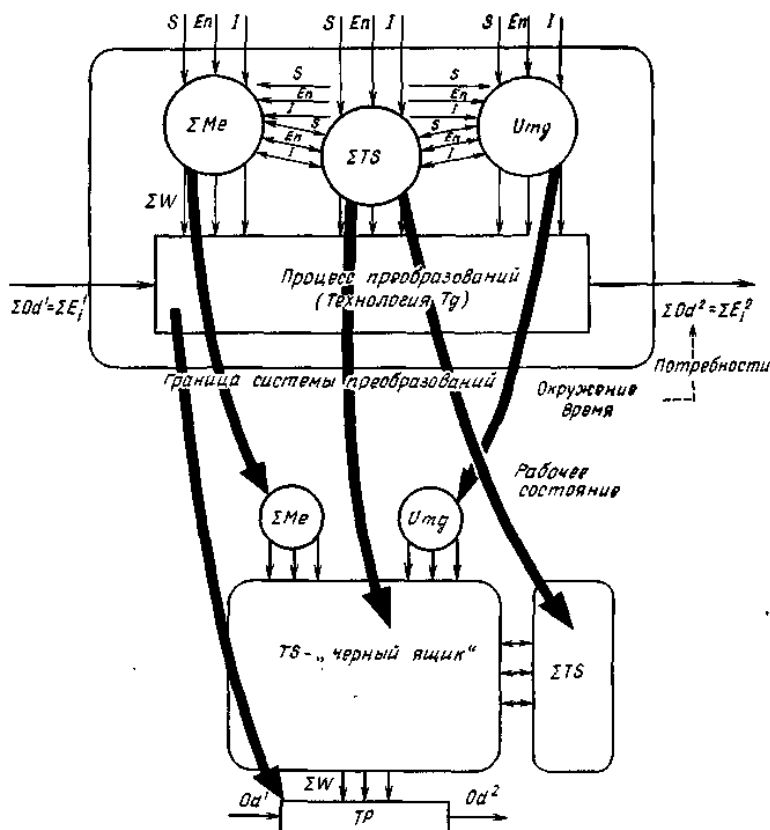


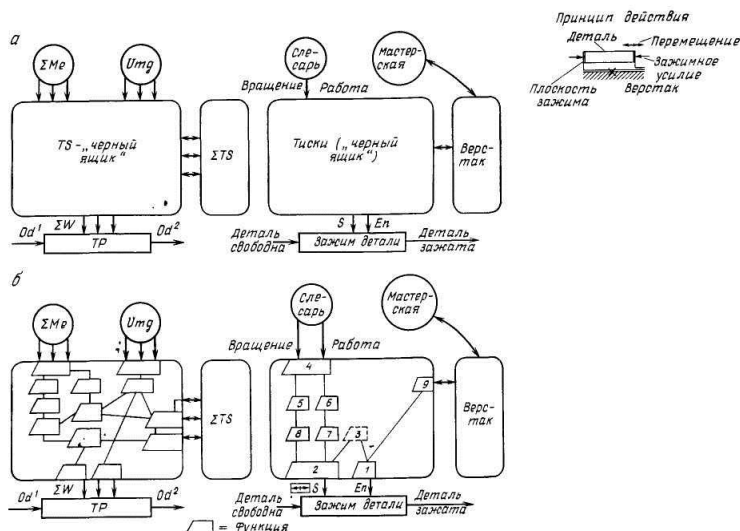
Рис. 6.2. Получение модели технической системы из модели системы преобразований.

Как элемент системы преобразований техническая система взаимосвязана со всеми остальными элементами системы преобразований, а именно:

- с операндами (назначение технической системы реализуется посредством воздействий, т. е. осуществляется преобразование $Od^1 \rightarrow Od^2$);
- с людьми-операторами (например, рабочие и вспомогательные действия);
- со следующей технической системой в системе преобразований;

— с реальным окружением (все технические системы имеют непосредственную или опосредованную связь с геосферой и другими системами).

Кроме указанных нужных связей имеются еще нежелательные и не всегда определенные побочные воздействия окружения, называемые помехами. Все внешние действия технической системы могут быть названы активными; каждому воздействию на техническую систему соответствует действие технической системы (по принципу «воздействие — реакция»). Эту ситуацию иллюстрирует рис. 6.3, а (случай «черного ящика»).



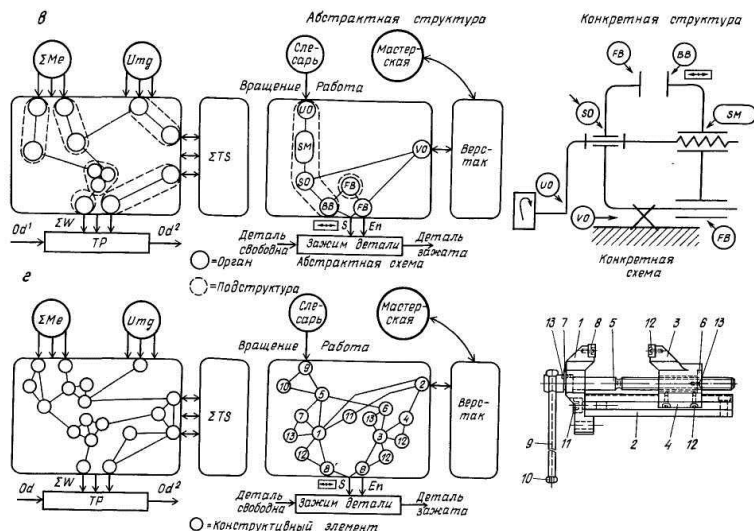


Рис.6.3. Модели технической системы.

а—техническая система как «черный ящик», б — функциональная структура, в — органоструктура, г — конструктивная схема

Теперь рассмотрим вопрос: каким образом осуществляется запланированное воздействие на операнд? Возможны несколько вариантов ответа, а именно, воздействия могут быть осуществлены, если:

- 1) имеются определенные входные воздействия (людей, технических систем или реального окружения), которые можно рассматривать как входы технической системы, и
 - 2а) с помощью технической системы выполняются определенные функции, причем эти функции зависят от отношений вход — выход и от выбранного способа действия, или
 - 2б) имеются определенные технические средства (исполнительные органы) и их отношения, выполняющие функции п. 2а и соответствующие отношения, или
 - 2в) имеются определенные конструктивные элементы и их отношения, реализующие органы п. 2б с их отношениями.
- Отметим, что варианты п. 2а—2в соответствуют технической системе на различных уровнях абстрагирования, а именно:
- 2а — на уровне функциональной структуры (рис. 6.3,б);
 - 2б — на уровне органоструктуры (рис. 6.3,в);
 - 2в — на уровне конструктивной схемы (рис. 6.3,г).

Для того чтобы эти определенные лишь предварительно понятия стали яснее, на примере простой технической системы — тисков — рассмотрим различные типы структур, дополнив их общей моделью.

№ п/п (рис. 5.3, б)	Функция	Исполнитель- ный орган (рис. 5.3, а)
1	Создать ограничение в виде неподвижной поверхности	FB
2	Создать ограничение в виде подвижной поверхности	BB
3	Подвести подвижную зажимную поверхность к неподвижной (отношение между функциями 1 и 2)	BB→FP
4	Установить деталь	UO
5	Превратить вращательное движение винта в поступательное движение рабочих поверхностей	SM
6	Создать усилие	SM
7	Зажать деталь	SO
8	Зафиксировать ручку тисков	SM
9	Проверить надежность зажима и фиксации	VO

Тем читателям, которым приведенные понятия абстрактных структур непривычны, следует начать с хорошо известной конструктивной схемы, а затем вернуться к абстрактным структурам.

Возможности символического представления системы не исчерпываются этими тремя структурами. Теоретически можно представить любое количество таких структур. Причина того, что наряду с конструктивной схемой обычно используются еще две абстрактные структуры, лежит в методологии конструирования. Функциональная структура довольно широко известна из литературы. Взаимосвязи между рассматриваемыми структурами системы легко понять, используя отношение «цель — средство». С этой точки зрения назначение технической системы (как цель) обеспечивается определенной функциональной структурой (как средством); эта функциональная структура (как цель) может быть реализована различными органоструктурами (как средством); органоструктуры (как цель) могут быть реализованы различными конструктивными схемами. В следующих разделах мы перейдем к более подробному рассмотрению упомянутых выше структур технических систем и их отношений.

6.3. Функциональная структура технической системы

В этом разделе технические системы рассматриваются с точки зрения выполняемых ими функций. **Функциональная структура** определяется как упорядоченная совокупность функций и отноше-

ний между ними. Функциональная структура строится применительно к рабочему состоянию системы.

Техническая функция

Техническая функция — это способность системы при определенных условиях преобразовывать входную величину в требуемую выходную величину, обеспечивая четкое соответствие зависимой выходной величины от независимой входной.

Отношения между функциями

Имеются в виду связи, взаимозависимости между отдельными функциями. Показанная на рис. 6.3 функциональная структура тисков может быть представлена в словесной форме следующим образом: энергия вращательного движения передается ручке (функция 4), энергия преобразуется с увеличением силы (6), вращение преобразуется в поступательное движение (5), а поступательное движение вместе с силой передается тискам (2). В результате получается функциональное описание способа действия технической системы.

Классификация технических функций

Из множества возможных характеристик функций рассмотрим следующие (особенно важны первые три из них):

1. *Сложность функции.* Каждая техническая функция характеризуется определенной степенью сложности. Самую нижнюю ступень в иерархии сложности занимают элементарные функции, которые дальше упростить уже нельзя.
2. *Степень абстрактности функции.* Техническую функцию можно описать с разной степенью конкретности (абстрактности), что влияет на количество возможных средств для ее выполнения (носителей функции). Например, если рассматривается функция «изменение движения», то диапазон средств для ее реализации очень широк. С конкретизацией условий и требований к функции (здесь могут появиться конструктивные признаки) диапазон средств все более сужается до тех пор, пока не остается конкретная техническая система. Пример иерархии технических функций по - степени их абстрактности показан на рис. 6.4.

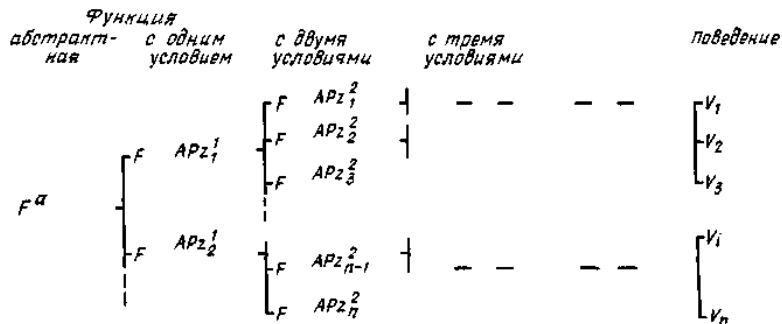


Рис. 6.4. Степень абстрактности функций технических систем.

Отметим, что степень абстрактности связана со степенью сложности технической функции. Необходимость разложения функции на подфункции, т. е. на функции с меньшей степенью сложности, появляется на определенных уровнях абстрагирования. В нашем случае она возникает, когда установлен способ действия (рабочий принцип) системы. Конструкторы, знакомые с методологией конструирования, увидят здесь аналогию с **морфологической матрицей**, когда переход от функции к носителю функции (исполнительному органу) осуществляется через способ действия. Остальные признаки при этом не рассматривают, считая их на этом уровне проектирования второстепенными.

3. **Назначение функции.** Рабочей функции, выполняющей прямое назначение технической системы на основе выбранного способа действия, обычно сопутствует ряд других функций, обеспечивающих или облегчающих ее осуществление. К таким функциям относятся вспомогательные функции подготовки, обслуживания, управления, регулирования и согласования.

Наряду с тремя рассмотренными характеристиками функций мы дадим определение еще нескольким часто используемым понятиям, связанным с функциями.

4. **Логическая функция.** Логическая функция преобразует одну или две независимые переменные величины в зависимые, которые могут принимать только два значения (например, 0, 1).

5. **Обобщенная элементарная функция.** Это элементарная функция, образующаяся при объединении операций общего характера (накопление, передача, превращение) с объектами таких обобщенных категорий, как человек, материя, энергия и информация.

6. *Нормативная элементарная функция.* Это элементарная функция, соответствующая стандартным операциям объединения, разделения и управления.

7. *Физическая элементарная функция.* Это функция, соответствующая двенадцати основным физическим операциям, к которым относятся: выпускать, отделять, собирать, подгонять, преобразовывать, умножать, направлять, соединять, подключать, изменять направление, устанавливать, хранить.

Функции, определенные в пп. 4—7, являются составными элементами функциональной структуры. Элементарные функции определяются тем соображением, что в технической системе могут существовать только материальные, энергетические и информационные потоки. Пример функциональной структуры представлен на рис. 6.5.

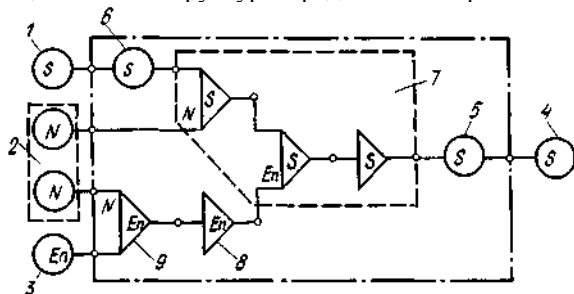


Рис. 6.5. Функциональная структура типа электрической схемы (кофемолка).

Декомпозиция функции

Теперь рассмотрим, как можно расчленить функцию более высокого уровня иерархии на подфункции. Решение этого вопроса далеко не настолько просто и не само собой разумеется, как это может показаться на первый взгляд. Мы выделим три основных класса подфункций:

- а) подфункции, которые определяются выбранными способами действия;
- б) подфункции, которые обуславливаются данной функцией как обязательные (функции этих двух классов определяются при разработке функциональной структуры или органоструктуры);
- в) подфункции, которые обеспечивают реализацию тех или иных свойств. Например, подфункция «закрепление» определяется необходимостью монтажа или транспортировки, а «изоляция» —

требованиями техники безопасности. Подфункции этого класса определяются при переходе от органоструктуры к конструктивной схеме.

Изображение функциональных структур

На рис. 6.3,б показана функциональная структура, иллюстрирующая функциональную связь элементов системы преобразований. Типовое упорядочение функций в функциональной структуре может быть исследовано на модели функциональной структуры. Показанное на рис. 6.3 изображение не является единственно возможным, и в литературе можно найти многочисленные примеры других форм. Специальную форму имеет, в частности, функциональная структура, показанная на рис. 6.5, которая по внешнему виду напоминает электрическую схему. Одним из вариантов функциональной структуры является схема функциональной декомпозиции (дерево функций), в которой показывается иерархия функций и не учитываются их отношения. Такая схема очень полезна для наглядности.

Построение функциональных структур и относящиеся к нему выводы

Функциональные структуры можно строить двумя путями. В первом случае исходят из представления технической системы в виде «черного ящика», т. е. функции и их отношения определяются как средства реализации воздействий в технологическом процессе. При этом задаются входы и способ действия, после чего осуществляется построение функциональной структуры на том или ином уровне абстрагирования и полноты.

Во втором случае по органоструктуре или конструктивной схеме строят детальную функциональную структуру с разнообразными функциями и подфункциями либо, полагая некоторые функции несущественными с точки зрения преобразований, получают структуру, аналогичную случаю «черного ящика». При этом в функциональной структуре уже оказывается зафиксированным ряд конструктивных признаков. Так, выходные воздействия технической системы задаются определенной технологией преобразований, а способ действия определяется местом, характером действия и условиями его выполнения (табл. 6.2).

Таблица 6.2.

Действия технических систем: признаки и их воплощение

Характеристика действия	I уровень		II уровень	
	Признак	Воплощение	Признак	Воплощение
Действие	Вид	Рабочее движение	Вид	Степени свободы (см. рис. 5.8)
		Воздействие, силы	Направление	$\rightarrow, \leftarrow, \leftrightarrow$
			Параметры	Скорость, ускорение
		Воздействие тепла	Протекание во времени	Длительность, частота
Место действия	Вид	Воздействие поля	Количество	1...n
			Величина	Ед. силы
		Пространство, поверхность, линия, точка	Направление	$\rightarrow, \leftarrow, \leftrightarrow$
		Фиксированное	Протекание во времени	$F = f(t)$
Условия действия	Вид	Среда	Количество	1...n
		Тело	Параметры	Тепловая энергия, температура
		Горизонтальное, вертикальное		Электрическое, магнитное
		Ед. длины	Вид	
Условия действия	Вид	Твердость, качество обработки		
		Газ, жидкость		
		Ед. давления		
		Ед. температуры		
Условия действия	Вид	Электрическое, магнитное		
				
				
				

Количество такого рода конструктивных признаков, определяемых функциональной структурой, зависит от степени абстрактности и характеризует степень полноты описания технической системы (см. рис. 6.9).

6.4. Органоструктура технической системы

Органоструктура — это абстрактная модель технической системы, содержащая технические средства (исполнительные органы) и отношения, реализующие способы действия определенного класса. На

стадии разработки органоструктуры вопрос о конструктивной реализации технической системы не рассматривается. Органоструктура разрабатывается и изображается применительно к рабочему состоянию технической системы; она содержит по меньшей мере две структуры — структуру цепочек действия и структуру связей.

Исполнительный орган

Исполнительным органом называется подсистема, которая реализует определенную внутреннюю функцию технической системы.

Исполнительный орган иногда называют также носителем функции или функциональной единицей. Понятие органа в технике заимствовано из биологии.

Аналогично функциям и конструктивным элементам исполнительный орган может быть определен на различных уровнях абстрактности.

Очень абстрактно этот термин может быть использован для обозначения группы носителей функции (например, опора), а более конкретно — для обозначения, например, винта в конструктивной схеме тисков. Соответственно может существенно различаться и графическое представление таких органов, оставаясь тем не менее символическим.

Отношения исполнительных органов

Основным типом отношений исполнительных органов являются связи: выход одного органа одновременно является входом следующего в цепочке действий. Связи между исполнительными органами могут быть различными. Указанные отношения проявляются уже в абстрактной структуре, а в изображении способа действия они представляют собой конкретные связи исполнительных органов. В принципиальной схеме органоструктуры должны быть определены, хотя бы приближенно, пространственные связи технической системы.

Классификация исполнительных органов

Насколько близки понятия исполнительных органов и функций, настолько же аналогичны способы их классификации и точки зрения, с которых они рассматриваются.

1. *СЛОЖНОСТЬ*. В зависимости от степени сложности исполнительных органов различают такие понятия, как структура и подструктура

(группа) исполнительных органов, исполнительный орган и подорган.

2. *Степень абстрактности.* Аналогично функциональной структуре исполнительный орган может быть определен как абстрактная или конкретная функциональная единица.

3. *Назначение.* Исполнительные органы могут выполнять функции преобразования, передачи энергии, управления или связи (согласования).

4. *Положение.* В зависимости от положения исполнительного органа относительно границы технической системы различают граничный исполнительный орган (рецептор или эффектор) и внутренний исполнительный орган.

Органоструктура преобразований

Отдельные исполнительные органы можно объединить по их отношению к преобразованиям и получить структуру более высокого уровня — органоструктуру преобразований (рис. 6.6), подобную совокупности функций технической системы.

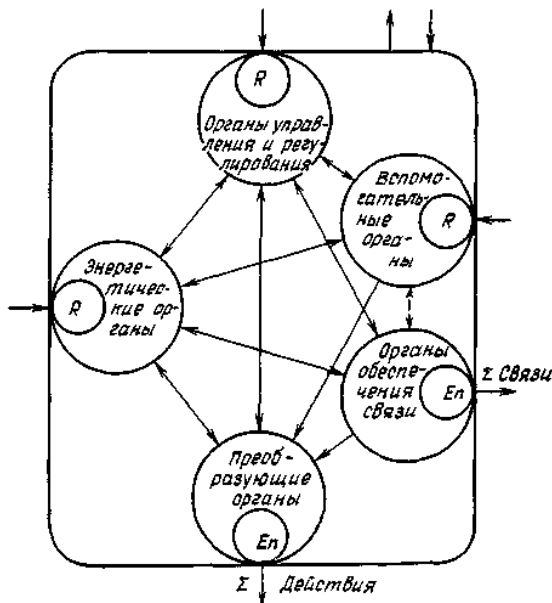


Рис. 6.6. Органоструктура преобразований

Виды органоструктур

Органоструктурное представление вызовет определенные затруднения у тех специалистов, которые привыкли расчленять машинную систему на конструктивные элементы, так как членение органоструктур отличается от привычного: например, болт и гайка, пара зубчатых колес с валами и опорами в органоструктурах не рассматриваются как отдельные детали, а связываются в структурные единицы. Имеются некоторые специальные области, в которых понятие исполнительного органа является довольно привычным.

В электротехнике исполнительными органами являются рубильник, конденсатор, реле или резистор; в гидравлике и пневматике исполнительными органами служат рабочий цилиндр, отсечной клапан, регулятор давления и т. п. Эти исполнительные органы объединяются в органоструктуру типа «электрической схемы». В рассмотренных случаях, конечно, ситуация несколько проще, так как здесь исполнительные органы одновременно являются конструктивными единицами.

Изображение органоструктуры также полезно при проектировании производственного комплекса. Структурными элементами соответствующей органоструктуры могут быть технические системы третьего уровня сложности — машины и аппараты в качестве исполнительных органов. В некоторых областях техники для разработки органоструктур развит специальный символический язык. Аналогичные примеры можно найти в строительном деле.

Структурирование строений на помещения, проводимое архитекторами, в принципе не что иное, как разработка органоструктуры.

Изображение органоструктур

Исполнительный орган — это система типа «объект», и, согласно принятым обозначениям, она изображается круговыми символами. Существуют различные представления органоструктур, часто связанные со специальными стандартными языками. Так, в ФРГ, например, стандартно изображают (DIN — германский промышленный стандарт):

- полупроводниковые элементы (DIN 40700);
- электротехнические схемы (DIN 40710 и т.д., SEV 9001);
- гидравлические схемы (DIN 24300, VSM 10332);
- монтажные схемы (DIN 40717);

— трубопроводы и арматура (VSM 10330) (рис. 6.7);

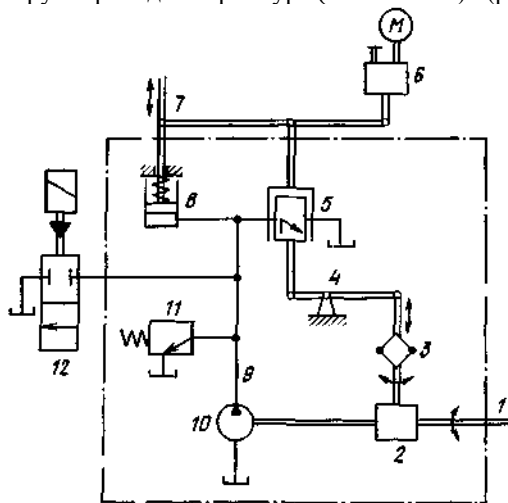


Рис. 6.7. Органоструктура типа монтажной схемы (регулятор турбины). 1— вал турбины, 2— редуктор, 3—центробежный регулятор, 4 — балансир, 5 — дроссель; 6 — механизм ручной регулировки, 7 — регулирующий рычаг, 8 — цилиндр, 9 — напорный трубопровод, 10 — насос, 11 — предохранительный клапан, 12— запорный клапан

— теплоэнергетические установки и теплосхемы (DIN 2481).

Построение органоструктур

Органоструктуру, по аналогии с функциональной структурой, можно построить из конструктивной схемы или функциональной структуры рассматриваемой технической системы. На основе функциональной структуры определяются некоторые конструктивные признаки (либо полностью, например, входы/выходы на стыках с другими системами, либо частично), способ действия технической системы и реализующие его исполнительные органы, отношения исполнительный орган — операция, а также примерное пространственное размещение технической системы. При построении органоструктуры на основе конструктивной схемы, наоборот, абстрагируется от некоторых конструктивных признаков, и при этом теряется некоторая доля полноты.

6.5. Конструктивная схема технической системы

Разработка конструктивной схемы технической системы представляет собой начальную стадию конкретного конструирования, и поэтому конструктивная схема должна отражать все свойства, установленные в качестве требований к этой системе, и условия, необходимые для ее реализации. При этом количество критериев оценки по сравнению с теми, которые выбирались и использовались при рассмотрении функциональных аспектов, резко возрастает.

Конструктивная схема представляет собой описание технической системы конструктивными элементами; она должна содержать **упорядоченную совокупность конструктивных элементов и их отношений**. Конструктивные схемы системы разрабатываются для рабочего состояния, а также для состояний «разборки» и «сборки» (деталировочные и компоновочные чертежи).

Конструктивный элемент

Конструктивная схема системы строится из конструктивных элементов. Конструктивный элемент может иметь любую степень сложности в зависимости от уровня сложности системы, степени абстрактности ее представления и происхождения конструктивного элемента (изготавливаемый или закупаемый). Для крупного производственного комплекса это может быть сооружение или сложная производственная линия, а для механизма — простой стержень. Относительность понятия «конструктивный элемент» еще более возрастает, если во внимание принимается такой аспект, как организация производства. Это хорошо видно из спецификации к рис. 6.1; так, спецификация коробки передач предусматривает четыре уровня структуры объекта.

Описание стандартизованных конструктивных элементов регламентируется. Например, стандартное обозначение винта в ФРГ может выглядеть следующим образом: M10X40, DIN912. Изготавливаемые конструктивные элементы характеризуют их конструктивными свойствами — формой, материалом, размерами, допусками и качеством поверхности.

Отношения между конструктивными элементами

Для состояния «сборка», соответствующего выходу из стадии «конструирование» (или «изготовление»), в конструктивной схеме технической системы существуют три типа отношений:

- пространственное отношение (определяется расположением элементов технической системы в пространстве);
- механические отношения (связи) отдельных элементов (характеризуются степенями свободы, см. рис. 6.8);

№	Степени свободы			Примеры известных механических связей	Символическое представление
	Перемещение	Вращение R	Σ		
1		0	0	Жесткое соединение	
2		1	1	Движение по направляющей	
3		0	1	Соединение с вращающимся валом	
4		2	2		
5		0	2	Карданный подвес	
6		3	3		
7		0	3	Шаровое соединение	
8		1	2	Соединение с вращением и перемещением	
9		1	2		
10		2	3		
11		1	3		
12		2	3		
13		1	3		
14		2	4		
15		2	4		
16		3	4		
17		1	4		
18		3	5		
19		2	5		
20		3	6	Две несвязанные детали	

Рис. 6.8. Механические связи с различными степенями свободы.

— энергетические отношения, которые задаются, в частности, силовым взаимодействием между деталями (например, задание силы вращением винта).

Из конструктивных элементов и их пространственных отношений автоматически вытекают некоторые энергетические и другие виды связей (например, по уплотнению, электроизоляции, теплоизоляции).

Классификация конструктивных элементов

На основе сформулированных выше точек зрения могут быть установлены следующие принципы классификации конструктивных элементов:

1. *Сложность*. Конструктивными элементами различной степени сложности являются деталь, узел (несколько уровней), машина, аппарат, предприятие (несколько уровней).
2. *Степень абстрактности*. Широкий диапазон степеней абстрактности, от абстрактного обозначения конструктивного элемента, охватывающего класс или группу элементов, до полного описания конструктивных элементов через конструктивные свойства (рис. 6.9).

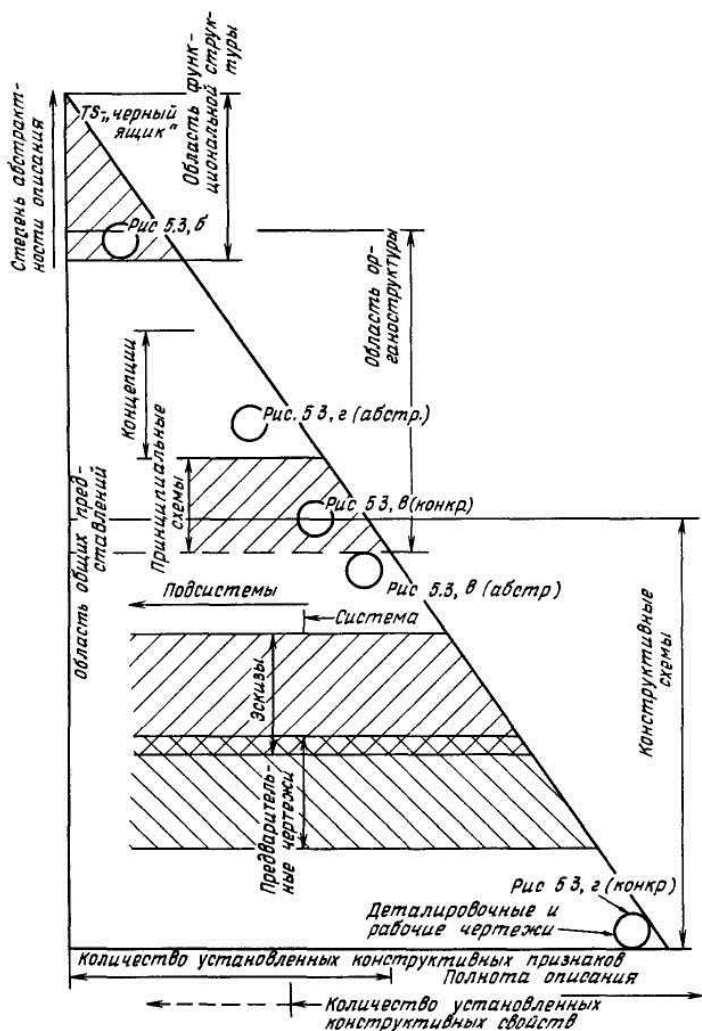


Рис. 6.9 Структура технических систем в координатах степень абстрактности — степень полноты.

3. Назначение. Широкий спектр назначений, характерных для различных областей техники. Определение машинных элементов в

машиностроении близко к принятому здесь определению конструктивных элементов.

4. *Положение.* В зависимости от положения конструктивного элемента относительно границы технической системы различают а) краевой или граничный конструктивный элемент и б) внутренний конструктивный элемент. Далее, нужно выделить внешние граничные элементы, которые лежат на внешней границе технической системы; они определяют место действия и непосредственно осуществляют воздействия. Кроме них, различают внутренние граничные элементы, расположенные на стыках между внутренними подсистемами.

Принципы построения конструктивной схемы

Конструктивная схема определяется не только типом воздействия, способом действия и свойствами операнда, которые она должна реализовать, но также и принципом ее построения. Типичным является, например, модульный принцип построения, который зарекомендовал себя не только при проектировании, но также при изготовлении и использовании технических систем. Часто комбинируют различные принципы, что позволяет сочетать их достоинства и исключать недостатки.

Изображение конструктивных схем

Поскольку представление конструктивной схемы в большинстве случаев является своего рода инструкцией по созданию технической системы, здесь в отличие от абстрактных структур уже давно разработаны нормы и правила, обеспечивающие единообразие. В еще большей степени регламентируются эскизные и конструкторские чертежи. В своем большинстве их изображение близко к стандартам для детализовочных чертежей, что обеспечивает простоту и единообразие показа связей между ними.

Разработка конструктивной схемы

Конструктивная схема может разрабатываться на основе либо функциональной структуры, либо органоструктуры. Вследствие сложности задачи конструирования переход от общих требований к конструктивным свойствам следует осуществлять методом последовательных приближений. В машиностроении,

например, конструктивная схема прорабатывается в три этапа: эскиз — конструкторские чертежи — детализовочные и сборочные чертежи. Значение этих конструкторских документов и их связь с другими структурными представлениями технической системы показаны на рис. 6.9. Конструкция, разработанная на основе конструктивной схемы, должна удовлетворять всем поставленным требованиям и воплощать все свойства, которые выявляются постепенно в процессе конструирования.

6.6. Сравнение структур технических систем и их преобразование

Три описанные выше структуры можно сравнить между собой по детальности описания технической системы в системе координат степень абстрактности — степень полноты. На рис. 6.9 представлена диаграмма, выявляющая качественные отношения между степенью абстрактности и степенью полноты (диагональ). На каждом уровне абстрагирования может быть достигнута большая или меньшая степень полноты описания технической системы путем установления тех или иных конструктивных признаков и свойств.

Для лучшего понимания взаимосвязей структур технической системы на рис. 6.9 отражены все структуры рис. 6.3. Техническая система может быть определена полностью только на стадии конкретного конструирования путем задания всех конструктивных признаков и свойств. Таким образом, ясно, что при проектировании нужно учитывать сложность технической системы, поскольку конструктивные признаки и свойства могут быть определены лишь в том случае, когда разработка системы (простой или сложной) доведена до уровня конструктивных элементов.

Описанная картина дает лишь общую ориентацию в процессе проектирования, так как количественная мера для таких понятий, как абстрактность и полнота, точно не установлена и установить ее трудно. Преобразование одной структуры в другую осуществляется путем установления некоторых (или абстрагирования от некоторых) конструктивных признаков или свойств. Так как каждый такой признак можно воплотить по-разному, в результате каждого шага при переходе от абстрактной к конкретной структуре будут возникать несколько вариантов. Таким образом, отношение между рассматриваемыми структурами является гомоморфным.

Именно преобразования структур технических систем и составляют содержание большинства инженерных задач, так что теория

технических систем приобретает особое значение в методологии проектирования и конструирования, позволяя разработать оптимальный алгоритм процесса.

Обзор конструктивных признаков для рассмотренных выше четырех видов структур технических систем содержится в табл. 6.3.

Таблица 6.3.

Конструктивные признаки технических систем¹

Структура	Характеристика	Примеры признаков
Технический процесс («черный ящик»)	Признаки, которые обосновывают и поясняют вид воздействия <i>TS</i> (как целого), вспомогательные действия и условия (рис. 4.1, табл. 4.1 и 5.2)	Технологический принцип Технология Последовательность частичных преобразований Степень участия <i>TS</i> в <i>TP</i> Количество <i>TS</i> в <i>TP</i> Действия Места действия Условия
Функциональная структура	Признаки, которые обосновывают и характеризуют внутренние функциональные цепочки, в которых требуемыми входами достигаются воздействия (рис. 5.3, б)	Входы, основные и побочные Цепочки действий Способы действия Выполняемые функции Степень механизации Степень автоматизации Агрегация функций Разделение функций
Органо-структура	Признаки, которые обосновывают и характеризуют внутренние цепочки действия (внутренние задачи), предназначенные для выполнения функциональных цепочек (рис. 5.3, в)	Принцип действия Выполняемые функции и исполнительные органы Агрегация исполнительных органов Разделение исполнительных органов ... Предварительное упорядочение в пространстве мест связи с другими <i>TS</i>
Конструктивная схема	Признаки, которые обосновывают и характеризуют реализуемость <i>TS</i> и всех свойств при заданных условиях (рис. 5.3, г)	Вид схемы Выполняемые функции Прочность, жесткость, упругость, износ, твердость, коррозионная стойкость Температурная стабильность ... Группы конструктивных элементов ...

¹ Конструктивные признаки реализуются через конструктивные свойства. Конструктивные свойства формулируются так же, как требования.

Отношение цель — средство

Мы уже неоднократно сталкивались с этим важным для нас отношением; теперь мы рассмотрим его более подробно. При обсуждении причинности упоминалось понятие целесообразности. Отношение **цель — средство** отражает процесс синтеза, при котором, исходя из поставленной цели (разработка системы), можно найти средства, пригодные для ее выполнения. Рассматривая элементы системы целей, необходимо каждый раз повторять вопрос о средствах (чем?), прежде чем может быть осуществлен переход к уровню конструктивных элементов, которые можно уже описать полностью. При этом процесс проектирования можно либо описать в наиболее общем виде как последовательность шагов **цель — средство** (в которой на каждом последующем шаге средство становится целью), либо определить требуемое действие системы как цель и задавать техническую или машинную систему как средство. На рис. 6.10 показан такой итеративный процесс и выделены отдельные шаги — «чем?».

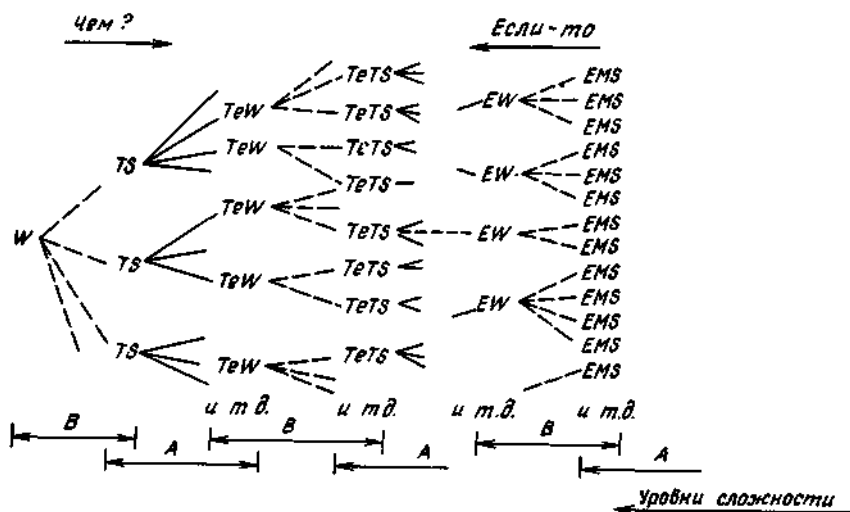


Рис. 6.10. Отношения цель — средство (действие W — техническая система TS).

Шаг «А»: для совокупного действия каждой TS необходим ряд частных действий. Шаг «В»: действие может быть реализовано различными TS . Элементарные конструктивные свойства элемента технической системы: форма, размеры, материал, способ изготовления, допуски, качество поверхности.

Эта схема одновременно указывает на связь целесообразности и причинности. Описанный подход можно усовершенствовать, если между целью и средством ввести обобщенный определяющий признак. Принципиальное описание поиска подходящей технической системы, т. е. процесса конструирования, можно назвать «решающим алгоритмом». Рассматривая структуры технических систем и соответствующие им понятия как цели, мы можем определить средства их реализации (шаг «чем»?) следующим образом:

<i>Цель</i>	<i>Средство</i>
Изменение операнда	Технология (частичные преобразования)
Технология	Действия
Действие (целевая функция)	Вход (техническая функция)
Техническая функция	Исполнительный орган
Исполнительный орган	Конструктивный элемент

При переходе от цели к средству возникают задачи воплощения конструктивных признаков, которые решаются сразу же или методом последовательных приближений.

6.7. Граница технической системы и их окружение

Граница технической системы

Граница технической системы определяет TS как пространственно-функциональную единицу. На внешней поверхности технической системы расположены граничные элементы (рецепторы и эффекторы), осуществляющие связи с окружением (входы, выходы). Они определяют место действия технической системы. На всех рисунках структур рецепторы и эффекторы изображаются непосредственно на линии границы (см., например, рис. 6.3).

Окружение технической системы

Как видно из рис. 6.3, в непосредственном окружении рассматриваемой технической системы всегда имеются следующие системы, находящиеся во взаимозависимости или взаимодействии (см. рис. 3.1):

— технологический процесс с операндами, которые являются объектами воздействия технической системы;

— человек-оператор (группа людей), находящийся в непосредственной связи с технической системой и участвующий в технологическом процессе или управляющий технической системой;

— технические системы, которые каким-либо образом влияют на преобразование операндов в технологическом процессе;

— реальное окружение, т. е. та часть общего окружения, которая находится в непосредственной связи с рассматриваемой технической системой. К реальному окружению относится также геосфера, связи с которой должны существовать всегда. Законы природы также входят в реальное окружение, т. е. техническая система находится в сфере действия этих законов.

Принадлежность отдельных компонентов к окружению технической системы, когда это не очевидно, следует определять на основе соответствующей системы преобразований и особенно технологического процесса. Окружение технической системы также зависит от определения системы преобразований. Любая техническая система принадлежит к нескольким системам преобразований и в зависимости от этого может рассматриваться в различных окружениях. Более того, окружение изменяется в процессе развития технической системы (см. табл. 6.1). Собственно говоря, виды окружения и определяют работоспособность системы.

Обобщим результаты, полученные в этом разделе, следующим образом.

1. Задачей (целью, назначением) технической системы является выполнение определенного воздействия в технологическом процессе.
2. Воздействия происходят на границе технической системы при определенных условиях (см. табл. 6.2).
3. Воздействие реализуется как выход процесса действия (цепочки действий), в котором входные величины преобразуются в выходные воздействия. Процесс действия определяется структурой технологического процесса, в связи с чем говорят о способе действия технической системы. Процесс действия вызывается непосредственной причиной и происходит в нужный момент (период) времени.
4. Процесс и способ действия характеризуются преобразованиями материальных, энергетических и информационных входов внутри технической системы. Внутренние преобразования называются техническими функциями и образуют, в рамках одной технической системы, ее функциональную структуру.

5. Техническая функция описывает способность технической системы преобразовывать входные величины в требуемые выходные путем упорядоченного использования природных явлений и законов.
6. Входными величинами технических систем являются материя, энергия и информация.
7. На стадиях возникновения, использования и ликвидации техническая система находится в различных состояниях и играет различные роли в зависимости от того, к какой системе преобразований она принадлежит.
8. Техническая система находится в определенных отношениях со всеми элементами соответствующей системы преобразований. Наряду с известными желательными входами и выходами имеются также и нежелательные входы и выходы, которые называются помехами.
9. Функции (функциональные структуры) реализуются исполнительными органами (органоструктурами) на основе определенных принципов действия. Исполнительные органы являются средством реализации функции. Для выполнения каждой функции может быть выбран желаемый принцип действия из нескольких возможных.
10. Исполнительные органы (органоструктуры) конкретизируются в виде конструктивных элементов (конструктивных схем). Конструктивная схема воплощает (в обобщенном виде) все требуемые свойства и признаки технической системы. Органоструктура может быть реализована различными конструктивными схемами.
11. Каждая из названных выше структур технической системы характеризуется большей или меньшей степенью воплощения тех или иных конструктивных признаков и свойств.
12. Каждая функция, каждый исполнительный орган или конструктивный элемент представляет собой систему подфункций, подорганов или конструктивных подэлементов, что вытекает из системного характера технических объектов и процессов.

6.8. Классификация технических систем

Определение «техническая система» было выбрано в качестве обобщающего термина для всех видов машин. В различных процессах технические системы осуществляют необходимые преобразования объектов действия (операндов). Область применения технических систем очень широка и включает в себя все отрасли экономики. В табл.

6.4 приведены примеры технических систем, используемых в важнейших отраслях экономики, и показано все их многообразие.

Таблица 6.4.

Примеры технических систем в различных отраслях экономики

Отрасль экономики	Техническая система	
	назначение	машина
Горное дело	Добыча Транспортировка Обогащение	Врубовая машина Транспортер Сортировальная машина
Энергетика	Выработка пара Выработка электричества	Паровой котел, барабан Паровая турбина, гидротурбина, генератор
Металлургия	Производство чугуна Производство стали Производство проката	Доменная печь Мартеновская печь Прокатный стан
Химическая промышленность	Очистка и переработка нефти Производство красителей Производство пластмасс	Резервуар Реактор Колонна
Фармацевтическая промышленность	Производство медикаментов	Пресс, каландр
Металлообрабатывающая промышленность	Обработка давлением Обработка резанием Термообработка Литье Сборка	Пресс, молот Станок Печь Формовочная машина Конвейер
Строительная промышленность	Строительство оснований и фундаментов Строительство надземных сооружений Земляные работы Гидротехническое строительство Производство стройматериалов	Экскаватор Подъемный кран Скрепер Бетономешалка Формовочный пресс
Транспорт	Железнодорожное сообщение Судоходство Воздушное сообщение	Локомотив, вагон Пароход Самолет
Текстильная промышленность	Производство текстиля Изготовление готового платья	Прядильная машина, ткацкий станок Швейная машина
Пищевая промышленность	Производство муки Производство пищевых жиров Переработка молока	Мукомольная мельница Пресс Центрифуга
Медицина	Диагностика Терапия	Рентгеновский аппарат Протез

Отрасль экономики	Техническая система	
	назначение	машина
Типографское и конторское дело	Печатание Конторские нужды	Печатная машина Пишущая машинка, счетная машина
Сельское и лесное хозяйство	Обработка земли Уборка урожая Заготовка древесины	Трактор с плугом Комбайн Электропила
Распределение, торговля	Самообслуживание Упаковка	Контрольная касса Упаковочная машина

Классификация технических систем по различным определяющим признакам вносит достаточно стройный порядок в их обширное множество и позволяет лучше ориентироваться. Как следствие этого появляется возможность изучения передового опыта, что позволяет подчас обнаружить между довольно далекими техническими системами интересные, доселе скрытые отношения. Перечислим для начала те аспекты, которые принимаются во внимание при проведении классификации (систематизации) технических систем, и приведем характерные примеры.

Технические системы могут быть классифицированы по следующим признакам:

- по функции (рабочему действию), например, технические системы для фиксации, придания формы, вращения, подъема;
- по типу операнда, например, технические системы для преобразования материи, энергии, информации, биологических объектов;
- по принципу осуществления рабочего действия, например, технические системы, основанные на механическом, гидравлическом, пневматическом, электронном, химическом, оптическом, акустическом принципе;
- по характеру функционирования, например, мощностные, скоростные, импульсные технические системы, системы для различных условий окружающей среды (например, для тропического климата) и т. п.;
- по уровню сложности, например, конструктивные элементы, узлы, машины, предприятия в целом;
- по способу изготовления, например, технические системы, изготовленные путем литья,ковки, штамповки, обточки;
- по степени конструктивной сложности;

- по форме, например, технические системы (конструктивные элементы) в виде тела вращения, плоские, сложной формы;
- по способу упорядочения более низких уровней технической системы, например, установки с упорядочением подсистем по способам их действия или технологии;
- по материалу, например, технические системы из стали, меди, пластмассы;
- по степени оригинальности конструкции, например, заимствованные, доработанные, модифицированные, оригинальные технические системы;
- по типу производства, например, технические системы, изготовленные в условиях единичного, серийного или массового производства;
- по названию фирмы-изготовителя, например, технические системы «Сименс», «Фиат», «Зульцер»;
- по месту в технологическом процессе, по эксплуатационным свойствам, внешнему виду, технико-экономическим характеристикам и т. п.

Ясно, что одна и та же техническая система может принадлежать одновременно к нескольким классам.

Ниже более подробно будут рассмотрены те принципы классификации технических систем, которые, с точки зрения проектировщика и конструктора, являются особо важными.

1. Классификация технических систем по функции

Названия технических систем часто выбираются в соответствии с их функцией. Составление номенклатур изделий применительно к требованиям сбыта, планирования, контроля, сравнительной оценки и т. п. также осуществляется, как правило, в соответствии с функцией технических систем. Было бы очень неплохо для взаимопонимания специалистов, если бы классификация и терминология во всех областях науки и техники совпадали.

Изделия обозначаются по функции также в тех случаях, когда требуется помочь потенциальному потребителю найти то или иное техническое средство для выполнения определенной функции: этому служат торговые и промышленные каталоги, обзорные таблицы и т. п. Узлы и детали машин часто также можно рассматривать как технические системы. Их классификацию целесообразно проводить по функции, так как конструктор, производственник и эксплуатационник применяют различные детали в соответствии с их функциональной пригодностью. Такую классификацию мы называем конструктивно-

функциональной; наряду с классификацией по способу изготовления она является основной при заимствовании существующих технических систем, унификации, типизации и стандартизации элементов и групп. Классификация по этим принципам позволяет экономить рабочее время конструктора.

Провести полную конструктивно-функциональную классификацию элементов и узлов, используемых в различных отраслях техники, чрезвычайно трудно в силу их многообразия. Тем не менее важно вести такого рода работу, опираясь по возможности на общие принципы. На любом предприятии используется множество элементов и узлов, выполняющих в различных отраслях техники одну определенную функцию, таких, как крепежные детали, редукторы, соединительные муфты, измерительные, регулирующие и сигнальные приборы, гидравлические и пневматические приборы и их части, специализированные электротехнические устройства и т. п. Если попытаться составить из всех этих разнообразных конструктивных элементов одну общую номенклатуру, она получилась бы необозримой.

При разработке классификации элементов, узлов и систем следует помнить о том, что условные обозначения не должны быть слишком сложными, в противном случае их будет трудно запоминать и расшифровывать. Рекомендуется использовать трех-, максимум четырехзначную маркировку. Классификация по конструктивно-функциональному принципу значительно упрощается, если обозначения технических систем и их элементов в чертежах давать в соответствии с их рабочей функцией (а не внешним видом); так, например, кольцевидные прокладки целесообразно называть «прокладка», а не «кольцо».

2. Классификация технических систем по принципу действия

Для конструктора важно, чтобы технические системы, выполняющие одинаковые функции, были далее сгруппированы по еще какому-либо важному признаку. Таким признаком можно считать принцип действия технической системы. Так, например, технические системы «промышленные печи» можно подразделить по принципу действия: печи электрические, газовые, печи на твердом и жидком топливах; электрические печи в свою очередь можно подразделить по используемому физическому принципу на электрические печи сопротивления (с прямым или косвенным нагревом) и индукционные электропечи. По характеру функционирования можно различать печи непрерывного действия и печи с прерываемым рабочим циклом

(например, камерные печи). Печи непрерывного действия далее можно классифицировать по способу транспортировки изделий: роликовые, полезные, подвозные печи и печи с шагающим подом.

Такого рода признаки технических систем относятся преимущественно к группе функционально обусловленных свойств, весьма характерных для технических систем и имеющих большое значение для методической работы конструктора.

3. Классификация технических систем по уровню сложности

Деление технических систем на классы по их структуре — обычное дело в работе конструктора. Основным признаком, по которому образуются классы, должна служить функция системы. Однако, учитывая потребности производства, например по соображениям монтажа, порой возникает необходимость в проведении иной классификации. Иногда в силу организационных причин технические системы целесообразно разделять на подсистемы.

Табл. 6.5 дает общее представление о классификации технических систем по уровню сложности, а на рис. 6.11 показан пример связей в технической системе, определяемых уровнями сложности составляющих ее частей.

Таблица 6.5.

Классификация технических систем по уровню сложности

Уровень сложности	Техническая система	Характеристика	Примеры
I	Конструктивный элемент Деталь машины	Элементарная система, изготовленная без монтажных операций	Болт, подшипниковая втулка, пружина, шайба
II	Подгруппа Группа Узел Механизм	Простая система, выполняющая несложную функцию	Коробка передач, гидравлический привод, шпиндельная бабка токарного станка
III	Машина Прибор Аппарат	Система, состоящая из групп и элементов и выполняющая определенную функцию	Токарный станок, автомобиль, электромотор
IV	Установка Предприятие Промышленный комплекс	Сложная система, состоящая из машин, групп и элементов, выполняющая ряд функций и характеризующая упорядоченные совокупности функций и места	Технологическая линия, цех термической обработки, нефтехимический комплекс

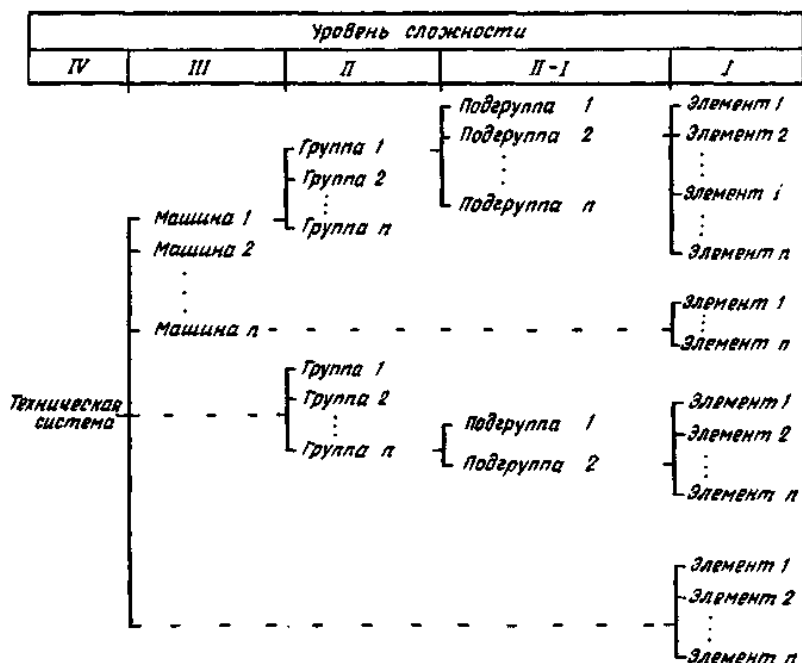


Рис. 6.11. Связи уровней сложности технической системы.

На более высоких уровнях сложности можно различать еще и промежуточные уровни. Тем не менее следует помнить, что речь идет об относительной иерархии. Одна и та же система более низкого уровня, например электромотор или коробка передач, в одной системе рассматривается как подгруппа, а в другой системе — как группа или машина (подсистема).

На практике общепризнано, что нижние уровни технических систем находят более универсальное применение; например, элемент «винт» применяется в машиностроении повсеместно, «электромотор» довольно часто, а «технологическая линия» используется лишь в определенных, специальных процессах.

Классификация технических систем по уровню сложности имеет немаловажное значение для конструктора, поскольку уровень сложности технической системы а) находится в определенном соотношении со степенью сложности решения поставленной перед конструктором задачи; б) предполагает установление известных

границ для специализации конструктора (например, инженер-проектировщик имеет дело с предприятием, инженер-конструктор — с машиной, конструктор деталей — с элементами машины); в) помогает конструктору ориентироваться в процессе работы, ибо, если он решает задачу на каком-то определенном уровне сложности, ему важно знать лишь то, как его задача согласована с более высоким уровнем (в отношении более низкого уровня конструктор принимает чаще всего только принципиальные решения).

На основании сборочного чертежа отдельные уровни сложности можно рассматривать так же, как совокупности процессов изготовления и монтажа. Образование соответствующих совокупностей, прежде всего из деталей, подгрупп и групп, является необходимым условием создания модульных конструкций, а также целесообразной организации производственного процесса.

4. Классификация технических систем по способу изготовления

Для изготовления определенных групп технических систем требуется однотипное технологическое оборудование. Например, на одном и том же оборудовании можно изготовить паровые котлы и химические емкости, на другом — токарные, фрезерные, сверлильные и другие станки. Детали машин можно также свести в технологические группы по принципу сходства технологических операций изготовления, где главным отличительным признаком будет служить форма.

Такая классификация позволяет рационально провести технологическую подготовку производства и повысить эффективность производственного процесса, поскольку дает возможность объединить рабочие места для изготовления одинаковых по способу изготовления деталей. Это в свою очередь облегчает осуществление самых различных мер рационализации, например специализацию рабочих цехов, предприятий. Значение такой классификации особенно велико при разработке и осуществлении планов подготовки производства, методов управления и планирования. Она является составной частью так называемой групповой технологии обработки. На рис. 6.12 показан пример возможной классификации деталей машин по способу изготовления.

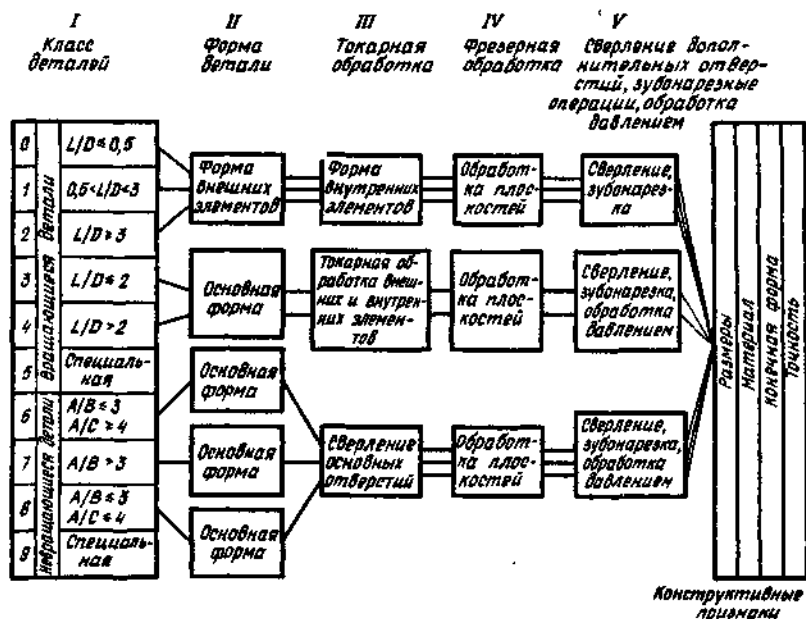


Рис. 6.12. Пример классификации деталей машин по способу изготовления.

5. Классификация технических систем по степени конструктивной сложности

Технические системы можно также классифицировать с точки зрения конструктивной сложности. В качестве примера в табл. 6.6 технические системы третьего уровня сложности (см. табл. 6.5) разделены на 6 категорий по степени их конструктивной сложности.

Таблица 6.6.

Примеры классификации технических систем III уровня сложности по степени конструктивной сложности

Степень кон- структивной сложности	Примеры
1	Небольшие резервуары, простые несущие конструкции, лотки, рас- пределители, консоли
2	Более сложные и крупные резервуары, вентили, простые муфты
3	Пресс-масленка, прицепной вагон, ванны с подогревом, дистилля- ционные аппараты, регуляторы
4	Специальные вентиляционные установки, печи для отжига, до- пастные насосы, шахтные подъемники
5	Газогенераторы, башенные краны, высоковольтные transforma- торы, поршневые насосы
6	Металлообрабатывающие автоматы, планетарные коробки передач, мощные генераторы электрического тока

В зависимости от уровня сложности рассматриваемой технической системы для решения связанных с ней проблем выбирается соответствующий специалист или группа специалистов. При планировании конструкторской работы степень конструктивной сложности разрабатываемой технической системы служит критерием для установки определенных временных рамок инженерной работы. Детали машин также можно классифицировать в зависимости от степени сложности их конструкции. Соответствующий пример классификации по этому принципу дан в табл. 6.7.

Таблица 6.7.

Примеры классификации деталей машин по степени конструктивной сложности

Степень кон- структивной сложности	Характеристика	Примеры
1	Очень простые детали с неболь- шим количеством контрольных размеров невысокой точности	Опорная шайба, простой рычаг, небольшой вал, болт, крепеж- ная скоба
2	Простые детали с большим коли- чеством контрольных размеров	Рычаг, шкив, простое штампо- ванное изделие
3	Более сложные детали	Шестерня, шлицевой вал
4	Более сложные детали с большим количеством контрольных разме- ров	Довольно сложные отливки, не- большие поковки
5	Очень сложные детали	Сложные отливки кожухов и поковки средних размеров
6	Очень сложные и большие детали	Каркасы, кожухи машин, свар- ные или литые станины
7	Особо сложные детали больших размеров и необычной формы с точным выдерживанием большого количества контрольных размеров	Лопасты турбины, большие по- ковки, прецизионные отливки сложной формы

Критериями оценки степени конструктивной сложности служат: степень оригинальности конструкции, сложность выполняемых функций, форм, структуры в целом; сложность расчетов; размеры, необходимые точность их выполнения и качество обработки; особые требования, предъявляемые к таким характеристикам, как масса, технологичность конструкции, затраты, требования к внешнему виду и т. п.

6. Классификация элементов технических систем по степени стандартизации и происхождению

Такая классификация очень важна для оценки экономичности конструкции. По степени стандартизации технической системы можно судить о целесообразности и возможных масштабах ее производства в рамках данного предприятия. На рис. 6.13 представлена структура групп и деталей некоторой технической системы.

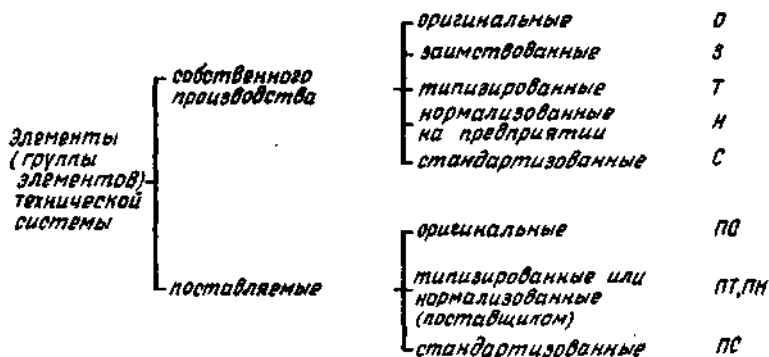


Рис. 6.13. Классификация групп и элементов по степени стандартизации и происхождению.

Если n обозначает количество всех конструктивных элементов (или групп) технических систем, а n с индексом — количество элементов в категориях, сформированных по степени стандартизации элементов, то

$$n = n_0 + n_3 + n_T + n_H + n_C + n_{ПО} + n_{ПН} + n_{ПС}.$$

Следующая формула выражает соотношение долей элементов отдельных категорий:

$$n_0/n + n_3/n + n_T/n + n_H/n + n_C/n + n_{ПО}/n + n_{ПН}/n + n_{ПС}/n = 1.$$

С экономической точки зрения n_o и $n_{по}$ должны быть как можно меньшими, поскольку они характеризуют требования, предъявляемые к конструкторской и технологической подготовке производства. При минимальных значениях n_o и $n_{по}$ благоприятны условия для организации серийного или даже массового производства.

Часто, впрочем, в силу каких-либо иных причин эти соображения не являются решающими. И все же в каждый отчет о проделанной конструкторской работе следует включать данные о соотношении отдельных категорий элементов различной степени стандартизации.

7. Классификация технических систем по степени оригинальности конструкции

При разработке новой машины конструктор всегда старается использовать в конструкции оправдавшие себя на практике узлы и детали. По степени оригинальности конструкции технические системы можно разделить на следующие категории.

Заемствованные технические системы. Для выполнения необходимой функции уже существуют какая-либо техническая система или даже несколько систем, из которых могут быть выбраны наиболее подходящие. К ним относятся в первую очередь унифицированные элементы и группы (болты, клинья, вентили, пружины), а также неунифицированные элементы и группы, которые могут быть заимствованы из других конструкций.

Доработанные технические системы. В наличии имеется какая-либо техническая система, выполняющая необходимую функцию, но не отвечающая некоторым требованиям. Возникает потребность, например, изменить габариты, мощность, число оборотов, скорость, установочные размеры, материал или технологию. Структуры системы и важнейшие свойства элементов в этом случае остаются без изменения. Таким образом, доработка технической системы проводится исключительно в целях приспособления ее к особым условиям и требованиям новой задачи, а новые материалы используются только в целях повышения качества, удешевления или модернизации.

Модифицированные технические системы. Существующие системы не отвечают требованиям, предъявляемым к некоторым свойствам групп и элементов конструкции. В модифицированной конструкции обычно не изменяются лишь функция, некоторые параметры и по возможности принцип действия. В элементах могут быть изменены форма, размеры, материал или технология, в сложных технических системах изменяются органоструктура и конструктивная схема, т. е. некоторые

элементы и группы, их соединение и размещение в пространстве. Обычно модификация осуществляется путем переделки конструкции. *Новые технические системы.* Для выполнения желаемой функции отсутствует техническая система или же существующая имеет недостатки принципиального характера. Необходима система с новым принципом действия и другими техническими свойствами. Например, эффект «нагревание», который раньше обеспечивался технической системой с сжиганием нефти, теперь должен обеспечиваться новой системой с резистивным нагревом.

8. Классификация технических систем по типу производства

Тип производства, который определяется количеством изготавливаемых единиц продукции, придает каждому изделию ряд характерных технических и экономических свойств.

Технические системы единичного производства. В этом случае конструкторские и подготовительные работы необходимо приспособить к нуждам поштучного производства, в условиях которого стоимость каждой изготовленной технической системы увеличивается. Не исключено, что в условиях единичного производства необходимая функция технической системы вообще не будет достигнута, поскольку при изготовлении крупных технических систем приходится работать без прототипа. Вот почему эта категория систем предъявляет высокие требования к конструктору.

Технические системы серийного или массового производства. Эти системы в целом лучше проработаны с точки зрения производства. Из-за большого объема партий изделий доля конструкторских затрат по отношению к общим расходам невелика. Однако поскольку контролю подвергается, как правило, лишь небольшая часть изделий, то не исключены различные погрешности и дефекты. Только при осуществлении непрерывного контроля за всеми операциями или выпускаемыми деталями и изделиями в целом можно добиться стабильного качества при серийном и массовом производствах. Специалисту упомянутые категории систем интересны и в том плане, что они формируют основу для определения возможного качества изделий.

Прослеживается четкая тенденция ко все большему использованию унифицированных, серийно выпускаемых технических систем, особенно для выполнения различных функций низких уровней, например элементов соединения, измерения, регулирования, привода, распределения. С другой стороны, возрастает количество технических систем специального назначения. Современное производство не может

обойтись без целого ряда вспомогательных средств, специализированных машин, автоматов и поточных линий, специального оборудования, т. е. без всего того, что обеспечивает выпуск дешевой унифицированной продукции в массовом количестве. Обе категории изделий предъявляют высокие требования к объему и качеству конструкторской работы.

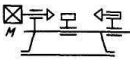
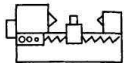
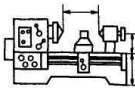
9. Классификация технических систем по степени абстрактности

Технические системы образуют одно из самых обширных множеств объектов окружающего нас мира; естественно, они могут быть подвергнуты классификации по иерархическому принципу. Для более подробной иерархической классификации могут быть использованы обозначения системных категорий по степени родства и по морфологическим признакам, а именно: класс, группа, семейство, тип, род, вид и т. д. Проведенное исследование структур (разд. 6.3—6.6) может дать этой классификации важную для нас направленность. Концентрируя внимание на классе машинных систем, мы получим иерархическую структуру, пример которой дан в табл. 6.8.

Таблица 6.8.

Степени абстрактности машинных систем и их представление (для примера типа «станок»)

Степень абстрактности	Общая иерархия			Примеры	
	Иерархический уровень	Графическое представление	Установленные конструктивные признаки	Название	Графическое представление
1,0	Класс «машинная система» (MS)		Преимущественно механический способ действия	Станок	
0,8	Группа		Группа операторов Класс преобразований	Металлообрабатывающий станок	
0,6	Семейство	«Черный ящик», схема технологии, основная функциональная структура	Семейство операторов, технологический принцип преобразований, требуемые воздействия и основные механические функции	Металлообрабатывающий токарный станок	

0,4	Тип	Детальная функциональная структура, общая конструктивная схема, предварительный эскиз конструкции	Тип операндов, функциональная структура, входы, семейства носителей функции и их предварительное упорядочение	Универсальный металлообрабатывающий токарный станок	
0,2	Род	Чертежи конструктивной схемы, эскизы узлов	Комплект элементов, их упорядочение, предварительное определение формы и некоторых размеров, виды материалов, некоторые указания о допусках и качестве поверхности	Токарный станок SUR	
0	Вид (типоразмер)	Полный комплект технической документации	Полные и точные данные о элементах и их упорядочении. Для всех элементов: форма, размеры, материал, технологический метод, допуски, качество поверхности	Токарный станок SUR5	

Здесь обозначение «станок» выступает в качестве типа названия группы машинных систем. При этом, как видно из рисунка, в определении степени абстрактности главную роль играют конструктивные признаки.

10. Классификация технических систем по типу операнда

Технические системы можно также классифицировать, как это уже было показано на примере процессов (табл.3.6), по известным типам объектов действия (операндов): материя, энергия, информация, биологические объекты.

Ранее применение этого принципа классификации привело к разделению машинных систем на машины для переработки материи (доменные печи, химические реакторы), машины для переработки энергии (двигатели, моторы и т.п.). Однако при нашем рассмотрении технической системы как системы, осуществляющей действие, такое деление окажется вторичным, а в некоторых случаях даже ошибочным, поскольку потенциальное применение данной технической системы может не ограничиваться названным типом операнда.

11. Классификация технических систем по их месту в техническом процессе

При рассмотрении системы преобразований и ее элементов уже упоминалось, что не все технические системы могут непосредственно использоваться в технологическом процессе в качестве исполнителей

действия. Эта классификация имеет мало общего с прочими классификационными принципами, особенно с принципом сложности, и подразделяет технические системы лишь на а) оконечные и б) внутренние (промежуточные).

Оконечные системы, выступающие в технологическом процессе в роли операторов, — это преимущественно установки, машины, аппараты, приборы и инструменты, осуществляющие действие. Они образуют с операндом пару взаимодействия. В отличие от них внутренние (промежуточные) технические системы, как например, редуктор, привод, трансформатор, только при совместной работе с оконечными техническими системами могут осуществлять необходимое воздействие на операнд.

12. Классификация технических систем по степени ценности

Критерии качества (потребительской ценности) рассматриваются в гл. 7. Классы качества, например, товаров широкого потребления точно установлены. Однако для технических систем в силу их разнообразия и специального назначения такая классификация довольно условна.

Материалы этой главы можно подытожить выводом:

Классификацию технических систем можно проводить с различных точек зрения; при этом из всего множества технических систем образуются подмножества, связанные общими отличительными признаками. Полученные категории могут служить различным целям, например систематизации, наглядности, каталогизации, оценке, анализу и т. п.

7. Свойства технических систем

При обсуждении технических систем часто приходится задавать вопросы, начинающиеся со слов типа «каков» («какой»), например: Каков автомобиль? Какая у него скорость? Какой у него расход горючего? Какова его надежность? Какой он внешне? Какого он цвета? Ответ на некоторые из этих вопросов гласит: автомобиль имеет максимальную скорость 150 км/ч, расход горючего 10 л / 100 км. Все, кто привык к количественным оценкам, согласятся с этим. Однако тот же ответ мог бы прозвучать и так: автомобиль идёт быстро; у него малый расход горючего. Такая характеристика уже не однозначна; она относительна, поскольку неизвестно, какая скорость понимается под словом «быстро»: то ли 20, то ли 200 км/ч. Чтобы избежать

неопределенности, мы можем вопрос уточнить: Сколько километров в час проезжает автомобиль? Вопрос, начинающийся со слова «сколько», очень важен, так как, зная ответ, мы можем дать количественную оценку рассматриваемому свойству, т. е. определить его величину.

Труднее ответить на вопрос: Какова надежность этого автомобиля? Ответ может быть следующим: он надежен или ненадежен. Это, естественно, относительная характеристика, зависящая от того, какими соображениями руководствуется тот, кто оценивает. Человек, у которого автомобиль в течение 6 месяцев отказал всего три раза, будет считать его более надежным, чем тот автомобиль, который он прежде ремонтировал каждую неделю. Ясно, что можно было бы сформулировать определение надежности и тем самым получить количественную оценку этого свойства, т. е. можно было бы применительно к надежности ответить на вопрос «сколько?».

Еще труднее ответить на вопрос: Каков автомобиль внешне? В ответ мы услышим: он красивый, некрасивый, элегантный и т. п. Это очень неопределенная оценка; надо признать, что не так-то просто определить количественно этот параметр, изменяющийся во времени и обычно определяемый субъективно. Такая оценка стала бы более определенной, если бы она делалась путем сравнения с другим автомобилем. При этом оценка внешнего облика другого автомобиля приобрела бы значение объективного критерия.

Определенную трудность представляет выбор свойства для характеристики той или иной технической системы. О том же самом объекте — автомобиле — можно, например, спросить: Какой у двигателя коэффициент сжатия? Какова его скорость вращения? У него передний привод или задний? По-видимому, подобные вопросы могут возникнуть скорее всего в разговоре специалистов, так как упомянутые параметры определены для них; им известно, хорошо это или плохо, если, например, коэффициент сжатия составляет 10:1, скорость вращения двигателя равна 7000 об/мин, а привод у него передний. Эти примеры лишь в общих чертах обрисовывают проблематику свойств технической системы и их оценки.

Заметим, кстати, что мы уже определяли *свойство* как *признак объекта* (в нашем случае — технической системы). Признаком могут служить, к примеру, способность вести себя определенным образом или удовлетворять какому-либо требованию. Через те или иные свойства дается характеристика технической системы. При этом для объективного анализа важно, чтобы оцениваемые свойства и критерии оценки могли быть определены количественно.

Не исключено, что широкое толкование понятия «свойство», включающее функцию, производительность, размеры, удобство обслуживания, транспортабельность, технологичность и т. п., вначале будет представлять известные трудности для инженера; однако в процессе более глубокого изучения и использования такое толкование этого понятия становится естественным и привычным.

Далее мы должны себе твердо уяснить, что самое важное, чем можно охарактеризовать техническую систему, — это наличие у нее желаемых свойств, поскольку техническая система является лишь исполнителем необходимой рабочей функции и соответственно предполагаемого поведения.

Во взаимосвязи с функцией находится целый ряд других свойств: удобство эксплуатации и обслуживания, ремонтпригодность и т. д. Наряду с этим имеется множество других свойств, которыми должна обладать техническая система (например, определенные прочность, габариты, форма), чтобы обеспечивалось желаемое функционирование. Заказывая техническую систему, мы определяем не только то, что эта система должна делать, мы одновременно устанавливаем ее желаемые свойства. На этапе постановки задачи будем называть желаемые свойства технической системы *требованиями*.

Таким образом, чтобы отвечать предъявляемым требованиям, техническая система должна не только выполнять желаемую рабочую функцию, но и обладать в известной мере определенными свойствами. Техническая система всегда является носителем самых различных свойств, но только мера этих свойств (ценность) имеет решающее значение. Совокупная ценность системы, обладающей многочисленными и разнообразными свойствами, определяется сочетанием оценок всех этих свойств.

В последующих разделах для облегчения работы по составлению перечня требований при постановке задачи или при проведении сравнительного, оценочного или контрольного анализов мы постараемся классифицировать все свойства по наиболее важным аспектам. Уже здесь необходимо подчеркнуть важность корректной постановки задачи по конструированию технической системы. Далее конструктору необходимо понять отношения между свойствами, так как сущность процесса конструирования заключается в выяснении и формировании желаемых свойств системы и отношений между ними.

7.1. Общие признаки технических систем

Технические системы характеризуются рядом признаков. Первым признаком следует считать уже упоминавшееся *многообразие*

элементов системы, не имеющее аналогии в других областях. Номенклатура в машиностроении, без учета типовых вариантов, включает в себя десятки тысяч различных деталей технических систем. Наряду с многообразием функций, выполняемых этой номенклатурой деталей, они имеют также множество других свойств. Хотя различные технические системы функционируют по-разному, целью функционирования всегда остается осуществление преобразований; в нормальных рабочих условиях оно детерминировано и им можно управлять. В случае повреждения системы ее функционирование нарушается. Значительную часть технических систем составляют сложные и очень сложные системы, состоящие из тысяч или даже десятков тысяч элементов. Естественным следствием такой сложности является высокая стоимость таких технических систем. К ним прежде всего относится дорогостоящее оборудование, изготовленное в условиях единичного производства. Основным материалом для изготовления технических систем по-прежнему служит металл, преимущественно сталь, хотя все более возрастает применение полимеров.

7.2. Категории свойств технических систем

Все многочисленные и разнообразные свойства технических систем можно классифицировать по различным категориям.

Классификация свойств по способу их установления

Это а) внешние и б) внутренние свойства.

Одни свойства технических систем легко можно установить либо посредством органов чувств, либо с помощью различных вспомогательных устройств. Назовем эти свойства внешними. К ним относятся, например, форма, размеры, цвет. Другие свойства внешне никак не проявляются, и для их установления приходится прибегать к специальным мерам. Такие свойства мы назовем внутренними.

Однако различение свойств технических систем только с таких позиций привело бы к неточным и относительным результатам. Для более точного определения указанных свойств необходимо исходить из системного принципа; тогда внешние свойства превратятся в отношения системы к ее окружению. Внутренние свойства предстанут в качестве отношений между элементами системы и свойствами элементов. Отношения на более низких уровнях мы принимать во

внимание не будем; это свойства отдельных элементов, например свойства материала или электрические свойства, которые определяются структурой элемента. Пользователя технической системы интересуют главным образом внешние свойства.

Классификация свойств по причинной связи

В соответствии с этой классификацией различают а) входные воздействия (причина) и б) функции (следствие). Между входными воздействиями и функциями существуют причинные отношения. Причинность как важная категория отношений уже была рассмотрена ранее. Способность конструктора тонко чувствовать различие между причиной и его следствием следует отнести к числу его главных творческих способностей. Ему приходится «конструировать» причины для получения требуемых следствий. Приведем примеры причинных отношений в технических системах: высокая температура свечи накаливания является причиной воспламенения сжатой топливной смеси; недостаточная жесткость станины токарного станка является причиной погрешности обработки деталей. Временная последовательность причины и действия проявляется в виде процесса.

Классификация свойств по функциональной зависимости

По функциональной зависимости свойства подразделяются на а) зависимо изменяющиеся и б) независимо изменяющиеся свойства. В областях естествознания и техники существует много формул, выражающих взаимозависимости свойств друг от друга. Так, например, существует зависимость между давлением и уровнем жидкости; прочностью и температурой материала; скоростью движения объекта и его кинетической энергией; скоростью потока и диаметром трубы. Все эти свойства находятся между собой в функциональной зависимости, которая может быть выражена аналитически. Те свойства, на которые оказывается влияние, называются зависимыми переменными (прочность). Те свойства, которые оказывают влияние на другие, называются независимыми переменными (материал, форма, габариты). Одни и те же свойства в различных технических системах могут выступать как зависимые или независимые.

Классификация свойств по возможности их количественного определения

По возможности их количественного определения свойства технической системы могут быть подразделены на а) определяемые легко; б) определяемые с трудом; в) не определяемые количественно. Примеры, приведенные в начале этой главы, могут служить иллюстрацией этих категорий. В тех случаях, когда невозможно оценить количественно свойства, применяется балльная система оценки, т. е. с использованием определенных критериев проводится оценка свойств по возрастающим уровням (классам). Такой метод, бесспорно, дает более точные результаты, чем некая субъективная, большей частью продиктованная эмоциями «оценка», заканчивающаяся обычно выводом «хорошо» или «плохо». Однако и здесь возникает множество неопределенностей, обусловленных, с одной стороны, трудностью определения «цены» балла, а с другой — субъективностью эксперта, дающего оценку. К этому вопросу мы еще вернемся .

Классификация свойств по их значимости

В этом отношении свойства подразделяются на а) очень важные (незаменимые), например функция, безопасность; б) важные, например надежность, срок службы, цена; в) менее важные, например отсутствие необходимости в упаковке, возможность длительного хранения; г) не существенные для функционирования технической системы свойства, например внешний вид, цвет.

Не все свойства технической системы легко определить с точки зрения их значимости. Кроме того, необходимо учитывать относительность этой значимости, поскольку значимость тех или иных свойств зависит от обстоятельств. Порой важнейшим свойством оказывается быстрота поставки системы или легкость ее демонтажа.

Масса, не имеющая существенного значения для, например, станка, является важнейшим параметром летательного аппарата. Поэтому значимость различных свойств нужно тщательно анализировать в каждом конкретном случае.

Классификация свойств по их физической сущности

Свойства можно классифицировать по их физической сущности следующим образом:

- а) геометрические свойства, например ширина, высота, симметрия, форма, межосевое расстояние, угол;
- б) кинематические свойства, например скорость, ускорение;

- в) механические свойства, например прочность, упругость, прогиб, герметичность;
- г) тепловые свойства, например нагрев, теплопроводность, тепловые потери, теплоизоляция;
- д) электрические и магнитные свойства, например емкость, напряжение, сопротивление, электрическая проводимость;
- е) оптические свойства, например фокусное расстояние, преломление, отражение, поляризация;
- ж) акустические свойства, например поглощающая способность, эхо, шум, звуковая частота;
- з) химические свойства, например химическая активность, концентрация, коррозия, химическое средство.

Классификация свойств по их физической сущности соответствует структуре технических наук. Приведенный перечень категорий свойств наглядно иллюстрирует пестроту и многообразие свойств технических систем.

Прежде чем перейти к следующим способам классификации, посмотрим, помог ли описанный подход достичь поставленной цели — облегчить работу конструктора. Наиболее существенное, что можно выделить в приведенной до сих пор классификации свойств, — это **раскрытие отношений между свойствами**; прежде всего это касается первых трех способов классификации. Классификация свойств по их физической сущности демонстрирует, кроме того, многообразие свойств технической системы.

Разумеется, ни одного из до сих пор приведенных способов классификации недостаточно для полной характеристики свойств, что необходимо для методической работы конструктора. Нельзя забывать, что такая характеристика должна не только быть применимой к свойствам любых технических систем (от детали до предприятия), но и учитывать технический, экономический, эргономический, манипуляционный, социальный, юридический и другие аспекты. Речь, таким образом, идет об очень сложной задаче, до настоящего времени полностью еще не решенной. В книге Кессельринга «Анализ конструкций» , а также в нормалях VDI приводится следующий перечень групп возможных свойств:

- 1) количественные (параметры); 2) геометрические; 3) механические;
- 4) тепловые; 5) электрические и магнитные; 6) оптические; 7) акустические; 8) химические; 9) производственные и монтажные;
- 10) эксплуатационные.

Группы 2—8 соответствуют областям знаний, а группы 1, 9 и 10 сформированы по другим принципам. Хотя этот перечень охватывает

далеко не все свойства, уже можно говорить о широком спектре свойств технических систем.

Классификация свойств по потребности в конструкторской работе

Примерный перечень соответствующих категорий свойств дан в табл. 7.1.

Таблица 7.1

Классификация свойств технических систем по потребности в конструкторской работе

Категория свойств	Вопросы	Примеры свойств
Функция F	Каковы функции системы?	Рабочая функция
Действие W	Что делает система?	Вспомогательная функция Подготовительная функция Функция управления и регулирования Функция согласования
Функционально обусловленные свойства Vd	Какие условия характерны для данной функции?	Производительность Скорость Размеры Масса

Категория свойств	Вопросы	Примеры свойств
Производственные свойства <i>Be</i>	Насколько пригодна система для производственного процесса?	Функциональные параметры Грузоподъемность Безопасность эксплуатации Надежность Срок службы Расход энергии Занимаемая площадь Ремонтопригодность
Эргономические свойства <i>Erg</i>	Как обслуживается система и какое влияние оказывает на человека?	Удобство обслуживания Способы обслуживания Виды помех Требования к оператору
Эстетические свойства <i>Au</i>	Как система воспринимается эстетически?	Форма Цвет Удобство Привлекательность
Манипуляционные свойства <i>Di</i>	Насколько пригодна система для целей транспортировки, хранения, упаковки?	Соответствует условиям транспортировки Соответствует условиям хранения Соответствует условиям упаковки Пригодна для непосредственного пуска в ход
Характеристики поставок и планирования <i>LP</i>	Когда система может быть поставлена? В каком количестве изготавливается?	Доступность Продукция серийного производства Продукция единичного производства
Правовые нормы <i>GN</i>	Соответствует ли система правовым юридическим нормам и положениям?	Соответствует юридическим нормам Нарушение патентного права Соответствует требованиям коллективного договора
Технологические свойства <i>Fe</i>	Насколько соответствует система принятой технологии?	Соответствует технологии Соответствует условиям монтажа
Экономические свойства <i>We</i>	Насколько экономичен процесс производства?	Эксплуатационные расходы Издержки производства Экономическая эффективность Цена
Конструктивные свойства <i>Ko</i>	Как реализуются внешние свойства?	Структура Форма Габариты Материал Качество поверхности Поле допуска Тип производства
Качество изготовления <i>He</i>	Кто и как изготовил систему?	Изготовитель Достоинства и недостатки изготовления

Представленные в таблице вопросы призваны помочь выявлению содержания категории. Приведенные примеры должны облегчить понимание. Ниже отдельные категории свойств будут рассмотрены подробнее.

Функции F и действия W . Основное (рабочее) и вспомогательные действия, необходимые для выполнения преобразований, осуществляются путем взаимодействия человека и технической системы. Техническая система TS выполняет большую часть таких действий. Человек оставляет за собой лишь некоторые действия по управлению системой. Техническая система, таким образом, должна выполнять необходимые функции, т. е. быть способной к заданным действиям. Функция является важнейшим свойством любой технической системы, ибо даже если все прочие свойства наличествуют, а рабочая функция не выполняется, т. е. если автомобиль не едет, токарный станок не обрабатывает, то такая техническая система бесполезна; по крайней мере она могла бы служить какой-нибудь иной цели, например в качестве украшения или выставочного экспоната.

Следует напомнить, что под «функцией» мы подразумеваем свойство, абстрагированное от определенных условий данной ситуации.

Например, автомобили, самолеты, суда, ракеты, ленточные транспортеры, краны, подъемники выполняют функцию транспортировки. Однако только автомобили обладают функцией безрельсовой транспортировки по суше; легковые автомобили имеют функцию безрельсовой транспортировки людей по суше; гоночные автомобили — функцию безрельсовой транспортировки людей по суше с максимальной скоростью в спортивных целях и т. д. На примере функции «транспортировка» в соответствии с количеством условий, налагаемых на нее, показаны различные уровни абстрактности данной функции (рис. 7.1).

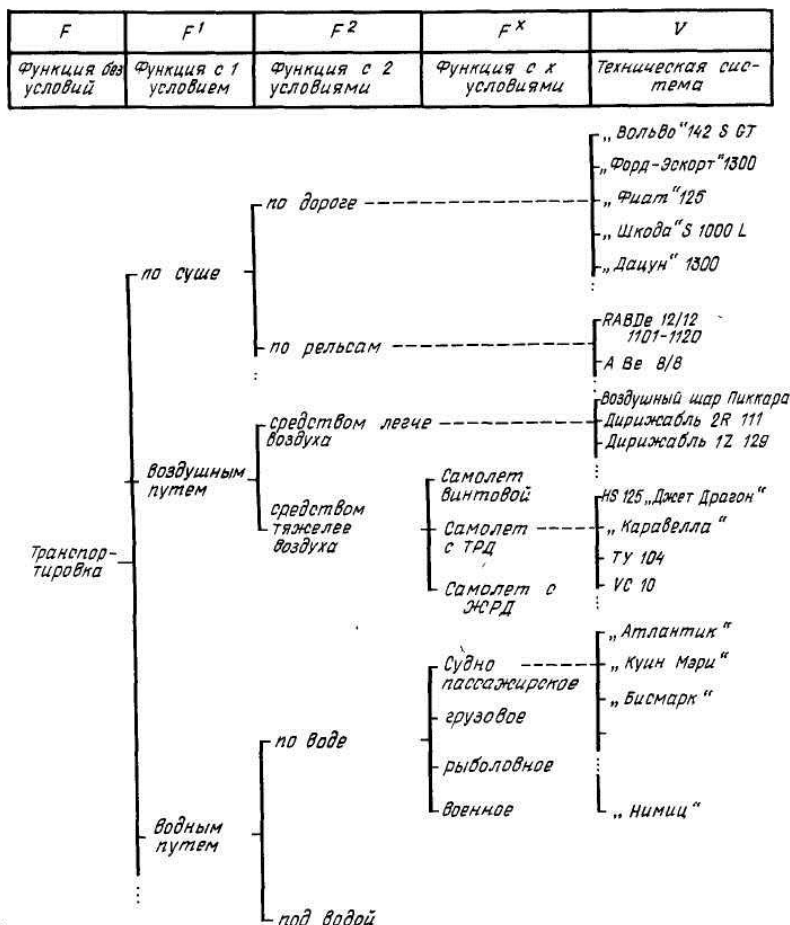


Рис. 7.1. Пример различных уровней абстрактности функции «транспортировка».

Анализируя этот пример, мы можем без труда сделать вывод о том, что функционирование какой-либо технической системы представляет собой реализацию функции, соответствующую условиям конкретной ситуации. Таким образом, теперь уже речь идет не об обширном классе технических систем, отличающимся определенным типом функционирования, а о некоем типе систем, имеющем вариации во второстепенных свойствах. Значение этого вывода отчетливо проявится при рассмотрении методологии конструирования.

Функции технической системы отличаются разнообразием и сложностью. Рассмотрим некоторые функции на примере производства автомобиля: изготовление коленчатого вала, испытания двигателя, общая сборка автомобиля. Все эти функции осуществляются на относительно высоком уровне абстрактности, причем одни функции более сложные, чем другие. Степень сложности возрастает от элементарной функции обточки вала до общей сборки автомобиля. Этот пример показывает наличие определенной иерархии функций. Таким образом, для производства автомобиля необходим целый ряд функций (испытания двигателя, общая сборка автомобиля), состоящих из нескольких более простых функций (испытание системы зажигания, сборка шасси), которые в свою очередь делятся на подфункции вплоть до элементарных (обточка вала, завинчивание гаек). Функция любого уровня выполняется с помощью ряда подфункций, зависящих от вида применяемой технологии.

В качестве элементарных функций удобно использовать объединительно-разъединительные функции, которые характеризуются высоким уровнем абстрактности и соответствуют ряду иерархических ступеней для различных наборов условий (рис. 7.2).

большее число функций. При этом говорят об агрегировании функций. Такая взаимосвязь между множеством функций и множеством технических систем на определенном уровне схематично показана на рис. 7.3. Возможность универсального применения какой-либо технической системы уменьшается по мере усложнения функции, которую требуется выполнить.

Как уже говорилось, для осуществления рабочего действия необходимы вспомогательные действия. В соответствии с этим функции можно разделить на рабочие (основные), подготовительные, управления и согласования.

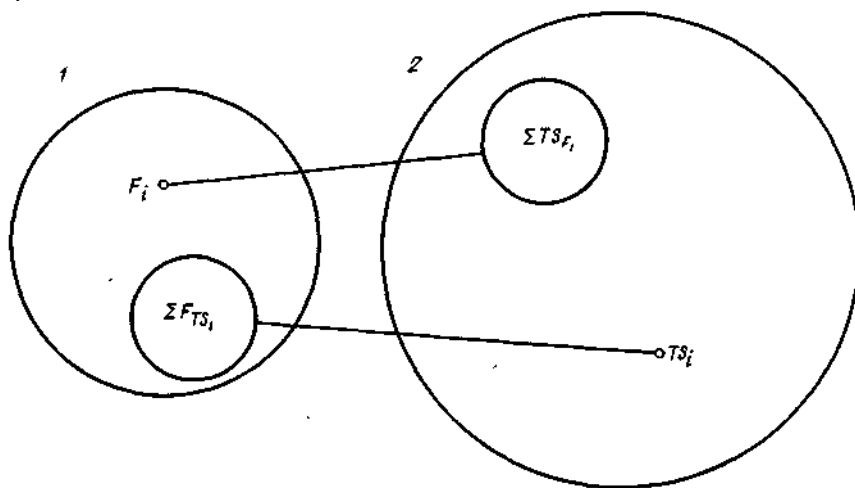


Рис. 7.3. Отношения между функциями F и техническими системами TS

1 — множество технических систем ΣTS , способных выполнять функции ΣF , 2 — множество функций ΣF , выполняемых техническими системами ΣTS , F_i — функция, TSF_i — техническая система, выполняющая функцию F_i , TS_i — техническая система, FTS_i — функция, выполняемая технической системой TS .

Однако необходимо учитывать всю относительность этих категорий. Поскольку такая классификация касается технических систем всех уровней сложности, она действительна как для больших систем, так и для их подсистем, групп и элементов. Как правило, вспомогательная функция большой системы осуществляется через подсистему, для которой эта функция является основной. Так, например, при выполнении задачи изготовления детали вспомогательная функция зажима заготовки на токарном станке реализуется с помощью

зажимного патрона. Но при постановке задачи зажима патрона функция «зажим» будет определяться уже как основная (рабочая) функция.

Функционально обусловленные свойства *Bd*. При рассмотрении функции транспортировки в предыдущем разделе мы остановились на транспортировке людей по суше с некоторой скоростью. Здесь мы уже имеем три условия, которые допускают множество других решений. Например, в соответствии с поставленной задачей (транспортировка людей по суше) можно было бы также использовать автобус или велосипед. Поэтому для постановки задачи создания конкретной технической системы мы должны указать дополнительные условия — признаки. В нашем случае это, скажем, количество людей, расстояние, качество дороги и т. п. Таковы условия для примера с функцией транспортировки. Аналогичным образом специалист определяет признаки и для других функций. Пользуясь методом анализа, мы можем констатировать, что данные о преобразовании, т. е. об объекте, технологии, условиях преобразования и т. п., представляют собой функционально обусловленные свойства. Эти важные свойства тесно связаны с функцией и соответственно с действием (функционированием) и всегда приводятся в каталогах и инструкциях, так как только посредством этих свойств объясняется назначение технической системы. Обычно они называются **параметрами технической системы**. С методической точки зрения функционально обусловленные свойства служат своего рода «ситом»: посредством конкретизации и систематизации условий мы сужаем рамки многообразия — множества средств — до тех пор, пока в нашем распоряжении не останется небольшое количество средств, выполняющих заданные условия. Эти отобранные средства (системы) затем анализируются и оптимизируются сообразно с другими свойствами.

Производственные свойства *Be*. Желаемое преобразование операнда осуществляется посредством действия технической системы. Пригодность технической системы для использования характеризуется группой свойств, которые мы будем называть производственными, например надежность, безопасность, срок службы, расход энергии, материалов, занимаемая площадь, ремонтпригодность, возможность регулировки и т. д. Поскольку выполнение системой рабочей функции фактически является предварительным условием, значение производственных свойств все более возрастает. Требования к производственным свойствам с течением времени изменяются; они различны в зависимости от отрасли техники. Там, где в роли оператора процесса или операнда выступает человек, должны

предъявляться высокие требования к надежности и безопасности. В процессе ускоренного развития техники изменяются представления о наиболее подходящем сроке службы какой-либо системы; так, часто уже неприемлемо требование, что какая-либо деталь должна иметь как можно более длительный срок службы, поскольку многие машины устаревают быстрее, чем изнашиваются. Другая тенденция наблюдается в отношении к габаритам и массе. Имеющиеся в распоряжении помещения становятся все меньше и обходятся все дороже. Поэтому уменьшение размеров и массы технической системы выступает в качестве все более важного требования.

В процессе эксплуатации техническая система обычно является источником шума, тепла, вибрации; она загрязняет воздух, вырабатывает вредные отходы, от которых нужно избавляться. С распространением технических систем эти так называемые побочные выходы начинают принимать катастрофические размеры и превращаться в угрозу человечеству. До настоящего времени последствия этих свойств технических систем часто игнорировались, поскольку преобладали экономические соображения. Однако если мы хотим «пережить» созданные нами машины, то действительно необходимо, чтобы эти требования учитывались соответствующим образом уже на этапах проектирования и конструирования.

Выявление производственных свойств технической системы вызывает определенные трудности прежде всего на стадиях эскизного и технического проектирования, так как эти свойства обычно проявляются во всей своей полноте лишь после продолжительного периода эксплуатации системы. Поэтому конструктору следует в течение длительного времени контролировать работу созданных им технических систем и анализировать причины возникновения неудовлетворительных производственных свойств. Лишь этот источник информации наряду с изучением производственных свойств других аналогичных машин может обеспечить конструктору необходимую степень понимания этих свойств.

Эргономические свойства *Erg*. Преобразования осуществляются, как указывалось выше, взаимодействием пары человек — техническая система. Получаемый результат зависит от слаженности совместной работы этих участников технологического процесса. Общество также заинтересовано в том, чтобы человеку при этом не наносилось никакого вреда. Появилась самостоятельная научная дисциплина — эргономика, которая изучает отношения в системе человек — машина. Практическим аспектом этих отношений занимаются прикладная эргономика и психотехника, опирающиеся на антропологию, физиологию, психологию, биологию, социологию, физику и

кибернетику. Задачи исследования отношений в системе человек — машина включают изучение влияния времени и движения в производственном процессе на человека, восприятие органами чувств, обработку данных и принятие решений, рабочие, эффективные и предельные нагрузки (время, усилие, температура, давление, ускорение, излучение, голод), нарушение нормальной ориентации (акустической, оптической и тактильной), адекватность реакций, адаптацию и т. п.

Эргономика не ограничивается техникой безопасности и условиями работы; она также ищет пределы человеческих способностей в экстремальных ситуациях. Разумеется, ряд задач эргономики существен только для определенных отраслей техники, например авиации и космонавтики. Однако существует множество общих вопросов, например проектирование технических систем с таким расчетом, чтобы они обеспечивали человеку со средними способностями оптимальные условия обслуживания и управления. В идеале работа не должна утомлять ни физически, ни психически. Конструктор ставит перед человеком в системе человек — машина определенные задачи; следовательно, ему должны быть хорошо известны все достоинства и недостатки обоих партнеров системы (табл. 7.2).

Таблица 7.2.

Характерные свойства человека и машины

Человек	Машина (техническая система)
Достоинства 1. Способность принимать важные решения даже при наличии ограниченной информации 2. Правильные реакции даже в неожиданных условиях 3. Способность выполнять определенные операции различными способами, что важно в том случае, если какой-то механизм поврежден 4. Гибкость в планировании работ Недостатки 1. Ограниченная способность к восприятию информации, в том числе ограниченная скорость восприятия 2. Ограниченные способности органов чувств (рецепторов) 3. Снижение производительности и внимания с течением времени 4. Необходимость комфортных условий (температуры, влажности, давления, шума и т. п.) 5. Относительная замедленность мыслительных операций и высокая вероятность ошибки	Достоинства 1. Изменяемая по желанию способность к восприятию информации 2. Возможность улучшения параметров восприятия 3. Постоянная производительность и точность во времени 4. Возможность приспособления к любым окружающим условиям 5. Быстрое выполнение логических операций почти без ошибок Недостатки 1. Функционирование лишь в соответствии с полученными командами 2. Возможность отказа в непредвиденных условиях 3. Ограниченность количества выполняемых операций 4. Трудоемкость изменения программы функционирования

Отношения в этой системе схематично показаны на рис. 7.4.

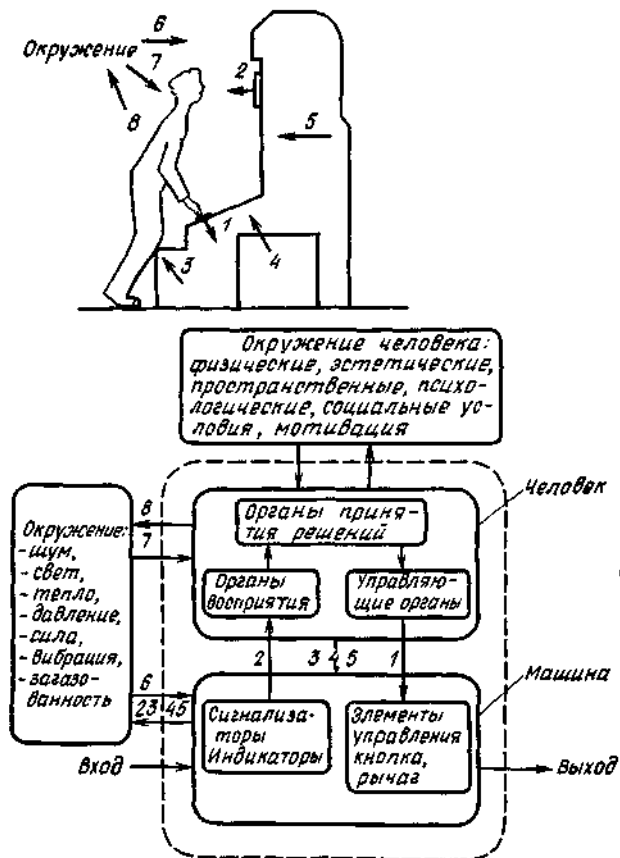


Рис. 7.4. Отношения в системе человек—машина

1—обслуживание и управление, 2 — сигналы о функционировании, 3—физические препятствия, 4 — шумы, 5 — вибрация, тепло, 6 — влияние окружения на машину (техническую систему), 7 — влияние окружения на человека, 8 — влияние человека на окружение.

Конкретные выводы, которые конструктор может сделать на основе результатов эргономических исследований, могут быть представлены, например, в следующем виде:

— возможности различных частей тела человека применительно к производственному процессу;

- размеры человека (антропометрия);
- производительность человека в различных окружающих условиях;
- расположение измерительных приборов и органов управления с расчетом достижения оптимальной производительности системы человек — машина.

Эстетические свойства *Аи*. Тенденция к объединению свойств практической полезности технической системы с эстетическими свойствами (т. е. объединению пользы и красоты) восходит к началам создания первых машин и проявилась уже на этапе ремесленного производства. Этот подход можно обнаружить и в самом учении о конструировании; так, уже Рёло в «Учении о конструировании» один из разделов посвятил стилю машин. Тем не менее формы старых рабочих машин, как правило, не сочетаются с современным пониманием пользы.

В более поздних работах, посвященных конструированию, проблема внешнего вида разрабатываемой технической системы решается с помощью простого правила, которое гласит: «все, что целесообразно, — красиво». Эта группа проблем начала формироваться в отдельную дисциплину, имеющую целью выявление закономерных связей между чувством прекрасного и внешним видом конструкции. При этом речь идет не столько о чувстве прекрасного и хорошем самочувствии как следствии эстетического восприятия, сколько о более практических результатах этого восприятия. Влияние эстетического воздействия на умственную и физическую деятельность было доказано примерами повышения производительности при одновременном снижении физиологической усталости организма. Для оказания существенного благоприятного влияния на состояние и способности человеческого организма необходимо использовать все средства стимулирования восприятия органами чувств.

Рекомендации на этот счет собраны в правилах технической эстетики; они призваны придать разрабатываемой системе эстетические свойства.

Основными элементами, посредством которых создаются эстетические свойства, являются:

- симметрия (абсолютная, относительная, контрастная);
- ритм как целесообразное чередование тел, плоскостей, цветов, структур;
- членение (в горизонтальном или вертикальном направлении);
- пропорциональность (соразмерность технической системы, человека и окружения);
- контраст (материалов, форм, размеров, цвета, света);

— композиция (форм, линий, цвета).

Другими элементами, которые используются для достижения желаемых результатов эстетического воздействия (будь то для успокоения, сосредоточения внимания, удовольствия, чувства, гармонии, собранности, радости и т. д.), являются следующие:

— форма (плавная или с резкими переходами);

— материал (дерево и металл в различных случаях и сочетаниях дают разный эстетический эффект);

— структура материала (естественный или обработанный);

— размер и объем;

— линия (горизонтальная, вертикальная, наклонная, прямая, ломаная, кривая);

— цвет (тон, вид поверхности — матовая, блестящая);

— текст (шрифт, цвет, размещение);

— свет (яркий, рассеянный, светотень).

С учетом всего сказанного средствами эстетики можно значительно повысить совокупную ценность технической системы (рис. 7.5).

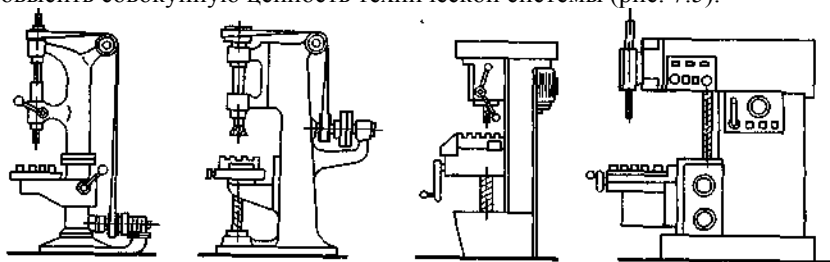


Рис. 7.5. Формы сверлильного станка.

Разумеется, значение в принципе верного эстетического подхода не следует преувеличивать, т. е. не следует сосредоточивать все усилия только на решении задачи красивого внешнего вида. Для каждой технической системы можно найти оптимальное соотношение между техническими, эргономическими и эстетическими свойствами (рис. 7.6).

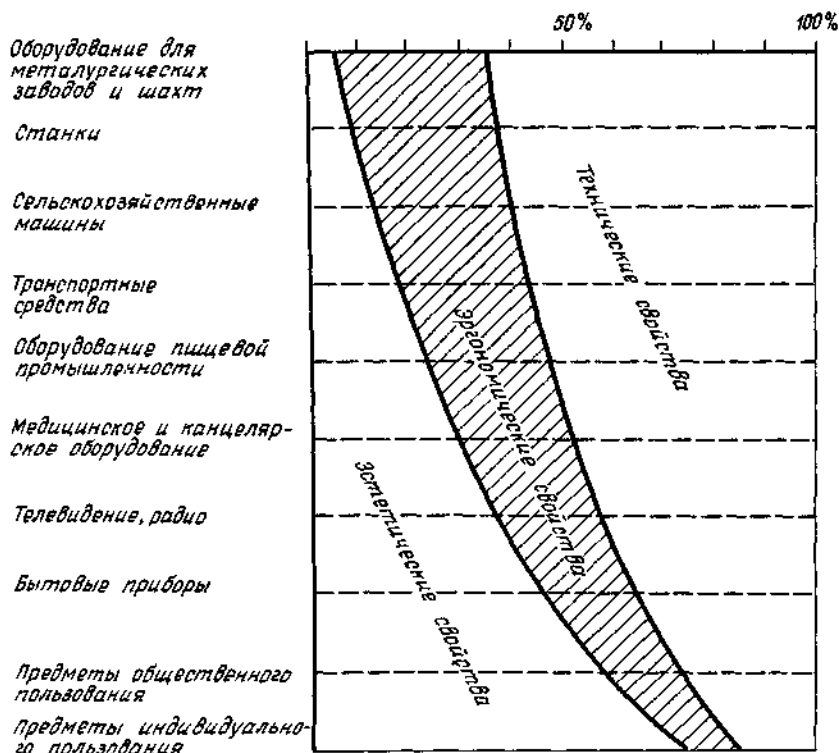


Рис. 7.6. Соотношение между техническими, эргономическими и эстетическими свойствами некоторых типов технических систем.

Теперь возникает вопрос, насколько в силах конструктора при разработке технической системы самостоятельно справиться с этой задачей, т. е. необходима ли ему при этом помощь специалиста по технической эстетике. Однако в любом случае конструктор должен знать, кроме специальных технических вопросов, принципы и правила эргономики и эстетики.

Проблемами внешнего вида технической системы обычно занимается дизайнер. Он имеет дело с вопросами не только эстетики, но и эргономики технической системы, поскольку речь идет об отношении между человеком и машиной.

Важен также момент времени, когда дизайнер привлекается к совместной с конструктором работе. Часто (что совершенно не

оправдано!) конструктор представляет дизайнеру уже сконструированное изделие с определенным расположением функциональных элементов. В этой ситуации участие дизайнера ограничивается лишь эстетическим оформлением. Правильно было бы, однако, начинать сотрудничество уже на этапе разработки функциональной структуры, самое позднее — органоструктуры проектируемой технической системы.

Манипуляционные свойства *Di*. Техническая система изготавливается на предприятии. Вплоть до ее ввода в действие у заказчика она подвергается в общем длительным и утомительным манипуляциям. Сначала она удаляется с места изготовления; затем она должна быть поднята с помощью крана достаточной грузоподъемности и размещена не складе, не заняв при этом слишком много места; далее она должна быть доставлена заказчику на автомобиле, на судне, по железной дороге. При этом она подвергается перемещениям и толчкам, воздействию погодных условий и условий хранения. Она должна быть доставлена на рабочее место и смонтирована в кратчайший срок (ибо монтаж и бездействие системы стоят дорого); наконец, она должна быть пущена в работу. Все эти процессы, условно названные манипуляциями, предъявляют высокие требования к технической системе. Поэтому разработанная техническая система должна обладать такими свойствами, которые позволили бы ей пройти манипуляционные процессы с минимальными затратами и ущербом. Известно, что затраты на складирование, упаковку, транспортировку и монтаж зависят от объема и массы изделия и сказываются на его стоимости.

Ниже указаны некоторые особенности технической системы, оказывающие влияние на характеристики хранения, упаковки и транспортировки изделия, а также приведены соответствующие рекомендации.

1. Габариты изделия являются важнейшим параметром для манипуляционных процессов; они не должны по возможности превышать допустимые для грузовых перевозок габариты.
2. Форма изделия может существенно влиять на удобство транспортировки и погрузочно-разгрузочных работ; если у изделия имеются какие-то выступающие части, то это сказывается на транспортных и упаковочных расходах; нелишне предусмотреть возможность разборки изделия для транспортировки.
3. Вес изделия не должен превышать грузоподъемности кранов и средств транспортировки; при необходимости следует разобрать изделие с расчетом на транспортировку по частям.

4. Хрупкие части изделия, которые могут быть повреждены при транспортировке, должны конструироваться с учетом их демонтажа и отдельной упаковки.
 5. Материалы, которые могут быть повреждены в неблагоприятных условиях транспортировки (низкая температура, тряска, высокая влажность), должны быть указаны в спецификации наряду с соответствующими мерами предосторожности.
 6. Прочность изделия должна быть достаточной для выполнения предусмотренных манипуляционных операций.
- Каждая техническая система должна быть снабжена грузовым крюком или другими приспособлениями для такелажных работ.
- Характеристики упаковки, транспортировки и хранения особенно существенны для изделий серийного и массового производства.
- Манипуляционные операции с такими изделиями должны быть тщательно продуманы (рис. 7.7.).

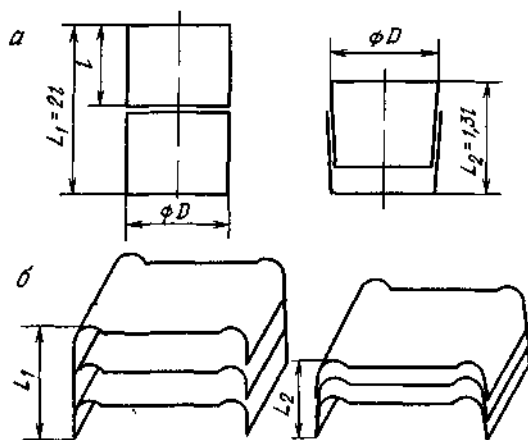


Рис. 7.7. Форма изделий и занимаемый ими объем
 а — кольцевые элементы б — штампованные детали

Требования, предъявляемые к монтажу, рассматриваются ниже в подразделе о технологических свойствах.

Характеристики поставок и планирования LP. Любой заказчик срок поставки или готовности изделия считает важным свойством, имеющим часто решающее значение при выборе подрядчика. С другой стороны, этот срок важен и для подрядчика (конструктора), ибо он очерчивает временные рамки для его работы.

Другой важный аспект работы конструктора — количество изделий, которые предполагается изготовить по выполняемому проекту. Этот фактор оказывает влияние на процесс разработки конструкторской документации, а также на сам процесс конструирования. Здесь возможны такие случаи:

1. Изготавливается одно либо небольшое количество изделий. В этом случае на тщательность формальной обработки чертежей не обращается большого внимания. С другой стороны, возникает проблема воплощения заданной функции и желаемых свойств, поскольку очень часто не предоставляется возможность изготовить и проверить опытный образец.
2. При разработке изделия для серийного или массового производства большое внимание уделяется чертежам. Когда вся техническая документация готова, для проверки выполнения изделием заданной функции изготавливается опытный образец или серия образцов (установочная серия). В рамках процесса подготовки производства закупается или изготавливается также большое количество специальных инструментов, оборудования и других вспомогательных средств. Характеристики поставок и планирования, независимо от типа производства (единичного, серийного или массового), оказывают существенное влияние на совокупное качество, а также на отдельные свойства изделия.

Характеристики соответствия правовым нормам GV. В любом государстве существуют правовые нормы, в рамках которых регламентируются отношения производителей и потребителей. Соответствие технической системы правовым нормам является одним из важных ее свойств. Конструктору необходимо знать все законы и постановления, касающиеся использования конструируемого изделия как в своей стране, так и в стране-потребителе (при разработке изделия на экспорт). Этот аспект часто не учитывается надлежащим образом, следствием чего бывают случаи отказа от изделия или возникновения конфликтных ситуаций. Аналогичным образом конструктор должен знать патентное право и патентную ситуацию в своей стране и в той стране, где разрабатываемая им техническая система будет эксплуатироваться. В случае необходимости, чтобы не нарушить авторское право, конструктор должен консультироваться со специалистами по патентам.

Кроме того, законодательство каждой страны требует, чтобы продукция изготовлялась в соответствии с принятыми техническими правилами и нормами, положениями по охране труда и инструкциями по технике безопасности.

Рассмотренные выше свойства относятся к категории внешних свойств и интересуют главным образом того, кто собирается пользоваться изделиями. Он желает, чтобы приобретаемое им изделие обладало необходимой функцией, параметрами, привлекательным внешним видом, эргономическими свойствами и т. д. Манипуляционные свойства и вопросы соблюдения правовых норм касаются также и изготовителя. Ниже рассматриваются две категории свойств, представляющих особый интерес для изготовителя.

Технологические свойства *Fe*. Технологические свойства дают представление о том, каким образом может быть изготовлено сконструированное изделие. Сколько операций требуется для изготовления детали нужной формы? Какова трудоемкость выполнения операций? Какая технологическая подготовка необходима для выполнения задачи? Какие специальные приспособления, инструменты, измерительные устройства нужно закупить или сконструировать и изготовить? Какая степень точности требуется при изготовлении? Что нужно регулировать и отлаживать при монтаже и вводе изделия в действие? Все эти и многие другие аналогичные вопросы весьма интересуют изготовителя, поскольку оказывают определяющее влияние на издержки производства.

Требования к технологическим свойствам меняются в зависимости от типа производства (единичное, серийное или массовое) и технического уровня производства.

Выбор того или иного конструктивного решения зависит также от сроков разработки и изготовления, так как некоторые технологические процессы могут оказаться неприемлемо длительными. Так, например, вместо литой конструкции может оказаться целесообразным применить сварную.

В любом случае с точки зрения технологических свойств очень важен способ изготовления. Технический прогресс меняет взгляды на этот счет, что можно продемонстрировать на следующем примере.

В свое время ввиду трудоемкости изготовления и обработки литых конструкций перешли к сварным и сборным. Однако благодаря прогрессу, достигнутому в литейном деле, и появлению новых материалов опять вернулись к применению отливок, которые стали теперь дешевле и качественнее сварных и сборных конструкций.

Переход к автоматизированному производству и особенно к механизированному и автоматизированному монтажу обуславливает возникновение новых требований к технологическим свойствам деталей.

Конструктивные свойства *Ko*. Пользователь обычно обращается с технической системой как с неким «черным ящиком». Используя

инструкции и собственный опыт эксплуатации аналогичных изделий, он познает закономерности в отношениях между входами (нажатием на кнопки, воздействием на рычаги) и выходами — функционированием системы, характеризующимися определенными параметрами. Пользователь знает также, что он должен чистить и смазывать свою техническую систему и следовать инструкции по ее применению. Его не интересует, каким образом осуществляется функционирование системы и чем обусловлены другие ее свойства. Ему лишь нужно достигнуть конкретной цели путем выполнения определенной функции — выработать электричество, обработать металл, охладить продукты питания, смолоть кофе — и все с помощью надежного устройства, которое к тому же должно иметь привлекательный внешний вид. В тот момент, когда пользователь отвинчивает и снимает крышку машины и его взору предстают валы, рычаги, шестерни, подшипники, пружины и другие скрытые элементы электрического или гидравлического оборудования, перед ним открывается мир специалиста: с одной стороны, конструктора, сотворившего их, и, с другой стороны, рабочего, изготовившего и пустившего машину в ход. Стоит только взглянуть повнимательнее внутрь такого «черного ящика», как обнаружится множество новых свойств, скрытых от глаз непосвященного. Эти внутренние свойства мы будем называть конструктивными свойствами, ибо они служат конструктору средством реализации рабочей функции и других желаемых внешних свойств. При таком определении конструктивных свойств к ним можно отнести коэффициент сжатия, передаточное число, коэффициент полезного действия, характеристики регулирования, форму, компоновку, прочность, твердость, размеры деталей, допуски, материалы, термообработку и т. д. Приведенные примеры показывают, что такое множество свойств следует классифицировать, поскольку уже с первого взгляда видны зависимости и родство. Так, прочность зависит от размеров, материала и формы, твердость — от материала и термообработки. Эти свойства и связи характерны практически для любых систем. КПД и коэффициент сжатия являются свойствами, которые тесно связаны с происходящим преобразованием; так, на примере бензинового двигателя можно говорить о преобразовании химической энергии бензина в механическую энергию на валу.

Учитывая все это, можно провести следующую классификацию конструктивных свойств:

1. *Специальные конструктивные свойства*, связанные с осуществляемым преобразованием. Иногда довольно трудно устано-

вить границу между функционально обусловленными свойствами и конструктивными свойствами. Впрочем, это не столь важно, важнее чтобы такое свойство не ускользнуло от внимания конструктора.

2. *Общие конструктивные свойства*, которые характерны для большинства технических систем различных классов, поскольку они имеют отвлеченный характер. К ним относятся прочность, жесткость, износ, твердость, упругость, коррозионная стойкость и т. п.

3. *Элементарные конструктивные свойства* являются средством, находящимся в распоряжении конструктора, с помощью которого он реализует все прочие свойства. Это структура (для технических систем высоких уровней сложности); форма, размеры, материал, обработка поверхности, допуски, способ изготов-

- ления (для деталей машин и простых технических систем).

Все специальные и общие конструктивные свойства, так или иначе, создаются конструктором из элементарных конструктивных свойств.

Даже среди элементарных свойств обнаруживаются взаимосвязи, обуславливающие сложность отношений технической системы и затрудняющие процесс конструирования.

Теперь рассмотрим, хотя бы в общих чертах, отдельные группы элементарных свойств, которые, как мы увидим позже, формируют методологию конструирования. Они определяют собственно элементарные операции, такие, как выбор формы, определение размеров, выбор материала, характера поверхности, допусков и способа изготовления.

Структура технической системы. Как указывалось выше, структура и функционирование являются важнейшими свойствами системы.

Функционирование системы однозначно определяется структурой.

Структура — это упорядоченное множество элементов и их отношений между собой. Теперь попытаемся конкретизировать эти общие аксиомы применительно к техническим системам.

Упорядоченное множество элементов, из которых состоит техническая система, группируется по системам более низких уровней сложности.

Пример иерархической структуры системы показан на рис. 6.11.

Деление элементов структуры по уровням сложности проведено в табл. 6.5.

Пока мы находимся на самом низком уровне иерархии, техническая система предстает перед нами в виде множества деталей и узлов, которые служат структурными элементами подсистем более высоких уровней. Следует помнить, что представления о структурных элементах более соответствуют представлениям об элементарных функциях, нежели представлениям о конструктивных элементах в том понимании, как это дается в табл. 6.5. Подшипник скольжения, например,

является узлом, который может состоять из нескольких деталей (корпуса, вкладыша, шариков, обоймы и т. д.). С другой стороны, этот узел представляет собой средство для выполнения элементарной функции вращения. Разумеется, не представляет трудности разобрать подшипник на составные конструктивные элементы. С точки зрения механики структуру машины можно описать как упорядоченную совокупность пар элементов, связанных в кинематические цепи. Техническая система, таким образом, состоит из кинематической цепи и может быть расчленена на подсистемы (механизмы, группы, узлы), которые в предусмотренных комбинациях выполняют подфункции и элементарные функции.

Отношения между подсистемами представляют собой важное средство достижения различных свойств. Если, например, соединить механическим путем в одном случае двигатель, находящийся в передней части автомобиля, а в другом — двигатель, находящийся в задней части, и шасси автомобиля с задним приводом, то результаты будут разными. И хотя схема механизмов останется той же самой, расположение двигателя окажет влияние на прочие свойства, например, на ходовые характеристики, уровень шума в салоне и т. д. Таким образом, функционирование какой-либо технической системы — это не просто сумма функций ее элементов; оно зависит от того, как элементы системы взаимодействуют между собой.

Связи между элементами системы могут быть различными, например механические, электрические, химические, магнитные, временные и пространственные. В рамках этой книги невозможно представить полную классификацию таких связей, поэтому мы ограничимся рассмотрением наиболее важных связей, встречающихся в технических системах.

Механическая связь ограничивает движение точки или тела в пространстве. Точка имеет самое большее три степени свободы, тело — шесть.

Связи бывают (см. рис. 5.8.):

- склерономные (стационарные), т.е. не зависящие от времени; они могут быть представлены системой независимых стационарных условий, выраженных в аналитической форме;
- реономные (изменяемые во времени); они описываются уравнениями, в которых фигурирует время;
- голономные, которые могут быть представлены отношениями между координатами точки и временем, т. е. склерономно-реономные связи;
- неголономные (они выражаются не интегрируемыми аналитически дифференциальными уравнениями).

Ограничения степеней свободы создают:

- односторонние связи, в которых движение тела ограничивается только по одному направлению (например, контакт двух тел или тело, подвешенное на нитке); эту связь можно выразить с помощью неравенств, аналогичных ранее упомянутым уравнениям;
- двусторонние связи, при которых тело путем перемещения или другой механической связи приведено в точно заданное движение или положение.

Далее следует различать:

- силовые связи, которые передают определенную величину силы;
- бессиловые связи, которые не передают силы (кинематическая связь); это не исключает, однако, возможности появления отдельных сил.

Другие виды связей:

- электрическая (токопроводящее или изолирующее соединение, которое можно выполнить в виде последовательной, параллельной или комбинированной схемы);
- химическая (агрессивная или нейтральная, первая ведет к химической реакции, например, коррозии);
- термическая (теплопроводящая, теплоизолирующая);
- магнитная (возбуждаемая магнитным полем или экранирующая от него);
- обратная связь (возвращение той или иной комбинации выходов на вход); такая связь характерна для регулируемых систем.

Форма. Форма относится к наиболее важным элементарным свойствам технической системы. В зависимости от целей, которым она служит, можно говорить о различных видах формы. Форму могут определять:

- функция (крыло самолета, сечение резьбы, профиль фрезы);
- эргономические свойства (рукоятка инструмента, ручка рычага, сидение водителя);
- эстетические свойства (форма утюга, станка);
- манипуляционные свойства (компактность, захваты, крюки, такелажные отверстия и проемы);
- технологические свойства (способы литья,ковки, обработки резанием, сборки);
- экономические свойства (максимально простая, технологичная форма);
- прочностные свойства (приливы, утолщения, фаски, галтели);
- износ (сменные крепления и направляющие);
- материал (поставляемые полуфабрикаты).

Форму, задаваемую какими-либо специальными требованиями, всегда следует оптимизировать в отношении внешнего вида, технологии, прочности, а также с экономической точки зрения.

Размеры. Размеры влияют практически на все свойства. Как и форма, они могут определяться, например, функционированием или прочностью. В таком случае говорят о функциональной или прочностной обусловленности.

Некоторые примеры определяющих свойств и определяемых ими размеров:

- функционирование — диаметр и ход поршня, диаметр ременного шкива,
- эксплуатационные свойства — создание запаса прочности путем изменения расчетных размеров, больший или меньший размер некоторых деталей в целях удобства обслуживания;
- эргономические свойства — размеры пульта управления, пространственное расположение рычагов управления;
- эстетические свойства — размерное соотношение между отдельными деталями, деление поверхностей («золотое сечение»);
- манипуляционные свойства — габариты с учетом характера транспортировки, выполнение условий упаковки, соблюдение стандартных размеров контейнеров;
- технологические свойства — толщина стенок отливки или поковки, радиус изгиба, чистота обработки;
- экономические свойства — простота конструкции, минимальные, по возможности, размеры с целью уменьшения затрат материала;
- точность — размеры цапф, подшипников, ширина и модуль шестерен;
- жесткость — габариты станины;
- износ — припуск на износ поверхностей трения;
- материал — размеры полуфабрикатов и заготовок;
- заданные монтажное пространство или установочные размеры — стандартный фланец или колено с нормированным радиусом изгиба.

Материал. Материал, из которого изготавливают детали машин, также определяется требуемыми внешними свойствами технической системы. Приведем примеры, как и выше, в форме определяющий фактор — определяемые свойства материала:

- функция — теплоизоляция, электрическая изоляция, химическая стойкость, поглощение вибрации;
- эксплуатационные свойства — износоустойчивость, долговечность, безопасность материала;

- эргономические свойства — остекление операторской, мягкие сидения в автомобиле;
 - эстетические свойства — деревянная облицовка, латунные ручки, эмалевые покрытия;
 - манипуляционные свойства — противокоррозионные покрытия и противоударные приспособления;
 - технологические свойства — выбор материала из соображений, например, обработки резанием или штамповки;
 - экономические свойства — принцип применения самого дешевого материала, удовлетворяющего всем другим требованиям;
 - прочностные свойства — легированные жаропрочные стали для высоконагруженных конструкций, например для лопаток паровых и газовых турбин;
 - износ — облицовка направляющих поверхностей станков сменяемым синтетическим материалом;
 - твердость — твердосплавная сталь для сверла, металлокерамические детали;
 - качество обработки поверхности — выбор оптимального, с точки зрения обеспечения заданного качества поверхности, материала;
 - коррозионная стойкость — легированные, например, хромоникелевые стали или полимеры.
- Качество поверхности.* Под качеством поверхности мы понимаем вид и цвет поверхности. Для достижения желаемых внешних свойств изделия оно зачастую важнее, чем принято считать. Качество поверхности влияет на целый ряд характеристик, требованиям которых оно должно удовлетворять:
- функционирование — направляющие поверхности, поверхности подшипников скольжения;
 - эксплуатационные свойства — длительный срок службы благодаря лучшим качествам поверхности;
 - эргономические свойства — отсутствие заусениц и острых кромок, рифленые рукоятки, исключающие выскальзывание из рук оператора;
 - внешний вид — качество поверхности в значительной мере определяет внешний вид;
 - экономические свойства — минимум обрабатываемых поверхностей;
 - прочностные свойства — гладкие поверхности и плавные переходы увеличивают усталостную прочность (долговечность);

- коррозионная стойкость — гладкие поверхности обычно более стойки в отношении коррозии;
- износ — хорошо обработанные поверхности, как правило, лучше противостоят износу;
- допуски — лучшее качество поверхности обеспечивает более точную сборку.

Допуски. Поле допусков влияет на пространственные отношения между элементами. Выбор допусков связан с целым рядом свойств более высокого уровня, например:

- функционирование — допуск на изготовление элементов подшипника, допуски изготовления вала и втулки котеса для горячего прессового соединения;
- эксплуатационные свойства — возможность замены деталей в узле;
- эргономические свойства — большие допуски вызывают сильные шумы в машинах;
- технологические свойства — допуски деталей и узлов без последующей подгонки при сборке и монтаже;
- износ — большие допуски ускоряют износ деталей, контактирующих в процессе движения;
- способ изготовления — допуск в значительной мере определяется процессом обработки;
- размеры — существует связь между размерами и возможной точностью их выполнения.

Способ изготовления. Способ изготовления через общие конструктивные свойства косвенно влияет на внешние свойства изделия. Между способом изготовления и элементарными конструктивными свойствами, напротив, существует тесная взаимосвязь, что видно на примерах:

- прочность — кованные детали прочнее деталей, изготовленных путем обработки резанием;
- твердость — закаленные детали обладают более твердой поверхностью;
- вязкость — термообработанные детали приобретают нужные механические характеристики, в том числе требуемую вязкость;
- форма — некоторые детали сложной формы можно изготовить только литьем;
- материал — свойства материала зачастую определяют способ изготовления — токарная обработка, штамповка, литье;
- поверхность — способ обработки определяется заданным качеством поверхности;
- размеры — величина детали определяет способ ее изготовления.

Конструктивные признаки. Понятие «конструктивный признак» уже не раз встречалось выше в различных контекстах. Здесь мы еще раз напомним, что конструктивный признак представляет собой внутренний признак технической системы, классифицирующий и характеризующий ее понятным для специалистов образом.

К конструктивным признакам относятся группы специальных и общих конструктивных свойств. Группа элементарных конструктивных свойств была рассмотрена на предыдущих страницах книги. Понятие «конструктивный признак» широко применяется в литературе, но трактуется неоднозначно и непоследовательно.

На некоторых примерах попытаемся пояснить понятие «конструктивный признак». Проанализируем следующее описание: «Это поршневая машина с четырьмя цилиндрами в ряд, работающими в четырехтактном режиме, создающем определенную степень сжатия, с прямым впрыскиванием топлива и водяным охлаждением. В машине используются плавающие цапфы, уплотнительные прокладки между цилиндром и его головкой, поршень с неразрезной юбкой и подшипники качения. Первое критическое число оборотов вала выше рабочего числа оборотов».

Все указанные в этом описании признаки характеризуют для специалиста данную машину в различных аспектах. Опираясь на них, он может судить не только о принципе действия или эксплуатационных свойствах, но и о своеобразии органоструктуры, конструктивной схемы и т. д.

Этот пример вводит еще один аспект рассмотрения конструктивных свойств и признаков — их конкретное воплощение. При анализе этого описания у нас возникает вопрос: существуют ли какие-нибудь другие возможности реализации этой технической системы, кроме поршневой машины, четырехтактного режима, заданной степени сжатия, прокладок и подшипников? Совершенно очевидно, что в этом примере в качестве описания определенной технической системы указаны не конструктивные признаки, а их воплощение.

По своей сути конструктивный признак занимает место между внешними свойствами, определяющими назначение технической системы, и элементарными конструктивными свойствами, служащими средством. Диаграмма на рис. 7.13 иллюстрирует конструктивные признаки по отношению к двум другим категориям. Эти отношения далеко не всегда можно выразить аналитически. Часто они отражают лишь определенный профессиональный опыт.

В табл. 7.3 систематизированы некоторые общие конструктивные признаки на четырех уровнях абстрактности (система преобразований, функциональная структура, органоструктура и конструктивная схема) .

Эти данные также делают более ясным понятие конструктивного признака.

Конструктивные признаки могут находиться как в равноправных, так и в иерархических отношениях. Иерархические отношения, задаются, как правило, через определения, например, структура устанавливает совокупность элементов и их соподчиненность. При этом подчиненные конструктивные признаки определяются через конструктивные признаки более высокого порядка. Так, например, функциональная структура содержит конструктивные признаки из области системы преобразований (например, технология, рабочие воздействия и участие технической системы в их осуществлении) и устанавливает другие для органоструктуры, например формы входных воздействий, функции преобразования и функциональные группы элементов.

Экономические свойства *We*. Экономические свойства объединяют ряд уже описанных свойств в единый показатель. Они представляют, с одной стороны, затраты, с другой — доход. Расходы возникают на стадиях разработки конструкции, подготовки производства, изготовления, реализации и эксплуатации изделий. В качестве дохода от выполняемого системой рабочего процесса понимают полезность выполняемых ею преобразований, т. е. совокупный эффект рабочего действия (см. также в табл. 3.3 показатели технологических процессов).

Для экономической оценки производства необходимо знать себестоимость используемой технической системы, элементы которой приведены в табл. 7.3.

Таблица 7.3.

Расчетные схемы цены, себестоимости и затрат

Цена	Себестоимость	Производственные расходы	Расходы на основной материал Общие затраты на материалы Оплата труда Одноразовые затраты на изготовление (например, лицензионные) Общие затраты на изготовление
		Общие опытно-конструкторские расходы Общие административные расходы и расходы по сбыту Одноразовые накладные расходы	
	Различные надбавки и налоги, норма прибыли		

Цена	Себестоимость	Производственные расходы	Затраты на изготовление	Расходы на основной материал Общие затраты на материалы Почасовое и штучное время машин Оплата труда Общие социальные расходы Общие затраты на изготовление
			Возможные одноразовые производственные затраты	
		Общие опытно-конструкторские расходы Общие административные расходы и расходы по сбыту Одноразовые накладные расходы		
		Различные надбавки и налоги, норма прибыли		

Для изготовителя себестоимость, является самым значимым экономическим свойством. Она охватывает все прямые затраты на создание технической системы, такие, как затраты на материалы, оплату труда, исследования, разработку, испытания, подготовку производства-, потребление энергии в процессе производства и т. п. Поскольку не все перечисленные выше затраты можно отнести к конкретной изготовленной системе, некоторые общие виды затрат включают в накладные расходы.

Оценка отдельных статей расходов на техническую систему, т. е. анализ расходов, необходимых для достижения требуемых свойств системы, раскрывает много интересных аспектов конструкции. При этом важны как абсолютная величина расходов, так и взаимосвязь отдельных статей. Например, соотношение расходов на материалы и оплату труда является информативным показателем. На рис. 7.8 показано, какой различной может быть доля расходов на материалы в зависимости от изделий.

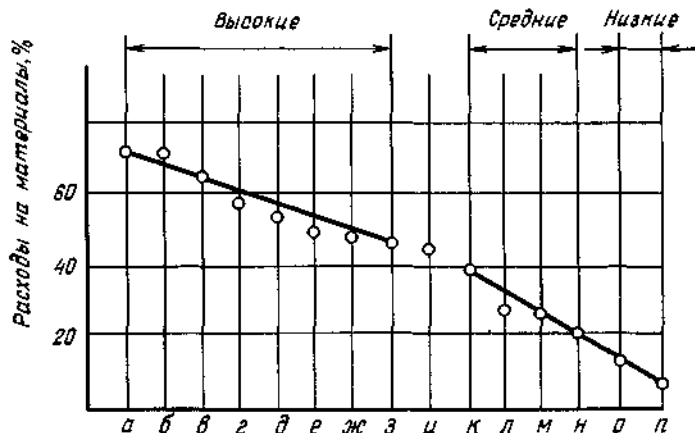


Рис. 7.8. Доля расходов на материалы в затратах на изготовление некоторых изделий.

а — строительные конструкции, б — цистерны и другие емкости, в — железнодорожные вагоны, г — грузовые автомобили, д — подъемно-транспортное оборудование, е — дизельные двигатели, ж — станки, з — легковые автомобили, и — усилители и трансформаторы, к — измерительные приборы, л — готовальни, м — часы, н — стеклянные приборы, о — электронные приборы, п — полупроводниковые приборы

Предварительная оценка себестоимости производится по эмпирическим формулам, отражающим накопленный опыт, для каждой статьи расходов. Основой для расчета накладных расходов являются расходы на оплату труда (табл. 7.3), а также нормативы.

Себестоимость служит исходным пунктом для определения ориентировочной цены — самого важного экономического свойства изделия. Рыночная цена и себестоимость могут сильно отличаться от ориентировочной расчетной цены.

Величина партии изготавливаемых изделий и технология влияют на стоимость изготовления изделия (рис. 7.9).

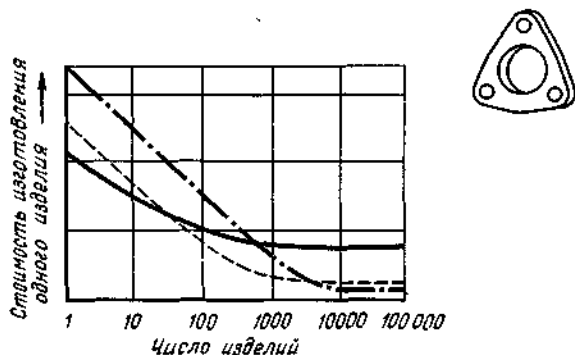


Рис. 7.9 Влияние технологии и величины партии изделий на стоимость изготовления одного изделия.

— — — — технология, ориентированная на массовое производство

Стоимость изготовления в целом снижается с ростом количества изготавливаемых изделий. Для общих производственных затрат важным показателем является оптимальное число изделий в партии (рис. 7.10).

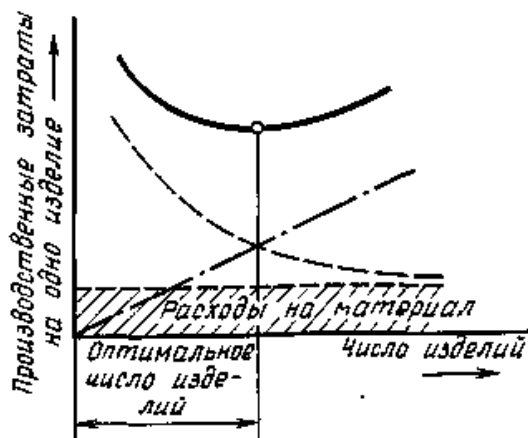


Рис. 7.10. Оптимальное (с точки зрения производственных затрат) число изделий в партии.

----- стоимость изготовления, — — — — расходы на хранение и финансирование, ——— общие производственные расходы

Производственные расходы зависят также от степени новизны изделия и его деталей в соответствии с категориями рис. 6.13.

Предварительный расчет производственных расходов позволяет изготовителю технической системы решить, выгоднее ли изготавливать некоторые детали у себя или же покупать их.

Для потребителя очень важны эксплуатационные расходы, связанные с поддержанием работоспособности и нормальной работой технической системы. Важнейшим показателем является амортизация затрат за время службы системы, определяемых закупочной ценой, расходами на транспортировку, монтаж, отладку и обеспечение производственного процесса.

Эксплуатационным расходам противопоставляется достигнутый за тот же период времени экономический и социальный эффект (полезность). Полезный эффект выражается в денежных либо технических единицах. Соотношение полезный эффект/эксплуатационные расходы называется *экономической эффективностью*, которая представляет собой один из важнейших экономических показателей.

Другой важной экономической характеристикой является рентабельность, которая определяется отношением прибыль/капиталовложения за тот же период времени.

Так как расчет экономических свойств — дело довольно хлопотное и не входит непосредственно в обязанности конструктора, такими расчетами он зачастую пренебрегает на самых важных стадиях создания технической системы в ущерб оценке и выбору оптимального варианта.

Качество изготовления *He*. Недостаточно лишь хорошо сконструировать изделие; оно должно быть и хорошо выполнено, что сказывается на его функционировании. Качество изготовления является лишь результатом эффективного производства, которое представляет собой самостоятельную задачу, так как производственный процесс можно осуществить по-разному.

Слаженный и квалифицированный рабочий коллектив, пожалуй, сможет, опираясь на заурядные конструкторские чертежи, выпустить хорошую продукцию, тогда как посредственная бригада рабочих, даже используя отличные конструкторские чертежи, не застрахована от неудачи. Название фирмы, если она зарекомендовала себя у потребителей, служит в некоторой степени гарантом качества. Конструктор в процессе своей работы должен считаться с факторами влияния характера производства на качество изделия. Такими факторами являются квалификация операторов и техническая оснащенность предприятия. Конструктор должен приспосабливать

технические требования чертежей к этим факторам. Таким образом, контакт конструктора с производством — важное условие получения качественного изделия.

7.3. Отношения между свойствами

Общее определение отношений приведено ранее. Рис. 7.11 иллюстрирует отношения между свойствами в наглядной форме.

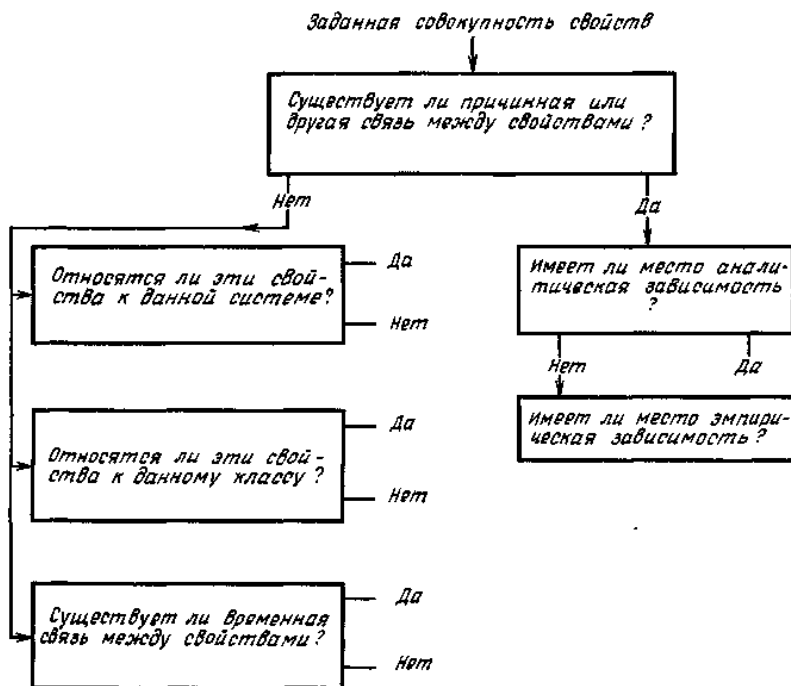


Рис. 7.11. Определение отношений между свойствами.

Выходы «Да» представляют классы отношений. Математическая логика и статистика являются важнейшими дисциплинами при изучении отношений; они позволяют выразить эти отношения аналитически, что позволяет оперировать ими. Свойства и их общие отношения изучаются в естествознании и технике, где они выражаются либо формулами, либо на словах. Для осуществления требуемого действия конкретной технической системы

общие отношения часто недостаточны. В случае сложных процессов общие отношения на практике всякий раз конкретизируют применительно к имеющимся условиям. Так, например, для распространения тепла имеются формулы переноса тепла теплопроводностью, излучением и конвекцией. Какие отношения имеют место, когда мы помещаем предмет в нагревательную печь, и каков их относительный вклад? Для ответа на эти вопросы конструктор должен обладать опытом в данной области, который позволил бы верно оценить конкретные условия в рассматриваемой ситуации. Особенности производства можно затем учесть поправочными коэффициентами в общих формулах. Очень сложные отношения между свойствами можно выразить также посредством матриц и математических символов, о чем говорилось ранее. Важно оценить и отношения между категориями свойств (рис. 7.12).

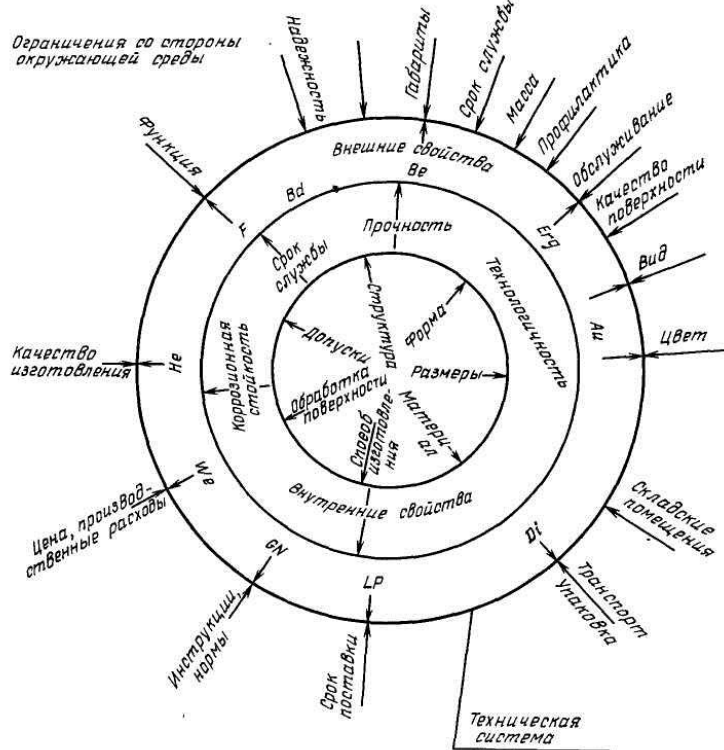


Рис. 7.12. Отношения между категориями свойств.

Из рисунка видно, что у готового изделия внутренние элементарные свойства (конструктивные и технологические) являются определяющими для внешних свойств, которые определяют в свою очередь экономические свойства.

Связи между категориями свойств показаны на рис. 7.13.

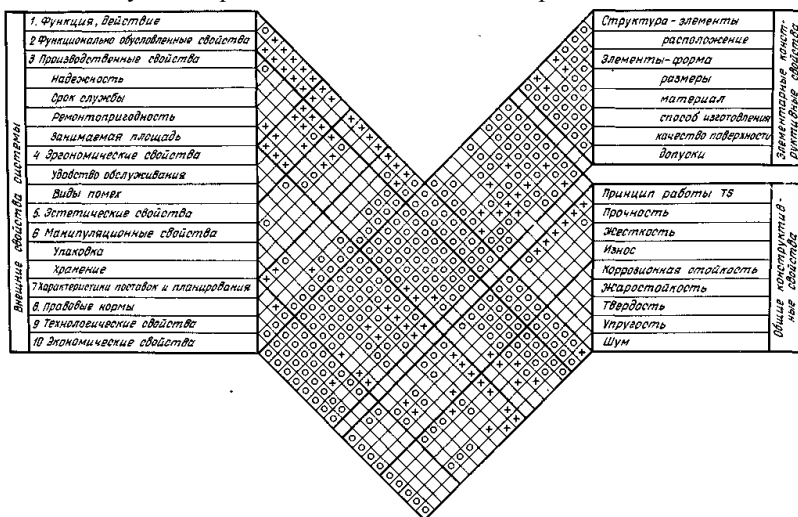


Рис. 7.13. Связи между категориями свойств и конструктивными признаками.

О непосредственная связь, + косвенная связь

Другая схема отношений между свойствами представлена на рис. 7.14.

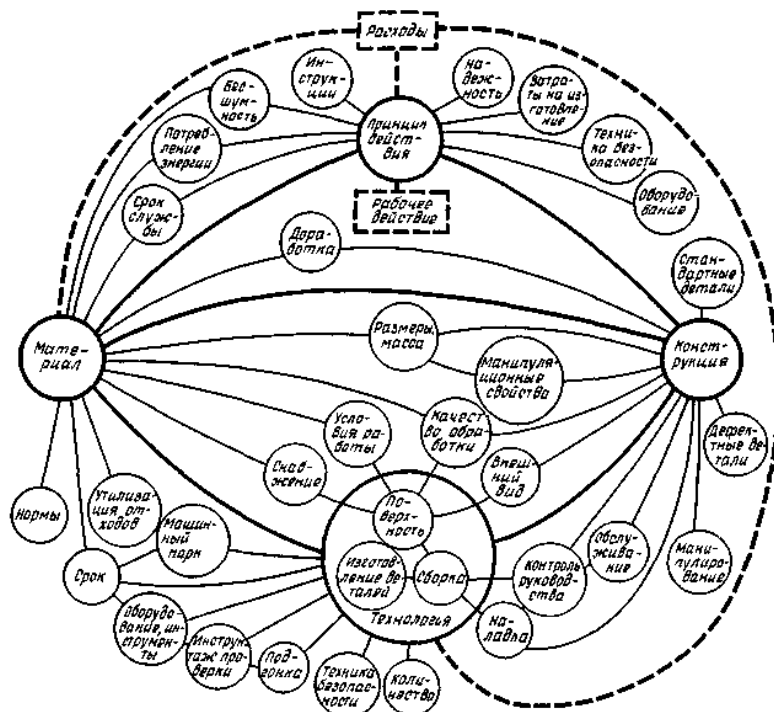


Рис. 7.14. Отношения между различными свойствами технической системы по Вёгербауэру.

Приведенные примеры обнаруживают чрезвычайную сложность отношений. Сложность еще более возрастает с учетом отношений в технической системе на более низких уровнях.

7.4. Установление свойств и их отношений

Чтобы сравнить свойства технической системы с требованиями, предъявляемыми к ним, или чтобы сравнить между собой различные системы и охарактеризовать их, следует определить свойства. Методы определения свойств технической системы меняются в зависимости от стадии ее существования, будь это концепция, изготовление или эксплуатация технической системы.

ис.

Для определения свойств используются в основном следующие методы и приемы: 1) измерения; 2) экспертные оценки; 3) моделирование; 4) вычисления или считывание; 5) сравнение; 6) определение оптимального уровня свойств.

1. *Измерение.* Определяемые количественно свойства технических систем, находящихся в эксплуатации, можно установить путем измерения, а определяемые качественно — с помощью экспертной оценки специалистов. Хотя в этой главе не рассматриваются подробности операций такого рода, мы все же слегка коснемся этой проблемы. Измерения — это профессиональные операции, имеющие свои рабочие методики. Как и в других операциях, подготовка к измерениям, которая включает разработку инструкций и формуляров, имеет немаловажное значение.

Особый подход необходим при продолжительных измерениях, например при определении срока службы деталей. Требуемые свойства можно определить и косвенным путем — по известной зависимости одной величины от другой, измеряемой проще, например срок службы в зависимости от износа.

2. *Экспертная оценка.* Определенную сложность представляет собой экспертная оценка неизмеряемых величин. Чтобы получить надежные результаты, нужно точно определить свойства и критерии их оценки. Если, например, о технической системе сказано, что она ремонтпригодна, то эта оценка должна быть подкреплена такими аргументами, как легкость определения отказа, доступность деталей, быстрота ремонта, гарантии фирмы-изготовителя и т. д. Еще сложнее выглядит ситуация с экспертной оценкой, если о системе имеется лишь приблизительное представление. В этом случае приходится использовать моделирование.

3. *Моделирование.* Исследование абстрактной машинной модели много сложнее измерений физической реальности. Пока конструктор может основывать свою идею на опыте схожих ситуаций, его выводы еще относительно просты. Напротив, когда требуется оценить новую ситуацию, для объективной оценки следует использовать моделирование.

Модель — это представление реального объекта или процесса доступными физическими или математическими средствами.

Отношения между моделью и оригиналом регламентируются законами подобия. Моделирование находит все более широкое применение благодаря использованию вычислительных машин.

Прежде всего следует выяснить, какого рода подобие между моделью и оригиналом нас интересует, т. е. какие свойства оригинала должны

быть учтены в модели и какие цели преследуются при этом. В отличие от опытного образца системы, у которого большинство свойств может быть определено, у модели определимы только некоторые, такие, как функционирование, структура, форма и т. п. Соответственно разрабатываются функциональные, структурные и масштабные модели. Модель всегда имеет определенное назначение. Речь идет не только об определении свойств; модель часто служит средством контроля, коммуникаций, инструктажа и обучения. В зависимости от средств моделирования различают следующие виды моделей:

- а) *Иконические*, которые воспроизводят оригинал в двух- или трехмерном измерении, по необходимости в уменьшенном или увеличенном масштабе, например чертежи, объемные модели машин и станков, фотографии. При этом внешнее сходство между моделью и оригиналом велико.
- б) *Аналоговые*, которые сходны с оригиналом только по некоторым функциональным свойствам. С помощью этих моделей можно воссоздать, например, статические и динамические свойства оригинала. К ним относятся графы, диаграммы и физические модели, в которых используется подобие явлений.
- в) *Символические*, которые изображают свойства оригинала с помощью слов или математических символов.

Закономерности сходства являются предметом теории подобия. Теория подобия опирается на теорию размерностей и критерии подобия, которые позволяют выразить процесс или свойство посредством безразмерной функции.

Целью экспериментов с моделью является подтверждение или отклонение определенной гипотезы, либо получение данных для количественного выражения закономерностей. Эксперименты следует проводить по тщательно разработанному плану, в котором должны содержаться точные указания об условиях проведения экспериментов и обработки результатов.

4. *Вычисления и считывание.* Когда имеются графики, можно получить искомые показатели для данных условий путем считывания. Показатели, выражаемые формулами, можно определить вычислением с применением аналоговых или цифровых вычислительных машин.

5. *Сравнение.* Эталон, например, формы поверхности (профиль зуба) служит вместе с тем и моделью. В этом случае степень реализации искомого свойства определяют путем сравнения изделия с образцом.

6. *Определение оптимального уровня свойств.* В технике часто необходимо определить оптимальный уровень свойства для заданных

условий, например оптимальные скорость, мощность, форму, количество.

Для определения оптимального уровня свойства используются знания об отношениях между свойствами. Обычно изменения условий на одни свойства воздействуют положительно, на другие — отрицательно.

Оптимальный уровень свойства может изменяться а) дискретно (например, число несущих опор, цилиндров двигателя, винтов) и б) непрерывно (например, толщина покрытия, скорость движения). Вернемся в связи с этим к рис. 7.10, где определяется оптимальное число изделий в серии. Известно, что с ростом количества изделий в серии удельные расходы на зарплату и накладные расходы понижаются, а расходы на материал для одного изделия остаются постоянными. Однако возникают дополнительные расходы на хранение, транспортировку и финансирование, так как не все изделия могут быть сразу отправлены потребителю (введены в действие). Если сложить все расходы, то получим кривую, минимум которой и определяет оптимальную величину серии.

Определение оптимального уровня свойства может быть осуществлено графически или аналитически путем расчета.

Широкое распространение получил аналитический метод, что связано с использованием вычислительных машин, сделавших возможными расчеты, которые ранее считались нецелесообразными из-за колоссальных затрат труда и времени.

Рассмотрим кратко метод определения оптимума, когда в качестве критерия используется минимум себестоимости или максимальная экономическая эффективность E_f .

а) *Дискретно изменяющиеся свойства.* Оптимальный вариант по себестоимости (и соответственно эффективность) определяется путем расчета всех дискретных вариантов. В сложных случаях применяют метод линейного, нелинейного или динамического программирования. Результаты расчетов помещают в таблицы или на диаграммах; наилучшим вариантом оказывается тот, для которого экономическая эффективность максимальна.

б) *Непрерывно изменяющиеся свойства.* В таком случае оптимальный уровень свойства определяется аналитическим методом. При этом рассматриваемое свойство E представляется в виде функции непрерывно изменяющегося оптимизирующего параметра x , например $E = F(x)$. Параметр x имеет оптимальное значение x_0 , когда $F(x)$ принимает экстремальное значение, определяемое из уравнения

$$F'(x_0) = \frac{dE}{dx} = 0 .$$

В зависимости от того, что нас интересует — максимум или минимум, — выбирается вариант, для которого вторая производная

$F''(x) = \frac{d^2E}{dx^2}$ меньше или больше нуля.

Трудностей с вычислением производных в большинстве случаев не возникает, однако уравнение, в котором первая производная равна 0, часто более высокого порядка, так что его однозначное аналитическое решение относительно x невозможно. В этом случае ищется приближенное решение численным методом.

Аналогичный метод применяют, когда определяют оптимум функции двух непрерывно изменяющихся параметров. Оптимизацию по двум параметрам нельзя проводить независимо. Исследуемое свойство выражается как функция параметров x , y , например расход через трубопровод в зависимости от его поперечного сечения и срока службы. При этом свойство E определяют как функцию оптимизирующих параметров x и y :

$$E = F(x, y).$$

Первые частные производные функции $F(x, y)$ по переменным x и y записываются так:

$$F'_x(x, y) = \frac{\partial E}{\partial x}; \quad F'_y(x, y) = \frac{\partial E}{\partial y}$$

При условии равенства нулю обеих частных производных получается система двух уравнений для оптимальных значений x_0 и y_0 :

$$F'_x(x_0, y_0) = 0; \quad F'_y(x_0, y_0) = 0,$$

из которой определяются x_0 и y_0 .

Все описанные выше операции по определению свойств и их отношений представляют собой элементарные операции процесса конструирования, которые более подробно рассматриваются в учебниках и пособиях по конкретному конструированию.

7.5. Перечень требуемых свойств технической системы

Когда говорят о технических требованиях, технических условиях или техническом задании, то всегда подразумевают наличие полного перечня точно сформулированных требуемых свойств технической системы. Нередко неполное указание требуемых свойств становится причиной дефектов готовых изделий; при этом уже неважно, явилось ли это следствием упущения либо ошибочного предположения о том, что наличие определенного свойства технической системы и/или существенно или является само собой разумеющимся.

Важность правильного и полного перечня требуемых свойств отражает известное положение: правильная постановка задачи — это уже половина решения. Содержание и форма перечня требуемых свойств различны в каждом конкретном случае; их обуславливают факторы:

- 1) сложность выполняемой функции — она различна, например, для производственного комплекса и отдельной детали машины;
- 2) конструктивная сложность — сложная конструкция в отличие от простой, новая конструкция в отличие от модернизируемой;
- 3) потребность в дополнительных свойствах, таких, как высокая надежность, большой срок службы, привлекательный внешний вид;
- 4) требования заказчика.

Первые три фактора уже были рассмотрены. Для пояснения последнего рассмотрим вопрос об *удовлетворении производственной потребности*. Существуют две возможности (рис. 7.15).

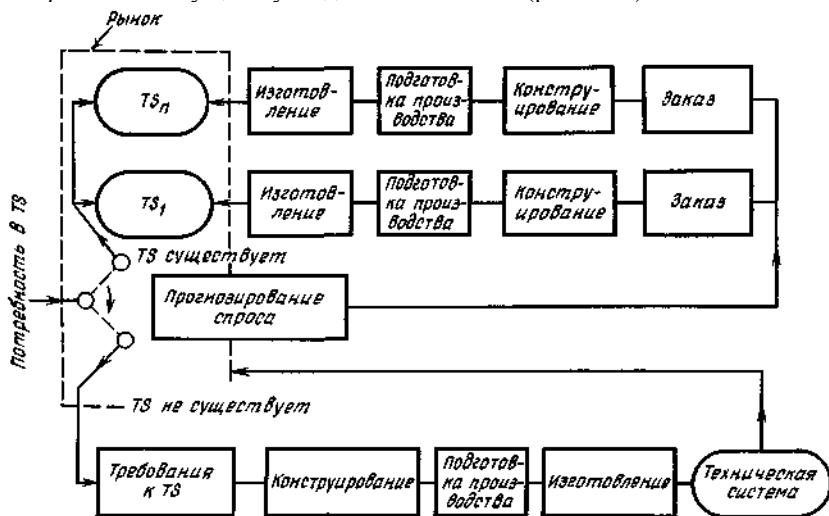


Рис. 7. 15. Способы удовлетворения производственных потребностей

а) *Закупка готового изделия.* Потребитель, желающий решить свою проблему с помощью того или иного изделия, выясняет, имеется ли в продаже изделие с желаемыми свойствами. Как правило, он имеет возможность выбирать между несколькими изделиями.

б) *Заказ специального изделия.* Если изделия с желаемыми свойствами нет в продаже, потребитель вынужден ожидать, пока будет пройден

долгий путь от момента оформления заказа через стадии конструирования и изготовления до поставки изделия. Изделие, изготовленное с учетом индивидуальных требований заказчика, соответственно обходится дороже. Для правильных постановки и решения такой задачи требуется активное сотрудничество ряда специалистов, среди которых конструктор должен играть главную роль.

Так как постановка задачи никогда не бывает идеальной, то при выполнении каждого заказа конструктор должен придавать проектируемой системе дополнительные свойства. Заказчики определенного рода, такие, как армия, почта, железная дорога, страховые агентства, нередко предъявляют особые требования, исходя из своего профессионального опыта. Все сказанное относится к изделию в целом, которое потребитель получает. Требования к подсистемам и элементам обычно определяет сам конструктор. Для обеспечения необходимой полноты перечня свойств целесообразно рассмотреть вопросы типа приведенных в табл. 7.1 и найти на них ответы. На основе соответствующих формулировок можно затем составить перечень свойств для конкретного изделия. Особую группу образуют постоянные *требования, которые не устанавливаются в явной форме*, но подразумеваются практически всегда. Их учитывает, пожалуй, каждый конструктор. К таким требованиям относятся:

- 1) максимально достижимый уровень эксплуатационных свойств, среди них, в частности, минимальные габариты, минимальная масса; минимальное потребление энергии, доступность и заменяемость всех элементов, имеющих ограниченный срок службы, оптимальная надежность;
- 2) оптимальные эргономические показатели: простое, легкое и удобное обслуживание; защита от вредных побочных выходов (помех), таких, как шум, тепло, вибрация, выхлопные газы, пыль и т. д.; минимальный вред окружающей среде;
- 3) максимальный учет всех особенностей существующего производства: применение имеющихся материалов, стандартных деталей и документации; использование имеющегося оборудования и технологии;
- 4) наилучшие экономические показатели: минимальные издержки производства; минимальные эксплуатационные расходы.

Из рассмотренных требований логически вытекают ряд принципов, которыми руководствуется конструктор, а именно:

- 1) выбор наиболее простого конструкторского решения;

- 2) одинаковый (не всегда максимальный) срок службы всех деталей или легкая заменяемость деталей, подверженных износу;
- 3) оптимальное использование прочности материалов;
- 4) учет особенностей окружения, включая естественные ресурсы и население;
- 5) уважение существующих традиций.

Некоторые из этих требований противоречат друг другу, и приходится искать компромиссное решение. Эта проблема будет рассмотрена позже.

Требованиями особого рода являются *ограничения*. Они диктуются конструктору природой и обществом, вследствие чего он ограничен в выборе решений. Природные факторы обычно не воспринимаются как ограничения, так как мы привыкли к законам природы и осознаем, что нельзя сконструировать вечный двигатель.

Общество накладывает ограничения правового характера.

Следует подразделять требования по их значению для покупателя на три категории:

- 1) Требования, которые обязательно должны быть выполнены, такие, как рабочая функция и цена. Их в свою очередь подразделяют на *жесткие требования*, отклонение от которых недопустимо, и *нежесткие требования*, отклонения от которых могут быть терпимы.
- 2) Требования, которые в исключительных случаях не обязательно выполнять в заданном объеме.
- 3) Пожелания, которые следует выполнять при благоприятных обстоятельствах.

Конструктор должен, сообразуясь с точкой зрения заказчика, определять, к какой категории относится каждое требование.

Теперь мы рассмотрим вопросы о мысленной реализации свойств технической системы и о том, от чего зависит ценность созданной технической системы. Бесспорно, что без качественного воплощения даже самый лучший проект остается бесполезным рисунком на бумаге, а без четкой организации производства пришлось бы потратить на изготовление неоправданно много времени и средств. Однако без технических знаний и работы фантазии конструктора было бы немисливо удовлетворить высокие современные требования к изделиям.

Конструктор является истинным создателем изделия и его функциональных свойств; в то же время он в меньшей степени влияет на качество изготовления и характеристики планирования и поставки, лишь создавая для них предпосылки. Он участвует в формировании внешнего вида и эргономических свойств, работая по возможности

совместно с дизайнером; он существенно влияет на экономические свойства изделия.

Ценность технической системы создается в процессе конструирования. К факторам, влияющим на результат этого процесса, относятся технические знания конструктора, длительность конструирования, количество участников и сложность создаваемой технической системы. Вёгербауэр построил зависимость качества проекта (как меры выполнения поставленных требований) от квалификации конструкторов (рис. 7.16).

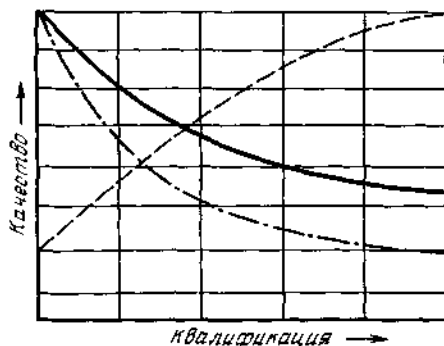


Рис. 7.16. Влияние квалификации конструкторов на качество разработки

----- совокупная ценность (качество), — затраты материалов,
— · — производственные расходы

К сожалению, термины «квалификация» и «качество проекта» не имеют точного определения и поэтому не получили количественного выражения. Зависимость качества проекта от длительности конструирования и от количества участников показана на рис. 7.17.

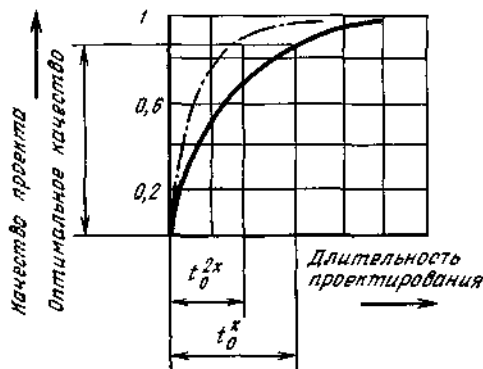


Рис. 7.17. Зависимость качества проекта от длительности процесса конструирования и количества конструкторов.

----- x конструкторов, — — — — 2x конструкторов. Основные факторы, от которых зависит ценность сконструированной «стемы» квалификация конструкторов (рис. 7.16), время конструирования и число усовершенствований

Численность конструкторского персонала можно выразить количеством денежных затрат в единицу времени. Из рисунка видно, что, с одной стороны, за время t_0^x качество возрастает относительно медленно, а с другой — использование двойной численности конструкторского персонала не обеспечивает пропорционального ускорения работы. График рис. 7.17 подходит только для хорошо подобранного и организованного конструкторского коллектива, который обладает определенным объемом специальных знаний. Для другого коллектива график выглядел бы иначе; так, например, для более слабого коллектива расстояние между кривой и асимптотой идеального решения на графике было бы значительно большим, а наклон кривой менее крутым. Любое усовершенствование существующей системы должно преследовать получение новых свойств или достижение более высоких уровней существующих свойств. Такой процесс можно изобразить на S-диаграмме (см. рис. 8.2). Однако возможности усовершенствования существующих технических систем ограничены.

Интересно обратить внимание на то, в какой степени конструктор влияет на величину издержек производства. Существует убеждение, что эти расходы зависят преимущественно от подготовки производства и самого производственного процесса. Однако более глубокое

исследование показывает, что конструкторские работы могут определяющим образом влиять на себестоимость изделия (рис. 7.18).

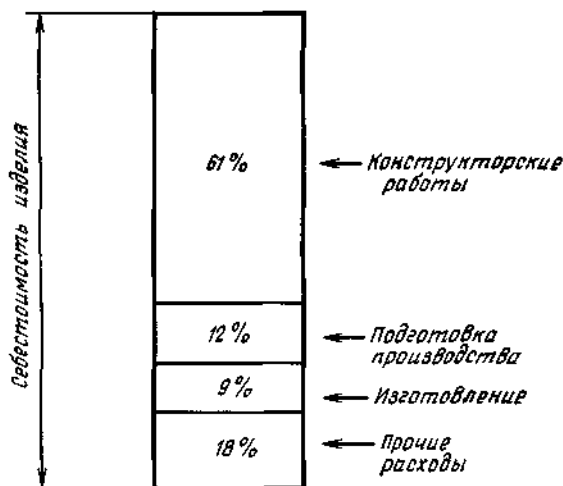


Рис. 7.18 Примерное распределение расходов при единичном производстве сложных изделий.

Рис. 7.16—7.18 имеют ориентировочный характер и их не следует интерпретировать буквально.

Свойство как категория, определяющая каждый объект и относящаяся к его сути, занимает центральное место в теории технических систем. Из рассмотрения свойств, проведенного в данной главе, можно сделать следующие выводы.

1. Для того чтобы техническую систему можно было применять, она должна выполнять рабочую функцию и обладать другими свойствами с определенным уровнем их воплощения. Некоторые из этих свойств обеспечивают выполнение рабочей функции, а другие характеризуют изделие в других отношениях.
2. Упорядочение свойств осуществляется в основном путем их классификации (рис. 7.1 и 7.13).
3. Любая техническая система обладает многочисленными свойствами независимо от того, создавались ли они сознательно в соответствии с поставленными требованиями или нет.
4. Разработке технической системы предшествует формулирование требований, определяющих ее желаемые свойства.

5. Свойства технической системы можно классифицировать с различных точек зрения. Важнейшими категориями свойств являются внешние (потребительские) и внутренние (конструктивные) свойства.

6. Внешние свойства зависят от внутренних конструктивных признаков и конструктивных свойств (рис. 7. 13).

8. Оценивание и представление технических систем

8. 1. Выбор критериев и проведение оценки

В предыдущем разделе был сформулирован вопрос: Каков автомобиль? Для ответа на него нужно не только выбрать и оценить определяющие свойства автомобиля, но и объединить их *в обобщенном показателе*. Однако это нелегкая задача, когда свойства различны, например скорость, мощность, оснащение салона автомобиля, внешний вид, прочность и т. д. Только если удастся найти такой обобщенный показатель, можно объективно утверждать, что один автомобиль лучше другого, т. е. обладает лучшими свойствами. В принципе возможны три схемы оценивания; они характеризуются вопросами, приведенными в табл. 8. 1.

Таблица 8.1.

Схемы оценивания		
Схема	Оцениваемый объект	Тип оценочного вопроса
I	Реализованная техническая система	Какова техническая система?
II	Постановка задачи (перечень требований) и вариант решения или образец	Соответствует ли система (модель) данной постановке задачи?
III	Постановка задачи и различные возможные решения, которые технически соответствуют постановке задачи	Какое решение лучшее (оптимальное)?

Оценивание проводится двумя способами: а) *интуитивно*;

б) *объективно*, т. е. на основе *определяющих критериев*.

Интуитивная оценка, несмотря на ее субъективность, не может быть полностью отвергнута. Она определяется не только субъективными ощущениями, но и часто долголетним опытом, поэтому следует систематически развивать навыки интуитивных оценок. Только тогда можно довериться так называемому конструкторскому чутью. Это

особенно важно при недостаточно полной информации, что характерно для начала процесса проектирования.

Оценивание может проводиться по схемам рис. 8.1.

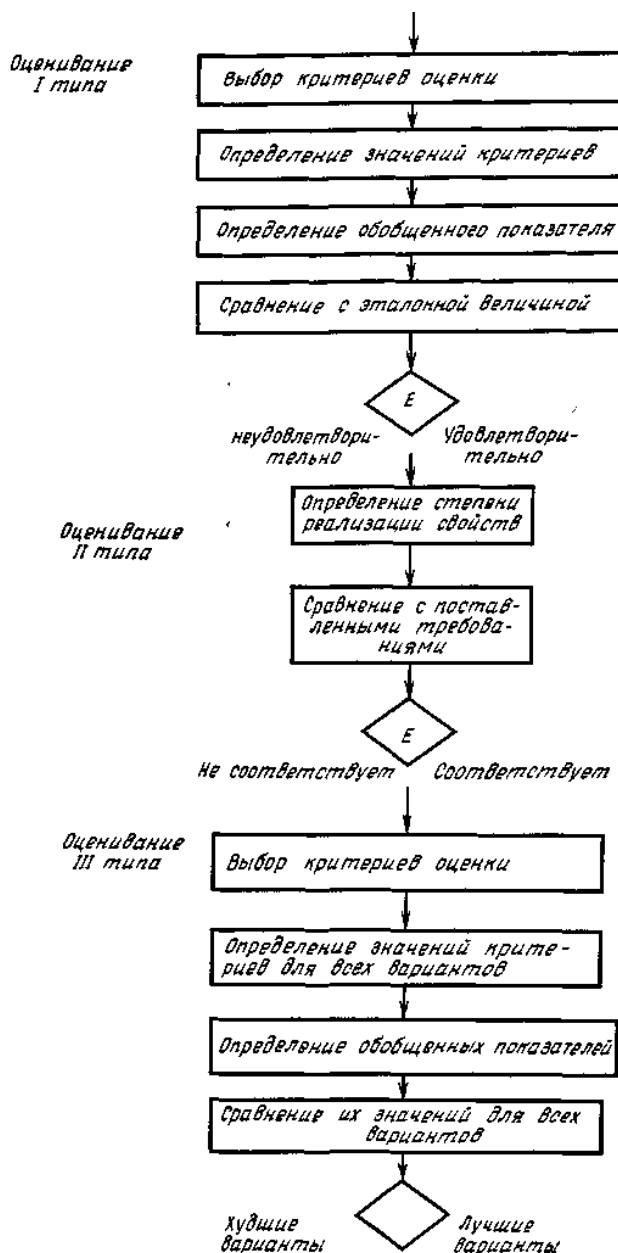


Рис. 8.1. Схемы оценивания.

В процессе оценивания выполняются следующие операции:

а) выбор обобщенного показателя; б) выбор критериев оценки (свойств); в) определение критериальных оценок; г) преобразование оценок в обобщенный показатель.

Определим некоторые понятия и поясним проблематику этих операций.

а) Для объективной оценки должны быть выбраны определяющие критерии. При этом обычно исходят из того, что техническая система обладает ценностью, которую нужно установить. Ценность системы определяется тем, какие потребности она удовлетворяет и в какой мере. В зависимости от точки зрения, с которой оценивается система, возможны различные разновидности ценности.

— *Техническая ценность*, определяемая как совокупность (вектор) качеств технических свойств данной технической системы. Здесь имеются в виду главным образом свойства функционирования, обслуживания и производства. Сюда можно отнести также технологичность и конструктивные свойства.

— *Экономическая ценность* — сумма (вектор) качеств экономических свойств. Аналогичным образом можно определить эргономическую, эстетическую и другие виды ценностей, соответствующие категориям свойств, указанным в табл. 7. 1.

— *Потребительская ценность* охватывает внешние свойства изделия, которые удовлетворяют потребности людей.

— *Совокупная ценность* представляет собой наиболее общий показатель, учитывающий ценности всех классов свойств данного изделия.

Указанные выше показатели можно назвать «абсолютными». Кроме них можно установить целый ряд относительных показателей. Для этого нужно определить «идеальное изделие» как носитель идеальных свойств. Тогда относительным показателем ценности будет отношение действительной ценности к ценности идеального изделия. В качестве относительных можно использовать и другие показатели, такие, как *экономическая эффективность*, определенная в разд. 7. 2.

Кессельринг связывает относительную техническую ценность с относительной экономической ценностью посредством S -диаграммы (S — обобщенная относительная ценность изделия), в которой изделие характеризуется точкой (рис. 8.2).

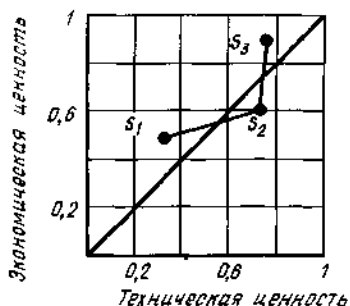


Рис. 8.2. S-диаграмма для определения обобщенной относительной ценности изделия по его вариантам

б) Важным этапом оценивания является выбор критериев, т. е. определяющих свойств системы. Такой выбор должен обеспечивать достаточную полноту рассмотрения системы. Это — общее правило, однако во многих случаях приходится идти на разумные ограничения, так как большое количество выбранных для оценивания свойств приводит к снижению обзорности. Выбор свойств должен проводиться с учетом возможности их количественной оценки. Кроме того, на выбор свойств влияет цель оценки, а также стадия, на которой находится техническая система. Например, техническая система на стадиях эскизного проекта и опытного образца может быть оценена по-разному. Некоторая свобода выбора критериев существует при первом и третьем типах оценивания, тогда как при втором типе выбор определяется заранее сформулированными требованиями (табл. 8.1).

в) Хотя многие свойства (критерии) могут быть оценены количественно, все же возникают трудности, связанные с объединением их в обобщенный показатель, так как различные свойства выражаются в разных единицах. При этом можно использовать денежные или балльные оценки. В случае оценки по баллам реализованная степень свойства может быть оценена определенным количеством баллов, например от 4 (отлично) до 0 (неудовлетворительно) в 5-балльной системе. Такой подход позволяет единообразно оценить разнородные свойства, что очень важно для получения обобщенной оценки. Следует подчеркнуть, что для важных свойств оценка «неудовлетворительно» не допустима. С другой стороны, полная реализация менее важных свойств необязательна.

г) Перед последующей математической обработкой оценок следует решить, достаточно ли для обобщенной оценки арифметическое среднее, или следует предпочесть взвешенное среднее, в котором учтена важность отдельных свойств. В обоих случаях речь идет об одномерном сравнении. Многомерное представление получается лишь при использовании средних геометрических. В этом случае обобщенную оценку можно представить как площадь многоугольника или объем многогранника, линейными размерами которого являются оценки отдельных свойств.

Обобщенную оценку можно определить так же, как векторную сумму — радиус-вектор, компонентами которого являются оценки свойств. В табл. 8. 2 представлены различные способы получения оценок.

Таблица 8.2

Способы обработки балльных оценок с целью определения ценности системы

Вид обобщенной оценки	Формула	Геометрический смысл	Примечание
1. Абсолютное среднее арифметическое	$\bar{p}_1 = \frac{\sum p_i}{n}$		Простой расчет
2. Относительное среднее арифметическое	$\bar{p}_2 = \frac{\sum p_i}{p_{\max}} = \frac{\bar{p}_1}{p_{\max}}$	Среднее значение	Сравнение с идеальным вариантом $\bar{p}_2 \leq 1$
3. Взвешенное абсолютное среднее арифметическое	$\bar{p}_3 = \frac{\sum p_i g_i}{\sum g_i}$		Учитывается важность свойств
4. Взвешенное относительное среднее арифметическое	$\bar{p}_4 = \frac{\sum p_i g_i}{\sum p_{\max} g_i} = \frac{\bar{p}_3}{p_{\max}}$		$\bar{p}_4 \leq 1$
5. Абсолютное среднее геометрическое	$\bar{p}_5 = \sqrt[n]{p_1 \dots p_n}$	Сторона n -мерного «куба», эквивалентного плитке со сторонами $p_1, \dots, p_n$	$p_5 = 0$ при $p_i = 0$
6. Относительное среднее геометрическое	$\bar{p}_6 = \bar{p}_5 / p_{\max}$		$\bar{p}_6 \leq 1$
7. Взвешенное абсолютное среднее геометрическое	$\bar{p}_7 = \frac{\sqrt[n]{p_1 g_1 \dots p_n g_n}}{\sqrt[n]{g_1 \dots g_n}}$		Несущественна, так как g выпадает Как п. 7
8. Взвешенное относительное среднее геометрическое	$\bar{p}_8 = \bar{p}_7 / p_{\max}$		
9. Абсолютный вектор	$\bar{p}_9 = \sqrt{p_1^2 + p_2^2 + \dots + p_n^2}$	Диагонали n -мерной плитки со сторонами $p_1, \dots, p_n$	$\bar{p}_{10} \leq 1$
10. Относительный вектор	$\bar{p}_{10} = \bar{p}_9 / p_{\max}$		
11. Взвешенный абсолютный вектор	$\bar{p}_{11} = \sqrt{(p_1 g_1)^2 + \dots + (p_n g_n)^2}$		
12. Взвешенный относительный вектор	$\bar{p}_{12} = \bar{p}_{11} / p_{\max}$		$\bar{p}_{12} \leq 1$

Обозначения $p_1, \dots, p_n$ — балльные оценки критериев 1, ..., n, $p_{\max}$ — балльная оценка идеального варианта, $g_1, \dots, g_n$ — значения (веса) критериев оценки, $\bar{p}$ — обобщенная оценка, n — количество критериев оценки, $\sum_{i=1}^n p_i$ — сумма баллов оценок от p_1 до p_n

Выводы

1. Обобщенная оценка совокупной ценности технической системы определяется суммой оценок частных ценностей (свойств).
2. Для оценки свойств системы необходимо выбрать наиболее объективную единицу измерений.
3. Выбранные для оценки определяющие свойства называются критериями. Выбор критериев является важнейшим этапом оценивания, обеспечивающим сопоставимость и полноту оценок.
4. Преобразование оценок отдельных критериев в обобщенную оценку можно проводить различными методами. Метод такого

преобразования представляет собой один из важнейших факторов объективности оценки.

5. На некоторых стадиях становления *TS* трудную проблему представляет правильное определение ценностей свойств. Объективное оценивание ценностей способствует лучшему качеству обобщенной оценки.

8.2. Представление технических систем

Теория представления технических систем подробно рассматривается в научной теории конструирования. Полученные путем того или иного представления модели описывают отдельные конструктивные свойства технической системы и вследствие этого сами являются системами типа «объект» .

Существует много видов и способов представления систем. Важными факторами при выборе вида и способа представления являются собственно объект и его состояние, а также назначение представления (модели) — сообщение, испытания, расчет, анализ, фиксация для памяти и т.п.

Так как создание и изучение моделей зачастую отнимает много времени, их рационализации уделяется большое внимание. С применением вычислительной техники в этой области произошли коренные изменения, но в свою очередь возникли и новые проблемы, например проблемы моделирования с использованием ЭВМ, которое мы рассмотрим ниже .

1. Способы представления технических систем

Выше были проанализированы технические системы, их структуры и свойства. При этом рассматривались различные способы представления технических систем в первую очередь в зависимости от степени абстрактности. Попытаемся упорядоченно описать их и увязать с нормативными документами. В качестве критерия упорядочения будем использовать степень абстрактности или противоположную ей степень полноты описания (рис. 5.9). В качестве иллюстрации на рис.8.3. перечислены различные типы конструктивных документов (от технических требований до сборочного чертежа) и стрелками указано на соответствующие изображения. Заштрихованные участки в правой половине таблицы характеризуют информативность соответствующего документа.

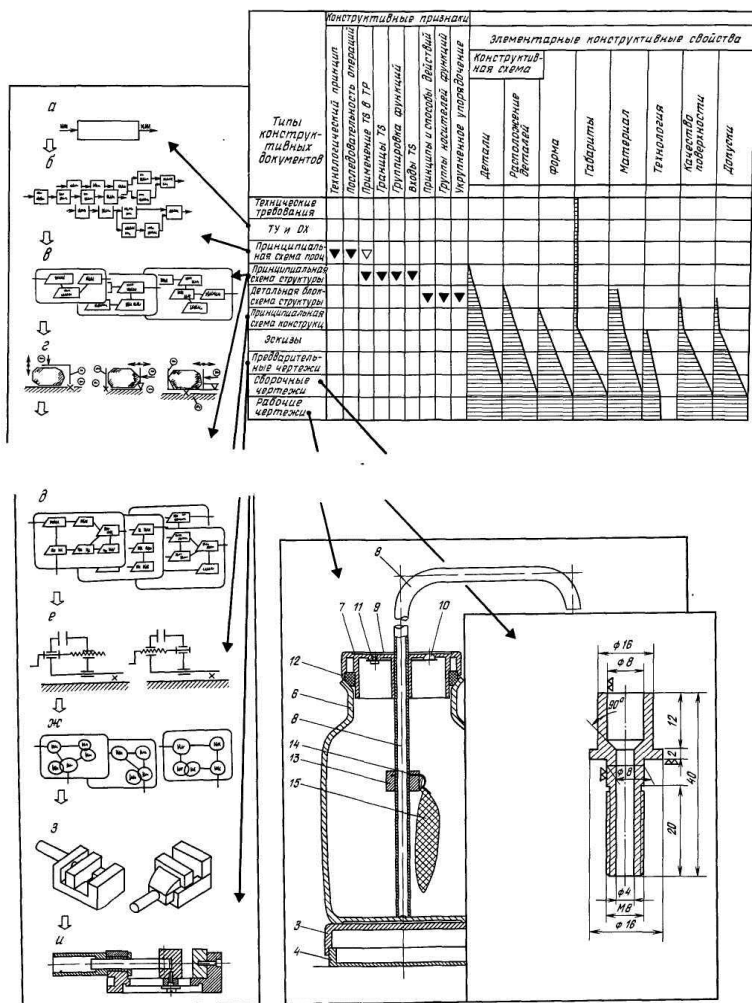


Рис. 8.3. Способы изображения технических систем и содержащаяся в них информация
 а — процесс типа «черный ящик», б — альтернативные структуры процесса, в — укрупненные функциональные структуры, г — альтернативные принципы, макроструктура, д — альтернативные функциональные структуры, е — альтернативные принципы действия ж — альтернативные принципиальные схемы, з — эскизы, и — чертёж

Здесь мы возвращаемся к понятиям конструктивных признаков и свойств, которые выступают в качестве средства описания технических систем. Показанные на рис.8.3 способы представления, конечно, не исчерпывают всех возможностей представления технических систем. Ниже мы проиллюстрируем некоторые способы, приобретающие особое значение в связи с применением ЭВМ.

На рис. 8.4 показаны конструкция шестерня — вал и ее изображение с помощью графа .

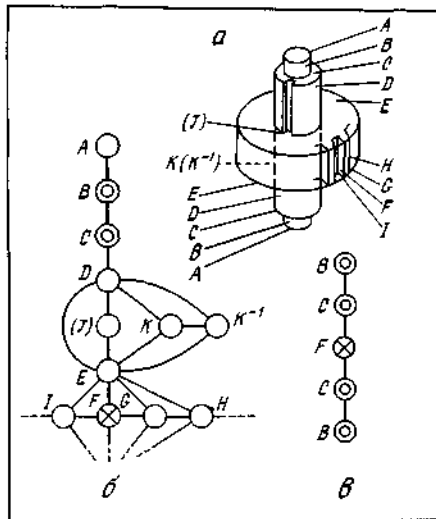


Рис. 8.4. Изображение структуры технической системы при помощи графа (на примере узла шестерня — вал)
 а — узел шестерня-вал б — граф конструктивных элементов и их отношений, в - граф функций

Узел (точка) графа обозначает конструктивный элемент, а отрезок (стрелка) — отношение. Соединение двух точек (двуместное отношение) может быть направленным (стрелка) или ненаправленным. Другое изображение рассматриваемой конструкции — это матрица связей, которая описывает элементы в их функциональном взаимодействии .

Изображения в матричной форме и с помощью графов могут быть преобразованы одно в другое.

2. Представление конструктивных элементов

Конструктивные элементы, являющиеся техническими системами низшего уровня сложности, нередко, особенно в машиностроении, очень многообразны, преимущественно по форме и размерам. Традиционные графические методы мы рассматривать не будем, поскольку они хорошо известны. Упомянем лишь методы, пригодные для работы с ЭВМ. Это вновь акцентирует внимание на системном подходе. С точки зрения геометрии тело можно рассматривать как систему точек, ребер, поверхностей или элементарных форм. Исходя из этого, могут быть рассмотрены следующие модели:

- а) *линейная модель*, которая описывает конструктивный элемент с помощью линий и точек (как в черчении). Эта модель не позволяет удовлетворительно передать форму, так как она неоднозначна, не создает пространственного эффекта и допускает бессмысленные формы;
- б) *плоскостная модель*, представляющая собой систему поверхностей детали. Поверхностями можно однозначно описать заготовку;
- в) *объемная модель* — упорядоченная совокупность элементарных тел соответствующей формы. Такое модельное представление геометрически в точности соответствует действительности и позволяет воссоздать разнообразные конструктивные элементы посредством различных комбинаций элементарных тел. На рис. 8.5 показаны важнейшие элементарные тела и построенный из них сложный конструктивный элемент

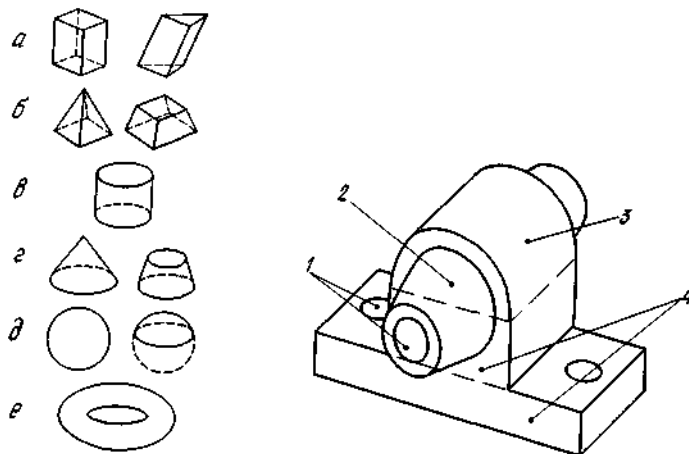


Рис. 8.5. Комбинирование элементарных тел с целью построения объемной модели

а — прямоугольная и наклонная призмы *б* — пирамида и усеченная пирамида, *в* — цилиндр, *г* — конус и усеченный конус, *д* — шар и шаровой сегмент, *е* — специальные тела вращения (тор). 1 — отверстие 2 — усеченный конус, 3 — круговой полуцилиндр, 4 — призмы

Выводы

1. Представления технической системы — это ее модели, в которых учитываются те или иные конструктивные свойства (конструктивные признаки).
2. В зависимости от вида, количества и степени учета конструктивных свойств представления могут различаться степенью абстрактности (степенью детализации).
3. Существующими способами и приемами могут быть построены любые модели любых технических систем.
4. Представление технической системы обычно служит определенной цели. В зависимости от этой цели и выбирается способ представления рассматриваемой технической системы.

8.3. Этапы создания и использования технической системы

Насколько сложна сама техническая система, настолько сложным и длительным является процесс ее создания. Как для планирования разработки, так и для последующего использования системы важно

знать в подробностях весь «цикл жизни» системы и влияющие на него факторы. Конструктор принимает участие (хотя бы как консультант) на всех стадиях создания системы.

Весь «цикл жизни» технической системы состоит из четырех этапов: создание, перемещение, использование по назначению и ликвидация.

Каждый из этих этапов содержит целый ряд стадий, операций и приемов. Ниже будут рассмотрены содержание, отношения и последовательность соответствующих процессов.

Блок-схема (рис.8.6) отображает посредством элементарных процессов этапы создания и использования технической системы.

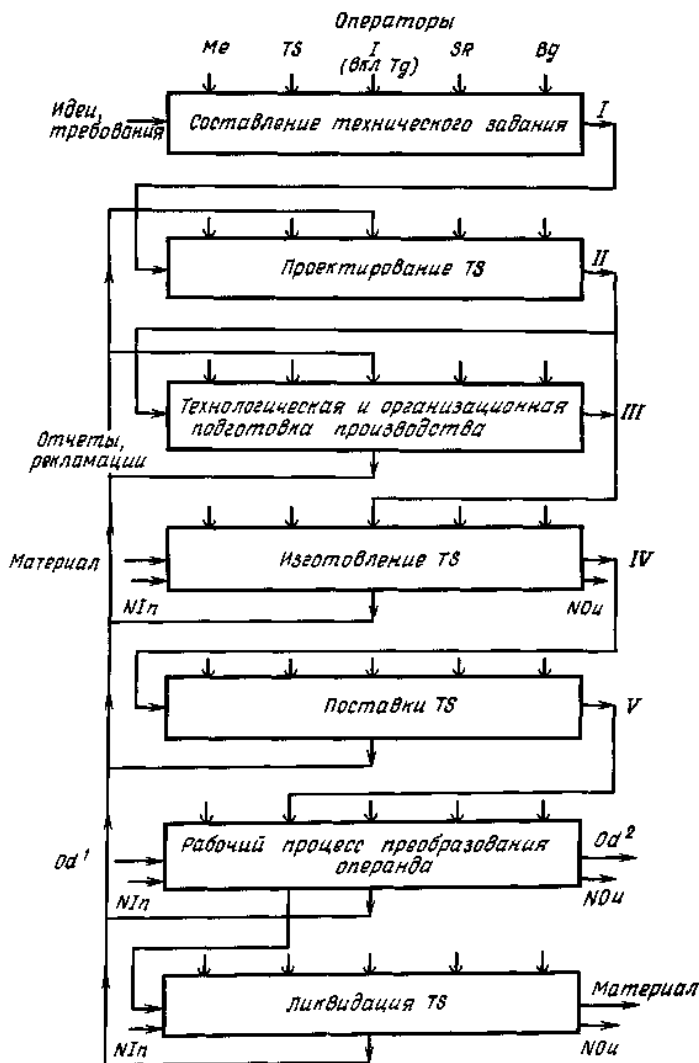


Рис. 8.6. Модель цикла «жизни» технической системы.

I — технические требования, отчет по исследованиям, технико-экономическое обоснование, II — описание ТС, схема, перечень составных частей, расчеты, III — планы-графики, технологические карты, оборудование, IV — хранение ТС и подготовка к поставкам, V — монтаж и отладка ТС у заказчика на месте работы.

Блок-схема и используемые в ней символы базируются на модели типа «процесс», описанной ранее.

При более подробном обсуждении мы будем учитывать влияние уровня сложности технической системы (см. табл. 6.2) и количества изготавливаемых экземпляров, что приводит к необходимости разделить технические системы на три категории, а именно:

1. Технические системы до третьего уровня сложности серийного производства.
2. Технические системы до третьего уровня сложности единичного изготовления.
3. Технические системы четвертого уровня сложности (предприятия).

Целесообразно сначала дать общую характеристику отдельных процессов и связей между ними. Более полное определение содержания процессов и включение их в общую структуру на начальной стадии было бы затруднительным и нецелесообразным.

На стадии подготовки к созданию технической системы устанавливается принципиальная возможность и целесообразность создания технической системы и составляется техническое задание для процесса проектирования. Содержание процесса подготовки сильно зависит от вида и степени сложности технической системы, а также предъявляемых к ней требований (см. рис. 7.15).

Наиболее важная часть процесса подготовки — поисковое исследование. Целью его является оценка реализуемости. В принципе, могут проводиться поисковые исследования двух видов:

- 1) исследования преобразований в рабочем процессе; например, при подготовке к созданию станка должны проводиться исследования по обработке материалов, а при подготовке к созданию транспортера — исследования характеристик мощности и грузоподъемности;
- 2) исследования с целью улучшения свойств технической системы; при этом анализируются соответствующие средства в отношении принципов работы, структуры, используемых материалов, формы, качества поверхности, срока службы и т.д.

При проведении прикладных исследований такого рода используются результаты фундаментальных исследований и практический опыт применения аналогичных систем. Важным элементом исследований является эксперимент. В рамках поисковых исследований иногда организуется опытное производство.

Результатом подготовительного процесса является постановка задачи, т.е. перечень технических требований к технической системе.

Стадия проектирования охватывает все операции конструкторских работ, от предварительных эскизов до детализированных чертежей и технических условий. Выходные параметры стадии проектирования становятся исходными данными для стадии подготовки производства, включающей как технические, так и организационные мероприятия (выбор технологии, оборудования, организация материально-технического снабжения, планирование процесса производства во времени, распределение работ и т.д.).

На стадии изготовления материалы подвергаются обработке в соответствии с технической информацией, полученной на предыдущих стадиях, и материализуется, наконец, сконструированная ранее техническая система.

На этапе перемещения изменяются принадлежность технической системы (производитель — потребитель) и ее местоположение (место производства—место использования). Типичными стадиями на этом этапе являются хранение на складе, реклама, упаковка, транспортировка, монтаж.

Техническая система используется в производственном процессе для преобразования операндов с целью получения желаемых выходных характеристик. Эти преобразования являются главной целью всей деятельности по созданию и использованию технической системы. В процессе использования техническую систему следует содержать в исправности и модернизировать. Наконец, вследствие морального или физического износа в процессе использования техническая система перестает отвечать предъявляемым требованиям и ликвидируется.

1. Стадии создания технических систем серийного производства

В табл. 8.3. проведено разделение этапов создания и использования технической системы на стадии; эти стадии можно представить в виде блок-схемы (рис.8.7).

Таблица 8.3.

Содержание этапов создания и использования технических систем серийного производства

Стадия	Основные процессы стадии	Основные входные данные	Результаты
Фундаментальные исследования	Сбор и анализ научной информации, формулировка идей, построение моделей, составление отчетов	Научная информация, техническая потребность	Отсчет о результатах исследований, научные статьи
Технические исследования	Обработка научно-технической информации, анализ технического задания, моделирование, обработка экспериментальных данных	Научно-техническая информация, техническое задание	Отчет о результатах исследований, отчет по результатам экспериментов
Экономические исследования	Изучение спроса, оценка конкурентоспособности, перспективы экономической конъюнктуры	Статистические исследования, проспекты фирм, отчеты	Отчет с оценками экономической ситуации и возможностей сбыта
Решение о разработке	Оценки реализуемости, обработка информации, оценка технических возможностей, принятие решения	Доклад о производственных возможностях, патентный обзор	Отчет с обоснованием принятого решения
Концептуальное исследование	Уточнение технических требований, формулировка задачи, определение концепции через принципы работы, функции и средства, предварительные расчеты и оценка	Постановка задачи, технические требования, анализ патентной ситуации	Описание концепций, расчеты, принципиальная схема, отчет
Эскизное проектирование	Определение конструктивной схемы, компоновка, предварительные расчеты, определение ограничений, сроки	Результаты проработок концепций, чертежи известных аналогичных систем, техническая информация	Расчеты, эскизные чертежи, отчет
Техническое проектирование	Расчеты, проработка узлов, определение материалов и методов изготовления	Проектные чертежи, перечень материалов, список производственного оборудования, технические условия, нормы	Комплект чертежей и инструкций
Подготовка к изготовлению экспериментального образца Изготовление	Разработка технологии, выбор средств изготовления, обеспечение материалами, инструментами и оснасткой Подготовка материа-	Комплект чертежей и инструкций, технические условия на материалы и работы Комплект чер-	Технологические карты, инструкции по изготовлению, сборке и испытаниям Эксперименталь-

Стадия	Основные процессы стадии	Основные входные данные	Результаты
ние экспериментального образца Испытания экспериментального образца	лов и заготовок, изготовление деталей, сборка, контроль Подготовка стенда и образца к испытаниям, обеспечение безопасности, испытания, оценка результатов, составление отчета	тежей, технологические карты, инструкции Инструкции по проведению испытаний, обработке и оценке результатов, инструкции и нормы по технике безопасности Все данные	ный образец, отчет по подготовке производства Отчет по результатам испытаний
Решение о серийном производстве Корректировка конструкторской и технологической документации Подготовка серийного производства	Анализ имеющейся технической, экономической и производственной информации Обоснование корректировки, изменение чертежей и инструкций Организация производства и материально-технического снабжения, подготовка оборудования и технологической документации, планирование хранения и сбыта продукции	Данные по изготовлению и испытаниям экспериментального образца Технологические и производственные данные	Решение Чертежи, технология и инструкции для серийного производства Отчеты, рекомендации и организационные мероприятия по подготовке серийного производства
Изготовление образцов установочной серии Испытания образцов установочной серии	Подготовка материалов и заготовок, изготовление деталей, сборка, контроль Подготовка стендов и образцов к испытаниям, обеспечение безопасности, испытания, оценка результатов, составление отчета	Технология и инструкции для серийного производства Отчет по испытаниям экспериментального образца, инструкции по проведению испытаний и технике безопасности	Изготовление установочной серии, производственный отчет Отчет о результатах испытаний и оценка
Заключительная корректировка конструкторской и технологической документации	Корректировка чертежей, инструкций и технологической документации и организации производства, проверка готовности средств производства и материально-технического снабжения	Производственные отчеты и отчеты об испытаниях	Комплект производственной документации
Серийное производство	Подготовка материалов и заготовок, изгото-	Уточненная конструкторская и	Серийное производство техни-

Стадия	Основные процессы стадии	Основные входные данные	Результаты
Процессы перемещения	товление деталей, сборка, контроль	технологическая документация для серийного производства	ческой системы
Использование технической системы	Хранение, продажа, упаковка, транспортировка, техническое обслуживание	Заказы, инструкции	Поставки и отладка изготовленных технических систем
Ликвидация технической системы	Рабочий процесс, обслуживание, ремонт, модернизация	Чертежи, инструкции по использованию	Рекламации
	Демонтаж, отправка в металлолом либо подготовка для другого использования	Чертежи, инструкции и отчеты	Вторсырье, другое использование

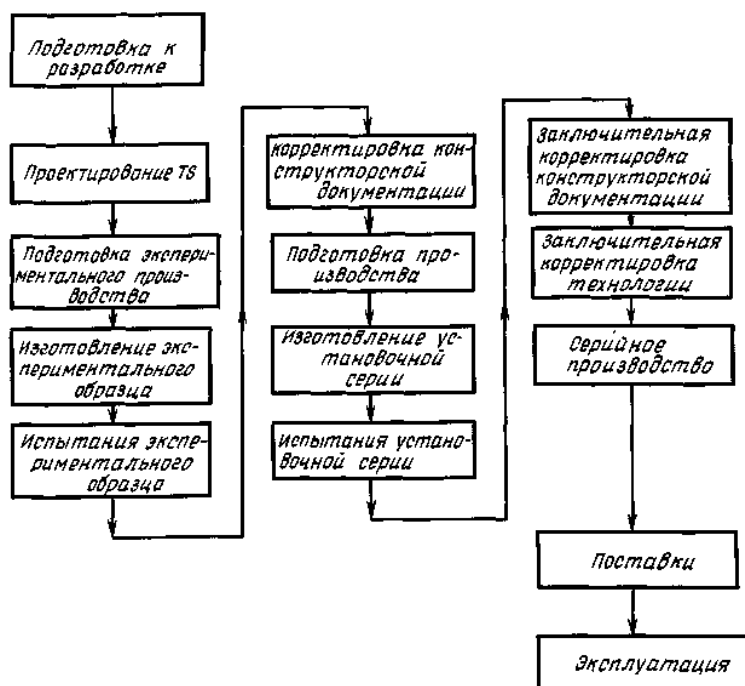


Рис. 8.7. Упрощенная блок-схема стадий создания и использования технической системы серийного производства.

Первые стадии уже были рассмотрены; теперь мы перейдем к стадии *проектирования*.

Стадия проектирования. В большинстве случаев техническая система проектируется опытным инженером-конструктором и его группой. На стадии проектирования должны быть разработаны следующие документы:

- отчет о результатах или о состоянии исследований по этому заданию;
- проект конструкции (общий вид) готового изделия и его важнейших узлов;
- функциональная и кинематическая схемы;
- отчет с технико-экономическим обоснованием для принятия решения о производстве;
- обзор патентных поисков с целью выяснения, не нарушены ли в разработанном проекте чьи-либо авторские права;
- указания на то, каким образом выполнены поставленные требования, и предложения о возможностях улучшения свойств разработанной технической системы;
- предварительный анализ экономической эффективности разработанной технической системы.

Выполненный проект должен быть проверен и утвержден с учетом замечаний специальной комиссии. Поскольку проект определяет большинство свойств будущей технической системы, его разработке, проверке и оценке следует уделить максимум внимания.

После утверждения проекта начинается выполнение рабочих чертежей технической системы, по которым должен быть изготовлен экспериментальный образец.

Подготовка экспериментального производства. Наряду с выполнением чертежей подготавливается экспериментальное производство. Подготовку производства экспериментального образца необходимо провести таким образом, чтобы он мог быть изготовлен с минимумом подготовительных работ и специальных инструментов.

Изготовление экспериментального образца. На стадии изготовления экспериментального образца необходимо еще раз оценить свойства технической системы, связанные с изготовлением, прежде всего свойства, касающиеся монтажных операций. Так как конструктор лучше других специалистов знает особенности своей конструкции, необходимо, чтобы он сам (или в крайнем случае его представитель) следил за процессом изготовления. Это поможет своевременно выявить все возможные ошибки и трудности и внести необходимые коррективы в техническую документацию.

Испытания экспериментального образца. Испытания являются решающим фактором в определении ценности разработанной технической системы. Испытания также очень важны в отношении накопления опыта, особенно для молодых конструкторов. Имея результаты испытаний, они могут сравнивать свои замыслы с реальностью, приобретать опыт, а вместе с ним — уверенность в себе и в своих силах, учиться на своих и чужих ошибках. Испытания экспериментального образца должны быть проведены под максимальной нагрузкой и в наиболее сложных из ожидаемых условий эксплуатации. В испытаниях должно быть оценено функционирование системы, должны быть выяснены функционально обусловленные, эксплуатационные и по возможности все другие свойства технической системы, в том числе пригодность к обслуживанию и ремонту. Ход и результаты испытаний должны быть изложены систематически и полно в отчете; особенно это относится к выявленным недостаткам и рекомендациям по улучшению экспериментального образца. С учетом результатов испытаний следует уточнить инструкции по защите от коррозии, упаковке, транспортировке, монтажу, вводу в действие, эксплуатации, обслуживанию и другую техническую документацию. На основании испытаний и оценки экспериментального образца принимается *решение о серийном производстве*. Подготовка к изготовлению экспериментального образца, его изготовление и испытания предоставляют необходимые данные для принятия такого решения. Эти данные должны быть дополнены сведениями по эргономическим, эстетическим и манипуляционным свойствам, а также оценками в отношении действующих правовых норм и инструкций. Анализ указанных данных позволяет с достаточной уверенностью судить об экономических свойствах разработанной технической системы. С учетом анализа возможностей сбыта, проведенного на новом уровне, делается заключение о том, будет ли успешным производство разработанной технической системы в имеющихся условиях производства и рынка. Иногда может оказаться рациональнее прекратить дальнейшую разработку во избежание еще больших убытков.

Корректировка конструкторской документации. После испытаний экспериментального образца осуществляется стадия корректировки конструкторской документации для серийного или массового производства на основании приобретенного опыта с целью усовершенствования конструкции и технологии изготовления разработанной технической системы. Эта работа очень важна, хотя и малоинтересна. Для того чтобы такая корректировка была наиболее эффективной, следует подвергнуть анализу все недостатки,

обнаруженные в экспериментальном образце, установить их причины и внести в конструкцию соответствующие усовершенствования.

Подготовка производства. Подготовка серийного производства охватывает, кроме организации производства и материально-технического снабжения, создание специальных станков и технологической оснастки, проработку вопросов хранения, транспортировки и сбыта продукции, взаимоотношений со смежниками и т. д. Этой стадии необходимо уделить в организационном отношении большое внимание. Это особенно важно для конструктора вспомогательного оборудования, работа которого всегда связана с жесткими ограничениями по срокам.

Производство. Производством завершается этап создания технической системы. Первая стадия серийного производства — это производство образцов установочной серии, представляющей собой своего рода «генеральную репетицию» отлаженного производственного процесса.

Изготовление образцов установочной серии предназначается главным образом для того, чтобы надежнее оценить свойства разработанной системы и еще раз испытать пригодность соответствующего оборудования для выполнения задачи серийного или массового производства, что составляет содержание двух следующих стадий — **заключительной корректировки конструкторской и технологической документации.** Следует напомнить, что при серийном и массовом производстве всегда стремятся к рационализации процесса производства с целью его облегчения и удешевления. Это особенно важно при сборке и монтаже, где используется в основном ручной труд. Механизация этих работ требует от конструктора хорошего знания технологии. Поэтому конструктор технической системы и особенно конструктор производственного оборудования должны хорошо знать состояние и возможности производства. Последующие этапы перемещения, использования и ликвидации уже были рассмотрены. Только постоянный контроль со стороны конструктора всех стадий создания и использования (см. обратные связи на рис. 8.6) позволяет реализовать все возможности, заложенные в конструкции технической системы.

2. Стадии создания технических систем единичного изготовления

Технические системы единичного изготовления обычно велики по масштабам или имеют уникальные свойства. Такие технические системы предъявляют повышенные требования к конструкторскому процессу. Концепция и конструктивная схема разработанной системы обязательно должны быть удачными. Если требуемые свойства

технической системы получить сразу не удастся, то доработки должны быть по возможности незначительными, с тем чтобы не затрагивать концепцию и конструктивную схему системы.

Часто функциональные испытания такой технической системы проводятся на месте монтажа у пользователя, а у изготовителя могут проверяться только подфункции системы и функции ее узлов.

Особняком стоят технические системы, которые должны удовлетворять специальным требованиям заказчика, а их функционирование и другие свойства уже известны и не включают в себе для изготовления никакого риска. Тем не менее изготовитель должен проявить внимание к конструкторским доработкам и тщательность в производстве, чтобы избежать неприятных неожиданностей.

Создание технической системы единичного изготовления проходит через следующие основные стадии: *эскизное проектирование, техническое проектирование, подготовка производства, производство, монтаж и отладка.*

При проектировании руководствуются характером будущего изделия. Разрабатывая техническую систему единичного изготовления, конструктор должен тесно сотрудничать с администрацией, экономистами и технологами в целях минимизации затрат, которые в случае нерациональных решений могут оказаться неприемлемыми.

Чтобы повысить надежность ожидаемого функционирования технической системы, целесообразно, если только это возможно, предварительно подвергнуть испытаниям функции новых и технически важных частей системы. Вся документация системы, выполняемой по специальному заказу, должна быть тщательно проверена и откорректирована с учетом требований заказчика.

Испытания технической системы единичного изготовления осуществляются в согласованных с заказчиком условиях. Весь опыт разработки, обнаруженные недостатки, меры по их устранению и предложения по улучшению характеристик должны быть зарегистрированы в документе об изменениях, а также учтены в модификациях конструкторской документации с целью учета их при возможном повторном производстве.

При эксплуатации технической системы все ее свойства постоянно подвергаются проверке. При этом конструктор должен в течение долгого времени следить за использованием разработанной им технической системы, так как он лучше других специалистов может оценить ее достоинства и недостатки и дать необходимые рекомендации. В процессе эксплуатации выявляются скрытые дефекты, такие, как преждевременный износ отдельных деталей. Контакт конструктора с пользователем очень важен для обеих сторон в

отношении приобретения опыта. Квалифицированное обслуживание системы имеет особое значение для достижения максимального экономического эффекта.

Рекомендуется по каждой поставляемой технической системе вести книгу учета дефектов и усовершенствований, в которой отмечался бы весь опыт создания и использования системы. Следует отметить, что крайне нежелательно форсировать создание технической системы за счет сокращения или исключения отдельных стадий; обычно это ведет, наоборот, к замедлению и удорожанию разработки. Это не относится к рациональной организации работ, включающей заимствование опыта и параллельное проведение некоторых операций.

3. Стадии создания технических систем четвертого уровня сложности

Особый случай по сравнению с рассмотренными выше техническими системами представляет, например, создание технологической линии, цеха или производственного комплекса. Создателем таких систем является инженер-проектировщик. Абстрактной моделью системы служит проект, выполненный в соответствии с требованиями заказчика. Технические системы четвертого уровня сложности имеют в своем составе готовые технические (машинные) системы более низких уровней сложности, которые закупаются или изготавливаются на заказ. Такой процесс проектирования протекает несколько по-иному. Главная задача проектировщика состоит в создании функциональных структур с использованием готовых элементов и расположении проектируемой системы в пространстве. Так как проектирование идет на уровне структурных элементов, которые могут располагаться в пространстве более или менее независимо друг от друга, то наряду с рабочей функцией большую роль играет функция связи. Входящие в состав проектируемой системы машины и их связи принадлежат различным специальным областям машиностроения, электроники, химической технологии и др. Поэтому инженер-проектировщик такой системы работает в тесном контакте со специалистами в соответствующих областях. Совместная работа различных специалистов в соответствующих проектных организациях тщательно регламентируется.

Проектные работы по техническим системам четвертого уровня сложности должны проводиться как минимум в две фазы и периодически подвергаться экспертизе. Иногда двух фаз недостаточно, так как предварительная концепция проекта может развиваться и детализироваться. Так, проект производственного цеха обычно

включает подпроекты здания, машинного оборудования, электрооборудования, связи, теплоснабжения, канализации и т. д. Можно указать следующие основные стадии создания и использования технических систем четвертого уровня сложности:

Проектирование. а) *Предпроектные проработки и технико-экономическое обоснование.* На этих стадиях конкретизируется постановка задачи, выдвигаются проектные идеи и делаются наброски основополагающих решений; технико-экономическое обоснование служит для принятия решения о реализации.

б) *Технический проект* включает пояснительную записку, схемы основных технических решений и спецификацию основного оборудования.

в) *Рабочая документация* охватывает все технологические и конструкторские схемы и чертежи подсистем, узлов и системы в целом.

Строительно-монтажные работы. На этой стадии ведутся поставки оборудования, строительство и монтаж на основании проектной документации и действующих норм и правил.

Пусконаладочные работы. На этой стадии, как при испытаниях обычной технической системы, должны быть продемонстрированы требуемые свойства технической системы (например, построенного промышленного комплекса).

Эксплуатация. После достижения проектных показателей техническая система вступает в период нормальной эксплуатации.

4. Временная последовательность стадий создания и использования технической системы

До сих пор стадии и операции этапов создания и использования технической системы описывались с точки зрения их содержания и взаимных отношений. В данном разделе рассматривается их временная последовательность с целью выявления тех факторов, которые влияют на длительность процессов создания и использования системы. Начав заниматься этой задачей, мы сразу же обнаружим противоречие. Как заказчик, так и изготовитель желают по возможности уменьшить время создания технической системы, поскольку при этом повышается экономическая эффективность предпринятой разработки. С другой стороны, заказчик заинтересован в высоком качестве изделия, что связано, однако, с затратами времени и средств. (Эта зависимость показана на рис. 7.17.) Аналогичная ситуация характерна для подготовки к производству и самого производства. Однако сравнив процессы создания технических систем серийного (рис.8.8) и единичного

(рис.8.9) производства, можно заметить, что в последнем случае ряд стадий отсутствует.

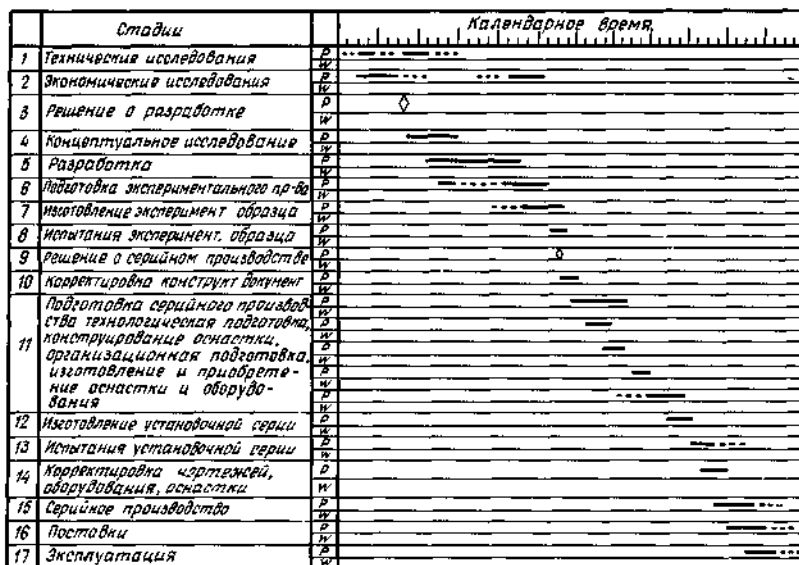


Рис. 8.8. Упрощенный план-график стадий создания технической системы серийного производства

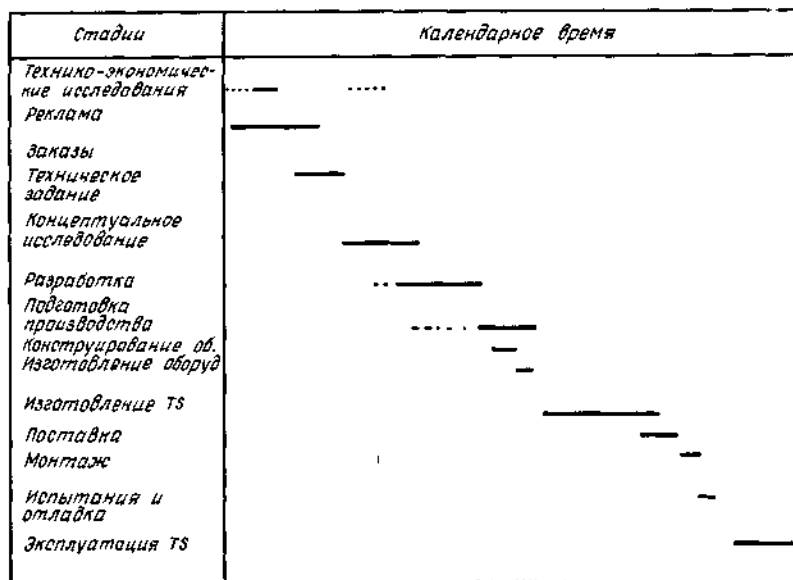


Рис. 8.9. Упрощенный план-график стадий создания технической системы единичного изготовления.

Тем не менее процесс единичного изготовления более длителен, так как некоторые стадии не имеют стандартного обеспечения. Значительную долю времени создания таких систем занимают монтаж и пусконаладочные работы как наиболее трудоемкие и ответственные стадии.

Понятно, что зависимость начала последующей операции от конца предыдущей увеличивает продолжительность соответствующей стадии. Желаемое сокращение времени может быть достигнуто, с одной стороны, путем уменьшения длительности отдельных стадий и операций, а с другой - путем их параллельного проведения, т. е. совмещения во времени.

Снижение длительности операций возможно главным образом на продолжительных стадиях, например при проектировании, подготовке к производству и самом производстве. Наибольшая экономия времени получается при пропуске целой стадии. Примерами таких возможностей могут служить приобретение лицензии, благодаря чему выпадает стадия проектирования, или закупка готовых подсистем и узлов, благодаря чему выпадают стадии подготовки к их производству и собственно производства,

а также сокращается стадия конструирования. Длительность стадии проектирования зависит, например, от объема работ (см. рис. 7.17), а также от квалификации конструкторов (рис.8.10).

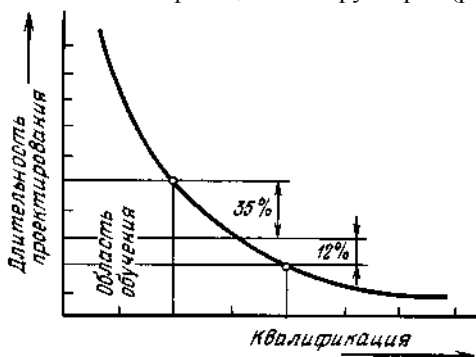


Рис. 8.10. Взаимозависимость между конструкторскими возможностями и временем, необходимым для производства машинной системы.

Влияние других факторов на длительность процесса проектирования (хотя бы в общих чертах) уже рассматривалось. Аналогичные факторы влияют на длительность процесса производства. Для известного технического уровня производства (решающего фактора) идеальной, т. е. самой короткой длительностью производства будет сумма самых продолжительных (но оптимизированных по продолжительности) последовательных операций. Схема такого оптимального по времени процесса производства изображена на рис. 8.11.

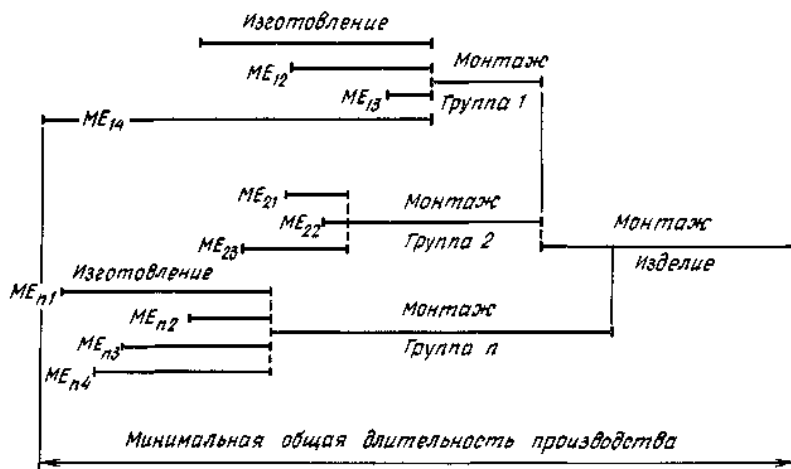


Рис. 8.11. Идеальная длительность производства изделия при данной технологии (ME_{ij} — машинные элементы).

Действительная продолжительность производства обычно оказывается больше вследствие возникающих в процессе производства затруднений и неувязок. При планировании процессов производства во времени широко используются диаграммы Ганта и методы сетевого планирования. Совмещение операций во времени возможно лишь тогда, когда это позволяют взаимосвязи между ними. Параллельные работы обычно приходится вести в условиях недостатка информации. Это может иметь негативные последствия, например, решение, принятое в условиях недостатка информации, может оказаться неоптимальным, а когда это выяснится, изменить его станет невозможно.

Полезный период существования технической системы, т. е. продолжительность ее использования, зависит от ценности (технического уровня) системы, от износа и, кроме того, от появления новой, более совершенной системы такого же назначения, т. е. от темпов технического прогресса. Поэтому, несмотря на естественное желание потребителя как можно дольше использовать привычную техническую систему, потребность в более эффективной системе оказывается сильнее.

5. Распределение стадий и операций между исполнителями

Описанные стадии и операции необходимо тщательно координировать и контролировать. На предприятии эти работы обычно распределяются

между различными отделами и исполнителями: В принципе каждый отдел должен нести полную ответственность за проведение определенных работ. Пример того, как могут быть распределены работы между различными подразделениями проектно-конструкторской организации, приведен в табл.8.4.

Таблица 8.4. Примерное распределение работ по созданию технической системы между подразделениями проектно-конструкторской организации

Стадии работ	Подразделения							организации					
	1	2	3	4	5	6	7	8	9	10	11	12	13
	Планный отдел	Исследовательский отдел	Расчетный отдел	Конструкторский отдел	Торгово-экономический отдел	Производственный отдел	Экспериментальное производство	Испытательный отдел	Конструкторско-технологический отдел	Производство	Отдел материально-технического снабжения	Цех оснастки	Руководство
0 Получение технического задания	+		×										
1 Технические исследования	×	+	×										×
2 Экономические исследования	×		×		+								
3 Решение о разработке	×	×	×		×						×		+
4 Концептуальное исследование		×	+	×									
5 Проектирование				+									
6 Подготовка экспериментального производства			×			+							
7 Изготовление экспериментального образца			×	×		×	+						
8 Испытания экспериментального образца		×	×	×			×	+					
9 Решение о серийном производстве	×		×	×	×	×	×	×		×	×		+
10 Корректировка конструкторской документации			×	+									
11 Подготовка серийного производства													
— технологическая				×		+							
— конструирование оснастки				×		×			+				
— изготовление оснастки								×			+		

Стадии работ	Подразделения организации												
	1	2	3	4	5	6	7	8	9	10	11	12	13
	Плановый отдел	Исследовательский отдел	Расчетный отдел	Конструкторский отдел	Торгово экономический отдел	Производственный отдел	Экспериментальное производство	Испытательный отдел	Конструкторско технологический отдел	Производство	Отдел материально технического снабжения	Цех осистки	Руководство
— организационная подготовка	+										×		
12 Производство установочной серии						×				+			
13 Испытания установочной серии			×	×				+					
14 Корректировка: — чертежи			×	+					+				
— оснастка												+	
15 Серийное производство						×				+			
16 Поставка			×	×	+			+		×			
17 Монтаж на месте			×	×				+		×			
18 Отладка			×	×						×			
19 Эксплуатация				×	+								
20 Обслуживание				×	+								
21 Модернизация				×	+								
													Заказчик

Обозначения + подразделение, ответственное за выполнение стадии × — подразделение, принимающее участие в работах, — возможное участие заказчика

Выводы

- 1.«Цикл жизни» технической системы можно разделить на этапы, стадии и операции.
2. Четырьмя основными этапами «цикла жизни» технической системы являются создание, перемещение, использование и ликвидация.
3. Техническая система на каждом этапе должна удовлетворить определенным требованиям, т. е. обладать определенными свойствами.
4. Количество и продолжительность стадий, на которые подразделяются этапы «жизни» системы, зависят от уровня сложности системы, оригинальности конструкции, способа производства и требований заказчика. В зависимости от проектно-кон-структорской организации и ее традиций некоторые стадии могут быть определены самым различным образом.

5. Совокупная ценность технической системы определяется главным образом на стадии конструирования.

8.4. Развитие технических систем

1. Закономерности развития технических систем

Повышение технического уровня изделий во времени

Сравнив между собой аналогичные технические системы, созданные в разное время, легко обнаружить определенные тенденции в изменениях. Характерным примером является легковой автомобиль, история развития конструкции которого представлена в виде иллюстраций рис. 8.12.

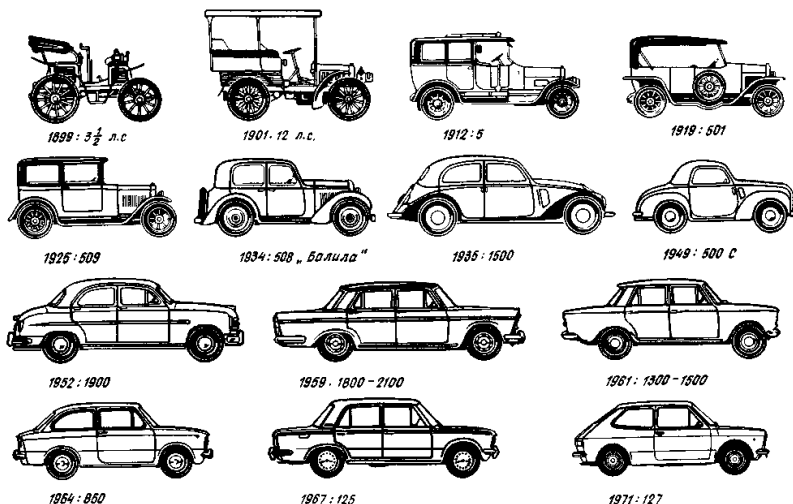


Рис. 8.12. Стадии развития автомобиля «Фиат».

Видно, что формы его менялись в направлении большего удобства и обтекаемости. В то же время развитие дизельного двигателя (рис.8.13) характеризуется в большей степени уменьшением размеров, массы и цены.

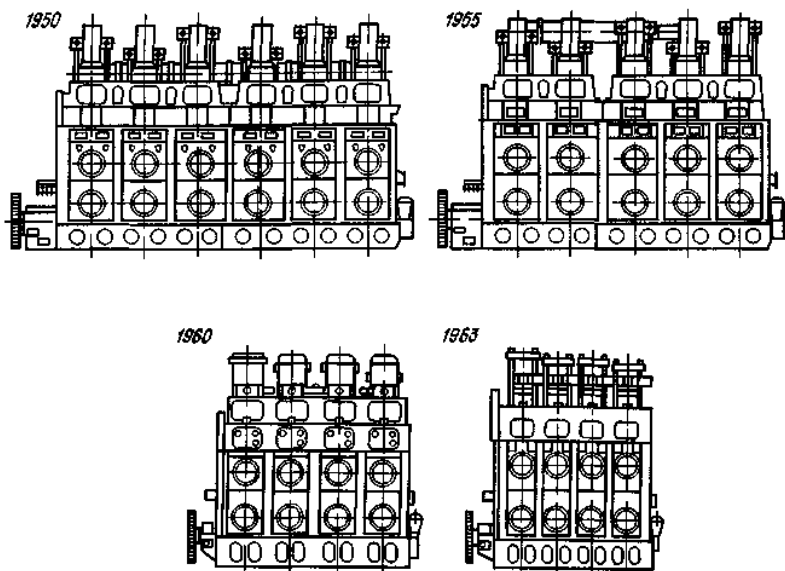


Рис.8.13. Стадии развития дизельных двигателей (У. Доксфорд).

Год выпуска	1950	1955	1960	1963
Мощность, л с	6800	7000	6640	7200
Длина, фт	57,5	52	33,5	28
Масса, т	404	395	260	240
Цена, фунт/л с	43,0	32,2	22,9	18,7

Хотя автомобиль остался автомобилем и дизель — дизелем, вместе с тем в связи с повышением технического уровня и ростом потребностей изменился и ряд их свойств. Изменились не только форма и эстетические свойства автомобиля, что легко видеть на рисунке, но также скорость, мощность, надежность, масса и др. Все это позволяет ввести понятие «технический уровень» технических систем для того или иного узкого интервала времени. **Под техническим уровнем понимается совокупность технических свойств — параметров, определяющих совокупную ценность изделия.**

Рассмотрим подробнее только один параметр — рабочее давление в технологическом и энергетическом оборудовании. На рис. 8.14 показано изменение рабочего давления в производственных и экспериментальных установках такого рода за 150 лет.

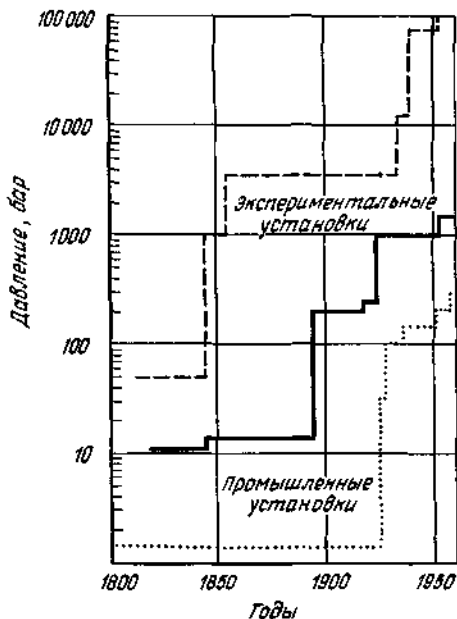


Рис.8.14. Изменение уровней рабочего давления в экспериментальных и производственных установках в процессе технического развития.

Для достижения повышенного рабочего давления в этих установках использовались различные принципы и средства, которые объединяются только результирующей абстрактной функцией — силой давления. Аналогичным образом по скорости движения можно объединять средства транспорта (рис.8.15).

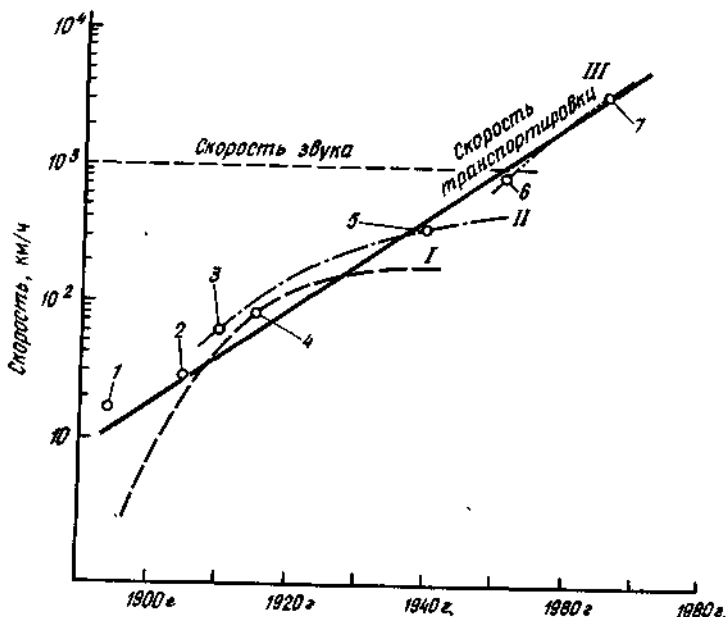


Рис. 8.15. Повышение скоростей транспортировки в процессе технического развития.

I — автомобили, II — винтовые самолеты, III — реактивные самолеты
 1 — пароходы, 2 — первые аэропланы, 3 — «Скаут» фирмы «Бристоль», 4 — «Фиат 510», 5 — «Кондор», 6 — «Каравелла» (805 км/ч), 7 — «Конкорд» (2335 км/ч)

На этом рисунке наряду с повышением скорости транспортировки вообще показано также изменение скорости по времени для различных видов транспортных средств (автомобиль, винтовой самолет, реактивный самолет). Классификация различных транспортных средств приведена на рис. 7.1.

Каждое из семейств средств передвижения с определенными функциями (рис.8.15) имеет характерную кривую эволюции скорости, асимптотически приближающуюся к определенному пределу. Этот предел обусловлен либо законом природы, например для транспортных средств с воздушным винтом, либо ограниченной применимостью средств, например для водного транспорта. Особым случаем являются ограничения, налагаемые окружающей средой на техническую систему, например дорога или планировка улиц. В этом случае эволюционная кривая неасимптотически приближается к границе, обусловленной влиянием окружающей среды. Часто ряд систем одного

технического уровня называют поколением. Так, говорят, например, о поколениях ЭВМ.

Из диаграммы эволюции рабочего давления (рис.8.14) видно, что в определенный момент времени для систем различного назначения характерны разные предельные величины одного и того же рабочего параметра. На рис. 8.16 показано семейство эволюционных кривых, отражающих, кроме того, различный технический уровень в разных областях от исследований до производства.

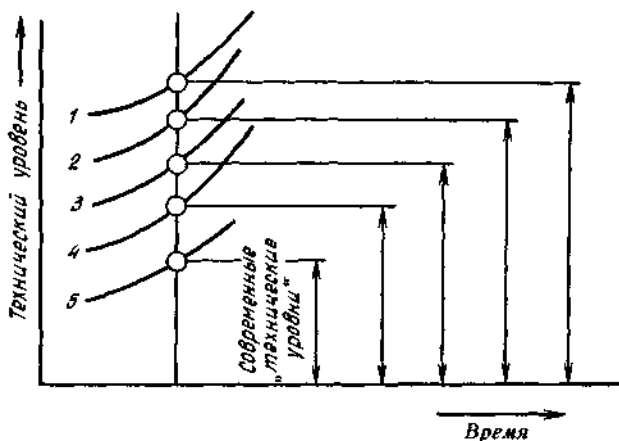


Рис. 8.16 Эволюционные кривые для некоторых областей деятельности и соответствующие «технические уровни».

1 — изобретения и открытия, 2 — исследования, 3 — разработки, 4 — производство, 5 — эксплуатация

Эволюционная кривая для технической системы может быть заимствована из биологии (рис.8.17), поскольку в этом отношении ее можно уподобить живому организму.

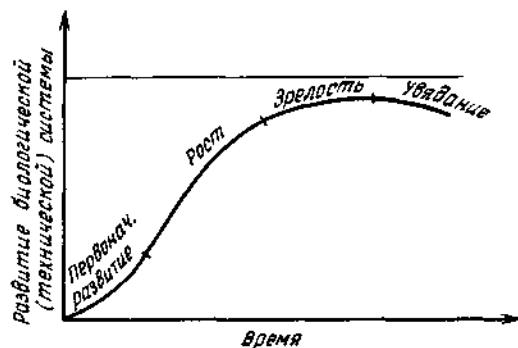


Рис.8.17. Кривая развития.

Эволюция спроса на техническую систему

Важной величиной, которая также сильно изменяется со временем, является эволюция спроса на техническую систему. Для простоты анализа будем полагать, что производство технической системы всегда соответствует спросу на нее. Действительное увеличение и уменьшение спроса с течением времени происходит примерно так, как показано на рис. 8.18, а.

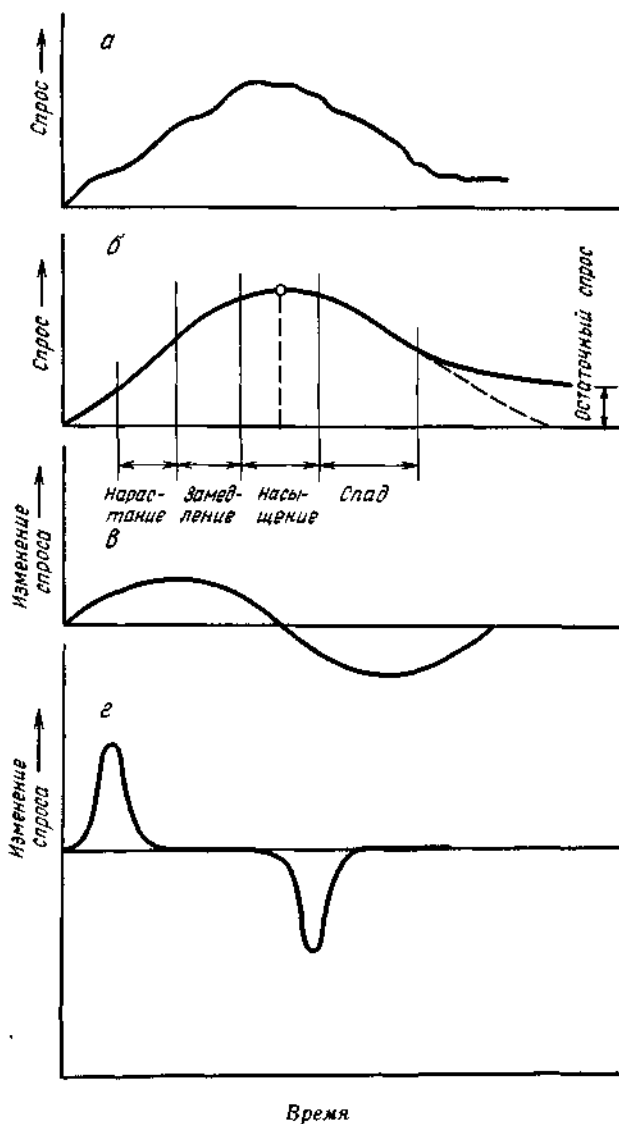


Рис.8.18. Кривые спроса.

a — реальная кривая спроса, *б* — идеальная кривая спроса; *в*—кривая производной от спроса; *г*— кривая производной с всплеском спроса

Эту кривую можно заменить упрощенной более плавной кривой (рис. 8.18,б). Спрос на техническую систему, даже после разработки новых более эффективных систем, не всегда падает до нуля, о чем свидетельствует ниспадающий участок кривых на рис. 8.18, а и б. Старые машины будут использоваться и в дальнейшем, хотя и в меньших масштабах. Так, например, специализированные сельскохозяйственные машины не могут полностью вытеснить трактор и комбайн. Кривую спроса на техническую систему можно представить также в виде первой производной предыдущей кривой. В результате получается кривая типа синусоиды (рис. 8.18, в). Естественно, что реальная кривая (рис. 8.18, г) не будет такой гладкой вследствие различных внешних воздействий, например политических, социальных или экономических изменений. На рис. 8.19 в качестве примера представлено фактическое изменение спроса на магнитофоны трех типов, выпущенных друг за другом.

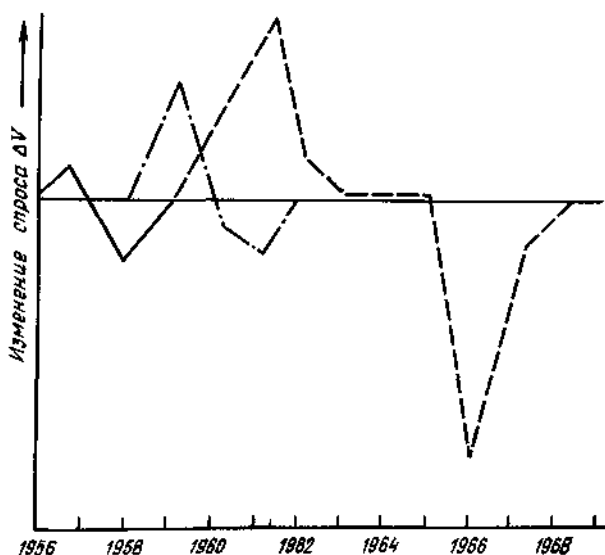


Рис. 8.19. Пример кривых спроса для трех типов выпускающихся последовательно магнитофонов
 — MGK 10 — — «Сонет I», — — — «Сонет II»

Если исследовать кривые спроса на различные технические системы, то можно установить, что длина волны этих кривых с течением времени укорачивается. Таким образом, технический прогресс

приводит к тому, что длительность использования изделий укорачивается. С другой стороны, при ускорении выпуска новых изделий амплитуда спроса на них увеличивается. Отсюда следует, что нужно всемерно ускорять разработку новых систем. Так, кстати, и было всегда. Это можно проиллюстрировать следующим примером. Длительности периода создания (от замысла до готового изделия) для ряда систем изменялись следующим образом: фотоаппарат — 112 лет (1727—1839); радио — 35 лет (1867—1902); радар — 15 лет (1925—1940); телевизор — 12 лет (1922—1934); атомная бомба — 6 лет (1939—1945); транзистор — 5 лет (1948—1953). Такое снижение длительности периода создания изделий, обусловленное техническим прогрессом, можно предвидеть и использовать в прогнозах. Другой важный признак кривой изменения спроса, а именно локализация изменения спроса на все более узком промежутке времени (рис.8.18, г), связан с динамизмом современной сферы потребления.

Факторы эволюционного процесса

Как известно, при создании технической системы используется, наряду с новыми методами изготовления, новым сырьем и новыми техническими решениями, накопленный в прошлом опыт. Прилагаются усилия к улучшению свойств существующих изделий и создаются новые изделия, которые должны удовлетворять возрастающим потребностям людей. С развитием цивилизации эти потребности изменяются. Если раньше людей интересовали в основном лишь проблемы поддержания жизни, то сегодня их потребности часто намного превышают жизненно необходимые. Однако изменяются и условия функционирования технических систем. С развитием науки и техники появляются знания, которые позволяют применить новые материалы, технологическое оборудование и технические решения. Эти технические возможности используются полностью лишь при благоприятной экономической ситуации. Кроме того, необходима мотивация для побуждения общества к решению той или иной технической проблемы. Эта мотивация может вытекать либо из безусловных потребностей общества, либо из сферы экономики (например, стремление к увеличению дохода), либо из развития науки (например, стремление к достижению практического эффекта от теории). При исследовании этих взаимосвязей не следует забывать, что общество, наука, техника и экономика развиваются взаимозависимо. Комплекс этих отношений в схематической форме представлен на рис.8.20.

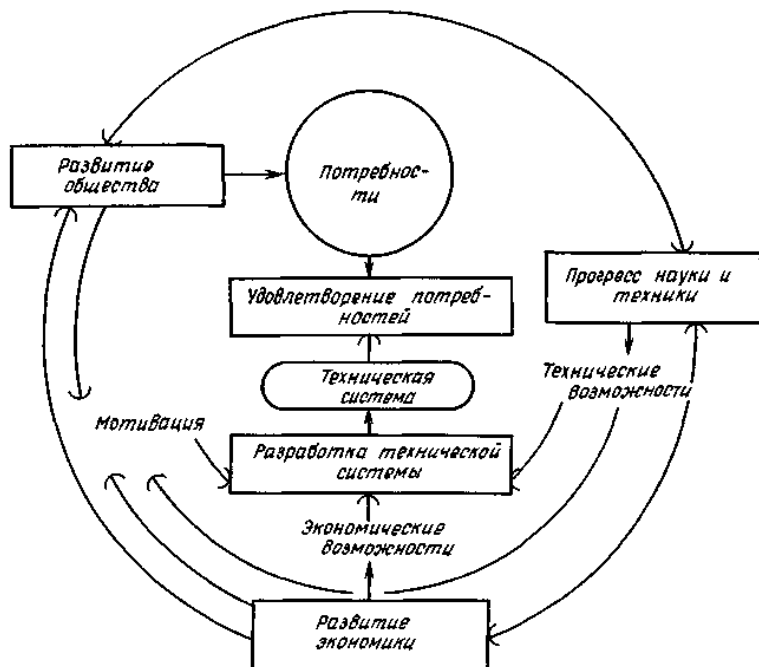


Рис. 8.20. Отношения в макросистеме общество — экономика — наука и техника

Кроме внешних отношений следует учитывать внутренние связи процесса конструирования, существующие между конструктором, специальной информацией, техническими средствами, руководством и условиями процесса конструирования.

На эволюцию технических систем оказывают влияние и такие факторы, как формирование потребностей и наличие сырья. Есть ли в стране собственные источники нефти, угля, железа, никеля, титана, урана? В условиях экономического сотрудничества различных стран эти вопросы отступают на второй план. Важную роль играет также приемлемая степень риска. Статистические исследования показывают, что, например, в США из 100 предпринятых разработок реализуется только 10, из которых лишь в трех случаях производится такое изделие, которое оказывается на уровне современных требований и находит широкое применение.

Организация и объем исследовательских и конструкторских работ

Одновременно с возрастающими требованиями к техническим системам, ростом их числа и увеличением сложности должен соответственно возрастать объем исследовательских и конструкторских работ. Косвенно он характеризуется количеством специалистов, занятых исследованиями и разработками. Можно считать, что в настоящее время во всем мире в этой области работают примерно 4 млн. человек, что составляет около 0,1% всего населения. Однако в промышленных странах этот процент выше; согласно статистическим данным, в 1966 г. в США он составлял 6,2% всего населения. Другим показателем объема исследовательских и конструкторских работ являются расходы на них. Промышленные страны тратят на эти работы все больше и больше средств. В настоящее время в высокоразвитых странах расходы на научно-исследовательские и опытно-конструкторские работы составляют 2—3% общей суммы национального дохода. Интересно также соотношение между отдельными видами расходов. Так, например, доля расходов на фундаментальные исследования составляет около 10% общих расходов на НИОКР. Возникает вопрос, каким образом покрывать эти расходы. Растущая стоимость разработок и повышающийся риск неудачи ограничивают возможность выполнения исследовательских и конструкторских работ без поддержки государства. Высшая школа, например в ФРГ, финансируется как государством, так и по прямым заказам промышленности. В связи с этим все больше высших учебных заведений занимается проведением научно-исследовательских и опытно-конструкторских работ.

Поддержка государства проявляется и в финансировании информационных центров и библиотек, которые предоставляют свою информацию заинтересованным лицам и организациям. Большие расходы и недостаток высококвалифицированного персонала побуждают повышать эффективность исследовательских и конструкторских работ. Для этого следует стремиться к сотрудничеству на возможно более широкой основе, в том числе на международном уровне, так как ни одно государство в одиночку не в состоянии проводить обширные исследования и разработки во всех областях. Сегодня решение исследовательских и конструкторских задач требует высокой степени специализации. Важно не только вести собственные разработки, но и следить за разработками в других странах, используя зарубежные достижения путем приобретения лицензий и уникального оборудования.

2. Тенденции технического развития

Для определения тенденций развития науки и техники следует исходить из целей общества.

Философия и социология, которые занимаются выяснением целей общества, обычно далеки и абстрактны для людей, занимающихся техникой. Однако в настоящее время ответственность инженеров за развитие общества подобна ответственности врачей за здоровье людей, так что решение этой проблемы нельзя больше оставлять политикам, философам и социологам. Ученые и инженеры, руководствуясь социально-нравственными нормами, должны препятствовать использованию техники в эгоистичных целях отдельных социальных групп.

Существует ряд неоспоримых целей общества, очевидных большинству людей. Не рассматривая вопрос о приоритетности этих целей, приведем наиболее значительные из них: а) исключение голода; б) исключение войн; в) избавление от болезней и увеличение продолжительности активной жизни людей; г) борьба с преступностью; д) повышение благосостояния; е) повышение уровня образования; ж) сокращение доли физического труда; з) сокращение рабочего времени.

Другой вопрос состоит в том, насколько эти целевые установки могут содействовать счастью и удовлетворенности людей. Мнения о том, какую роль в этом должна играть техника, разделились. Некоторые предлагают полностью отказаться от техники, другие выступают за ее бесконтрольное развитие. Независимо от высказываемых крайних мнений наука и техника будут развиваться. Важно, чтобы они развивались только на благо людей. Для того чтобы люди могли правильно управлять развитием техники, необходимо знать, в каком направлении оно должно происходить и каким образом оно должно регулироваться.

По-видимому, рассуждения на эту тему могут завести слишком далеко от темы книги. И все-таки каждый инженер должен понимать значение своей деятельности и всегда сопоставлять свою задачу с интересами общества.

Тенденции развития техники часто формулируются на уровне определенных свойств технических систем. Признанными тенденциями развития являются механизация, автоматизация, электрификация, компьютеризация, использование новых принципов работы и прогрессивных способов производства.

Механизацию можно определить как передачу функций действия от человека машине. Аналогично этому автоматизация — это передача функций управления и контроля от человека машине. Электрификация,

химизация и др. означают расширение использования электрических, химических и других явлений для решения технических задач.

Анализ этих тенденций показывает, что они непосредственно влияют только на снижение доли физического труда; связь с другими целями общества проявляется косвенно. Связь этих тенденций с целями общества станет яснее, если мы рассмотрим развитие важнейших отраслей техники. Технический уровень определяют:

а) информатика, электроника и вычислительная техника; б) ракетно-космическая техника; в) атомная энергетика; г) транспорт и связь; д) охрана окружающей среды; е) медицинская техника; ж) исследования морей и океанов.

Достижения в этих областях должны повышать благосостояние и охранять здоровье людей, удовлетворять их потребности и делать жизнь более полной.

Прогресс техники осуществляется как развитие свойств технических систем. Рассмотрим теперь тенденции развития на примере некоторых свойств систем.

Функции. *Рабочая функция* переживает бурное развитие, и в этом в первую очередь проявляется технический прогресс.

Функции подготовки и обслуживания, по-видимому, не должны коснуться какие-либо революционные изменения. Электрические, пневматические и гидравлические приводы как технически, так и экономически улучшаются постепенно.

Функции управления, регулирования и согласования интенсивно развиваются как одно из главных приложений автоматизации, кибернетики и электроники. Эти функции уже достигли высокого уровня и продолжают совершенствоваться.

Функционально обусловленные свойства. Определяемые количественно функционально обусловленные свойства характеризуют техническую ценность машины. Давление, мощность, температура, скорость и другие технические параметры машин возрастают. С целью достижения максимальной экономической эффективности создаются все более крупные функциональные единицы (турбины, корабли, самолеты).

Производственные свойства. Эти свойства также существенно изменяются под влиянием технического прогресса, но не всегда в направлении увеличения. Когда, например, повышаются требования к надежности, приходится мириться с более коротким периодом использования технической системы. Отчетливо проявляется тенденция к уменьшению размеров (миниатюризация) и массы изделий. Все более широкое применение находят блочный принцип и

кассетирование, облегчающие эксплуатацию и техническое обслуживание.

Эстетические свойства. Важность этой категории свойств повышается, и существует лишь относительно небольшая группа технических систем, для которых эстетические свойства не имеют значения. Привлекательный внешний вид изделия, как отмечалось выше, не только способствует лучшему восприятию, но и повышает производительность труда.

Эргономические свойства. Потребность в свойствах этой категории возникла недавно. Здесь необходимо отличать требования, вытекающие из физиологии и психологии человека, от преувеличенных претензий на оригинальность. Поэтому считается, что эргономика должна выполнять требования, учитывающие возможности человека и охраняющие его здоровье.

Манипуляционные свойства. Потребности в транспортных средствах для перемещения технических систем все больше растут, как и затраты на хранение, транспортировку и монтаж изделий. Повышаются также и требования к упаковке, и притом не только в отношении транспортировки, но и с точки зрения рекламы: упаковка должна производить благоприятное эстетическое впечатление.

Характеристики поставок, планирования и учета. Эти характеристики, обеспечивающие планирование и учет продукции, как и доставку товара потребителю, очень важны в условиях конкуренции и повышенных требований потребителя к обслуживанию.

Характеристики соответствия правовым нормам. Расширяющиеся международные экономические связи усложняют задачу удовлетворения всем правовым нормам, в том числе международным, а также проверку на патентную чистоту. В связи с этим все более возрастает ответственность изготовителей за качество своих изделий.

Экономические свойства. Для обеспечения конкурентоспособности изготовитель должен поддерживать как можно более низкую себестоимость. Средствами достижения высокой экономической эффективности изделий являются рационализация работ, стандартизация деталей, типизация узлов, высокопроизводительное оборудование, квалифицированный персонал, стимулирование высокопроизводительного труда и т. п.

Качество изготовления. Так как технические системы непрерывно усложняются и совершенствуются, то к качеству изготовления предъявляются все более высокие требования.

Конструктивные свойства. Внутренние (конструктивные) свойства определяют рассмотренные выше внешние свойства. Поэтому необходимо уделять большое внимание улучшению основных

конструктивных свойств изделия — структуре, форме, размерам (массе) и материалам.

Структура — это упорядоченное множество элементов и их отношений. Несмотря на нормализацию и стандартизацию, ассортимент конструктивных элементов возрастает, а отношения между ними во все более совершенных технических системах усложняются. Основными тенденциями в конструировании являются специализация и агрегирование функций конструктивных элементов. Аналогично и в пространственных отношениях элементов структуры существует несколько направлений развития; прежде всего это тенденция к созданию блочных систем, когда из подготовленных групп (блоков) можно сформировать различные варианты технической системы. Другим направлением является кассетирование, упрощающее манипуляции со сменяемыми элементами.

Основной тенденцией в отношении *формы* является упрощение или унификация, так как форма оказывает значительное влияние на стоимость. Однако это часто противоречит эстетическим требованиям, в связи с чем приходится искать компромиссное решение.

Компактность и минимальная масса обычно являются желательными конструктивными свойствами, если это не противоречит другим требованиям. Улучшение манипуляционных свойств, экономия материалов и производственных площадей — наиболее очевидные следствия компактности и минимальной массы изделий.

В связи с разработкой разнообразных новых материалов возможности конструкторов в отношении выбора материалов непрерывно возрастают. При этом следует отметить: а) расширение количества применяемых металлических материалов (в настоящее время — более 50); б) применение искусственных материалов (сплавов, пластмасс и синтетических материалов); в) повышение качества материалов (так, прочность стали повышена с 30 кГ/мм² до 70—80 кГ/мм²); г) использование композиционных материалов с различными свойствами.

Качество обработки поверхности улучшается с появлением и развитием новых технологических методов.

Применение высокопроизводительных станков повышает возможность достижения малых допусков, но, с другой стороны, конструкция должна быть такой, чтобы нормальное функционирование обеспечивалось при возможно больших допусках.

3. Управление процессом технического развития

Изложенное выше позволяет заключить, что развитие технических систем представляет собой длительный и дорогостоящий процесс. Взаимосвязи в этом процессе в упрощенной форме представлены на рис. 8.20. Для сложных технических систем характерна зависимость их развития от развития отдельных ее частей. Кроме того, результат развития зависит от технико-экономического уровня страны и предприятия, как и экономической ситуации в целом. Поэтому такой процесс не может протекать стихийно, тем более, что общество заинтересовано вести его быстро и эффективно.

Под управлением процессом развития технических систем мы понимаем организацию работ в научно-исследовательских институтах и опытно-конструкторских бюро с использованием непосредственных и косвенных (экономических и административных) механизмов для достижения наилучших результатов. В принципе, следует различать две стороны проблемы, а именно, а) постановку задач исследований и разработок и б) контроль за их решением.

Характер управления исследованиями и разработками зависит от уровня, на котором оно осуществляется. Рассмотрим сначала способы управления, применяемые на правительственном уровне. В большинстве стран подход к управлению исследованиями и разработками одинаков; различия имеются лишь в деталях.

Выявляются потребности и возможности развития важнейших областей экономики, таких, как промышленность, сельское хозяйство, здравоохранение, оборона, строительство, транспорт и др., и затем устанавливается общее направление политики развития и конкретизируются задачи. Управление осуществляется государственными (законодательными), исполнительными и совещательными правительственными органами. Часто наряду с этими органами существуют самостоятельные министерства и ведомства по науке и технике.

Техническая политика, состоящая в постановке задач исследований и разработок, определении их приоритетности и выборе наиболее эффективных средств решения, является важнейшим фактором технического развития.

Как мы уже упоминали, речь идет о двух аспектах управления:

постановке задач и контроле за их решением. Постановка задачи должна основываться на долгосрочных прогнозах развития, т. е. ко времени постановки задачи необходимо иметь возможно более точные представления о том, что будет желательно сделать через 5, 10 и более лет. Проблематику управления трудно изложить кратко, поэтому мы

ограничимся несколькими замечаниями, относящимися к методологии управления.

Управленческое решение должно опираться на объективные факты и закономерности, установленные достоверно. Ни один прогноз не может основываться только на интуиции. Добавим, что сбор и обработка необходимой для принятия решения информации — сложная проблема, особенно в связи с задачами перспективного планирования. При этом необходимо широко применять математические методы и вычислительную технику. Большие объемы памяти запоминающих устройств и высокое быстродействие современных вычислительных машин позволяют обрабатывать огромные массивы информации и вследствие этого подготавливать различные варианты решения, что делает принятие решения более обоснованным и объективным.

При решении проектно-конструкторских задач также необходимо преследовать цель достижения максимума результатов при минимальных затратах. Не все нужно разрабатывать самостоятельно. Нередко задача может быть решена лучше и качественнее путем передачи, например, процесса конструирования специальной конструкторской организации или путем приобретения лицензий.

Таким образом, управление техническим развитием — это, с одной стороны, планирование, при котором определяются тенденции развития, намечаются цели и формулируются задачи; с другой стороны, — это координация и контроль исследований и разработок с целью повышения их эффективности.

4. Мотивация исследований и разработок

Для эффективного технического развития недостаточно технических и экономических возможностей, необходимо также наличие стимулов, действующих как движущая сила развития.

Без заинтересованности в решении не может быть решена никакая задача. Можно даже сказать, что **только тогда, когда комбинация этих трех факторов — технического уровня, экономических возможностей и заинтересованности — приобретает нужную силу, приводится в движение механизм исследований и разработок.**

Когда имеется убедительная мотивация, она оказывает влияние на техническую и экономическую ситуации; с другой стороны, технический и экономический потенциалы облегчают мотивацию технического развития.

Убедительную мотивацию создает экономическое и техническое соревнования. Так были изобретены паровая машина, двигатель

внутреннего сгорания, радио, электрическая лампа, реактивный двигатель, вычислительная машина, атомный реактор. Важно лишь, чтобы такое соревнование имело целью благо и не переходило в соперничество государств.

Выводы

1. Развитие техники должно поддерживать и гарантировать гуманные цели общества. Каждый технический администратор, инженер и научный работник должен осознавать лежащую на нем обязанность и не допускать злоупотребления достижениями техники в частных интересах.
2. Тенденции развития технической системы можно выявлять путем аппроксимации изменений ее свойств на определенных (например, годовых) интервалах времени.
3. Темпы технического развития постоянно ускоряются благодаря а) сокращению времени разработки; б) сокращению эксплуатационного периода технических систем; в) повышению требований к свойствам систем.
4. Развитие технических систем находится под влиянием трех главных факторов: технического уровня, экономического потенциала и заинтересованности.

8.5. Специальные теории технических систем и требования к ним

При рассмотрении видов теории с точки зрения их применения указаны два вида теории: общая и специальная. До сих пор рассматривалась общая теория.

Специальные теории технических систем имеют дело с отдельными категориями систем. Возможности построения специальных теорий обсуждались в гл. 6. Классификация технических систем может быть проведена в соответствии с рассмотренными там аспектами; классы и даже некоторые виды технических систем могут быть объектами специальных теорий.

Обычно инженеры и научные работники специализируются на определенных видах, типах и семействах технических систем, таких, как гидравлические и поршневые машины, или паровые и газовые турбины, или насосы, компрессоры и т.п. (рис. 8.21).

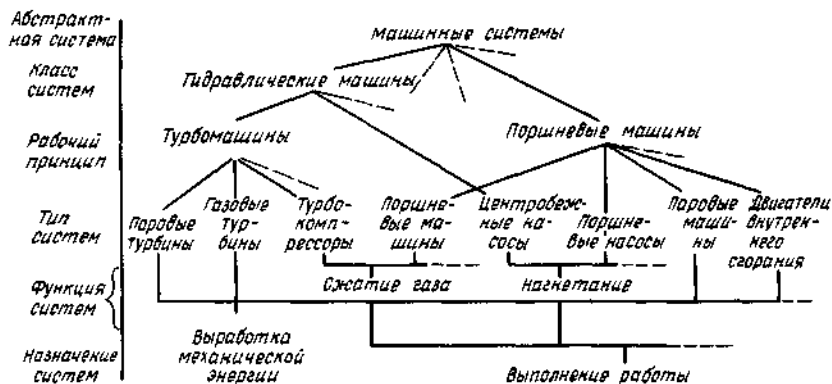


Рис. 8.21. Разделение технических систем в соответствии со специальными теориями систем (на примере из области машиностроения).

Аналогично данные, приведенные в табл. 6.5, могут служить исходным пунктом для классификации специальных теорий по иерархическому признаку, например, теории станков, металлообрабатывающих станков, токарных станков и т. д. С другой стороны, учение о механизмах и учение о машинных элементах (деталях машин) известны как специальные теории составных частей технических систем. Подобных примеров можно привести много.

Эти рассуждения и примеры позволяют определить следующие важные аспекты для построения специальных теорий: а) функция (насосы, станки, подъемные устройства); б) принцип действия (поршневые машины, электродвигатели, реактивные двигатели); в) уровень абстрактности (станки, металлообрабатывающие станки, токарные станки); г) степень сложности (машинные системы, механизмы, машинные элементы).

В задачи специальной теории входят получение и систематизация знаний о соответствующем типе или виде технических систем и построение на этой основе методов их разработки.

Выявление взаимосвязей между общей и специальными теориями технических систем позволяет создать **логичную и стройную единую структуру научно-технических знаний**. Польза этого очевидна.

Специальные теории технических систем должны охватывать следующий комплекс вопросов:

- исследование системы преобразований; анализ элементов системы преобразований, таких, как виды входных воздействий, технология, операнды и т. д.;
- структуры технических систем определенных категорий, в том числе типичные функциональные структуры, их варианты и модификации;
- принципы действия, которые используются или могут найти применение;
- типичные носители функций (исполнительные органы) и их отношения;
- типичные свойства технических систем, конструктивные признаки и их реализация, способы обеспечения требуемых свойств, проблема качества, сравнительные оценки;
- изучение закономерностей создания и использования данной категории (вида) технических систем;
- обобщение полученного опыта и выявление тенденций развития.

Специальная теория, включающая рассмотрение всех этих вопросов, будет сильно отличаться от теорий, излагаемых в современных учебниках (хотя она и будет основываться в основном на той же информации).

Кроме того, при построении таких теорий и систематизации специальных знаний следует приспосабливать их к возможностям вычислительной техники, что обеспечит дальнейший прогресс в соответствующей области.

Уже в самом начале ставился вопрос о применении теории технических систем, а в дальнейшем была выявлена польза, которую может принести такая теория. Теория технических систем может служить в качестве: а) базы знаний для разработки новых технических систем; б) ориентира для учений о конструкциях; в) исходного пункта для специальных теорий технических систем; г) основы для применения ЭВМ (алгоритмы, системы банков данных и т.д.); д) руководства для системы обучения; е) связующего звена между специалистами разного профиля; ж) базы для систематических исследований в области истории техники.

Современная высшая школа практически не имеет таких учебных планов, которые в процессе обучения давали бы студентам полное представление об отрасли техники, в которой им предстоит работать, выявляя ее топологию и взаимосвязи с другими отраслями и окружающим миром в целом, что способствовало бы формированию гармонично развитого специалиста. Теория технических систем могла

бы играть в этом отношении важную роль, так как в ней с достаточно высокой степенью обобщения излагаются все основные аспекты системного подхода и дается инструмент для ориентации в любой специальной области техники. Она позволяет увязать между собой различные учебные курсы и объяснить цель их изучения. Такой подход будет способствовать лучшему пониманию связей между отдельными элементами системы обучения и содержания многих учебников. Теория технических систем делает понятной обучающемуся всю систему обучения и проясняет место отдельных дисциплин, например термодинамики как специальной теории процессов в тепловых машинах (специальном семействе технических систем) или теории сопротивления материалов как общего учения о прочности — свойстве технических систем. Теория технических систем дает системный обзор специальной проблематики, благодаря чему легче выявляются пробелы, которые необходимо ликвидировать при повторении учебного материала, и выдвигает взаимосвязи между специальными дисциплинами на передний план. Более того, включение этой теории в общий курс инженерного обучения позволяет объяснить и довести до сознания всех, в том числе и неспециалистов, сущность техники и технических систем и таким образом сделать более доступным и распространенным **понимание техники как одного из основных элементов ноосферы — элемента, создаваемого и используемого человеком**. Понимание истории техники также может стать более глубоким благодаря применению теории технических систем. Процесс развития технических систем становится более понятным, когда, например, уясняется зависимость изменения конструктивных признаков от факторов окружающей среды. В заключение можно сказать, что теория технических систем является отражением современного развития техники и представляет собой одно из проявлений понимания инженерами своей роли в современном обществе.

9. Облачные управляющие системы как объекты автоматизированного проектирования

Развитие научно-технического прогресса предусматривает внедрение автоматизированных систем в различные сферы хозяйственной деятельности, и в первую очередь в проектирование, управление оборудованием и технологическими процессами, а также создание комплексно-автоматизированных производств, которые можно быстро и экономично перестраивать. Системы проектирования и управления необходимо объединить в единый комплекс, который призван

увеличить производительность и улучшить качество как проектирования, так и управления. Современные автоматизированные производства насыщаются микропроцессорными устройствами, управляющими ЭВМ и системами. Но опыт показывает, что эффективность автоматизации зависит не только от насыщенности производства средствами управления и облачными управляющими вычислительными системами (ОУВС). Необходимо обеспечить органическое системное единство объекта управления и ОУВС и их гибкость. Следует учитывать, что нерациональное использование ресурсов ОУВС так же непозволительно, как недопустим простой производственных модулей. Задачи, рассмотренные в данном разделе, связаны с вопросами обеспечения гибкости и повышения функциональной эффективности ОУВС на примере гибких автоматизированных производственных систем (ГАПС). Предпосылкой для гибкости ОУВС является ее программируемость, а основой проектирования — системотехнический анализ и моделирование. Хотя ОУВС ориентирована на преобразование информации, а технологический процесс организует преобразование энергии и материи, системные модели того и другого элементов производства на уровне логико-время-количественных отношений предложено строить на основе единых принципов, единого математического обеспечения и графического языка. В этом случае обеспечивается эффективное отображение взаимодействующих информационно-алгоритмических и материальных процессов автоматизированного производства в целом, а также использование в САПР средств интерактивной машинной графики. Поскольку задачи проектирования алгоритмов управления и структур ОУВС нельзя решить с помощью математических моделей одного класса, они объединяются единой математической концепцией, обуславливающей использование сетевых моделей: сетей массового обслуживания, сетей Петри и их расширений. Это позволяет отображать и исследовать алгоритмы управления, структуры ОУВС, формализовать системотехнический анализ и синтез ОУВС и ГАПС в целом. Именно концепция сетевой модели позволяет объединить различные программные компоненты САПР единой базой данных и обеспечить единый подход в использовании различных типов аналитических, численных и имитационных моделей в проектировании и управлении. Предлагаемый подход не является единственным и не претендует на полную завершенность. Он имеет пути для развития и совершенствования. Поддержка идей данного направления определена их новизной и актуальностью, а также тем, что в настоящее время нет других разработок, которые бы с такой широтой охватывали

достижения современной математики и вычислительной техники в облачных технологиях. Предлагаемые основы являются реальной базой для объединения в единую технологическую цепь и интегрирования в единый комплекс средств автоматизированного проектирования и управления ГАПС, обеспечивая их достаточную инвариантность к типу производств, производимых продуктов и ОУВС.

9.1. Задачи и средства проектирования ОУВС ГАПС

9.1.1. Распределенные ОУВС и организация программного управления ГАПС

ГАПС— это управляемые с помощью ЭВМ системы, обеспечивающие быструю перестройку на изготовление новых изделий в условиях конкретных производственных ситуаций. Поскольку перестройка ГАПС — операция массовая, ее следует выполнять методами индустриальной технологии проектирования с применением САПР. Современный подход к созданию распределенных ОУВС ГАПС состоит в объединении в единую систему или сеть множества обрабатывающих средств (процессоров), устройств хранения информации, устройств управления, средств обмена и коммутации структуры. Названные четыре группы агрегатов являются базой для создания различных комплексов микро- процессорной техники , предназначенных для компоновки проблемно-ориентированных облачных управляющих систем. Микроминиатюризация способствует приближению средств обработки информации непосредственно к местам ее получения и переходу от централизованных к распределенным системам облачного управления. Большие интегральные схемы позволяют компоновать ОУВС с проблемной ориентацией на область различных промышленных применений. В управляющих системах автоматизированных производств еще не достаточна доля и сложность вычислительных задач, но многие из них предназначены для стабилизации и регулирования логико-время-количественных отношений в материальных и энергетических потоках производства. Поддержание таких отношений — наиболее сложная форма работы системы реального времени. При рассмотрении ОУВС как объектов системотехнического проектирования основное внимание будет акцентировано на общих чертах, характерных

для ОУВС различных гибких производств, а не на том, чем они отличаются.

Цель системотехнического проектирования ОУВС — выбор аппаратных средств, системных и прикладных программ, структур данных и способов взаимодействия этих компонентов, обеспечивающих заданный режим применения ОУВС. Сформулированная цель представляется многочисленными критериями, взаимная увязка которых очень сложна. Поэтому комплексное решение задачи системного проектирования заменяется обычно решением последовательности проектных задач, упорядоченных по важности и эффекту от их решения. В центре методологических вопросов создания систем автоматизированного проектирования ГАПС вообще и их облачных управляющих систем в частности находится задача реализуемости разработки в заданные сроки. Управляющие вычислительные системы и сети современных ГАП строят на основе разнообразных мини- и микро-ЭВМ, выпускаемых промышленностью. На рис. 1—4 приведены физические структуры реальных систем управления ГАП холодной штамповки, гальванопокрытий, производства печатных плат, токарном механической обработки и распределение функций между их элементами.

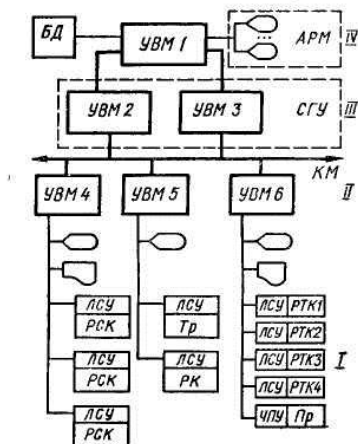


Рис. 1. Схема управления участком ГАПС холодной штамповки: БД — банк данных; АРМ — автоматизированные рабочие места технологов и оперативного персонала производства; СГУ — система группового управления участком с резервируемыми управляющими вычислительными машинами УВМ; РСК — роботизированный склад; Тр — транспорт; РК — роботокар; РТК — робототехнологический комплекс; Пр — пресс с числовым программным управлением ЧПУ; КМ — кольцевая магистраль

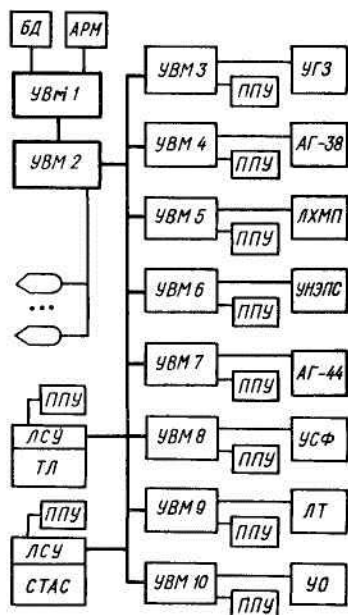


Рис. 2. Система управления участком ГАПС печатных плат: БД — банк данных; АРМ — автоматизированные рабочие места; УВМ1 — УВМЮ — управляющие вычислительные машины; ППУ — приемно-передающие устройства; 777 — транспортная линия; СТАС — стеллажный автоматизированный склад; УГЗ — устройство гидроабразивной зачистки; АГ-38 и АГ-44 — автоматизированные гальванические линии; ЛХМП — линия химико-механической подготовки; УНЭПС — установки нанесения фоторезиста, экспонирования, проявления и сушки; УСФ — установка снятия фоторезиста; ЛТ — линия травления; УО — установка оплавления

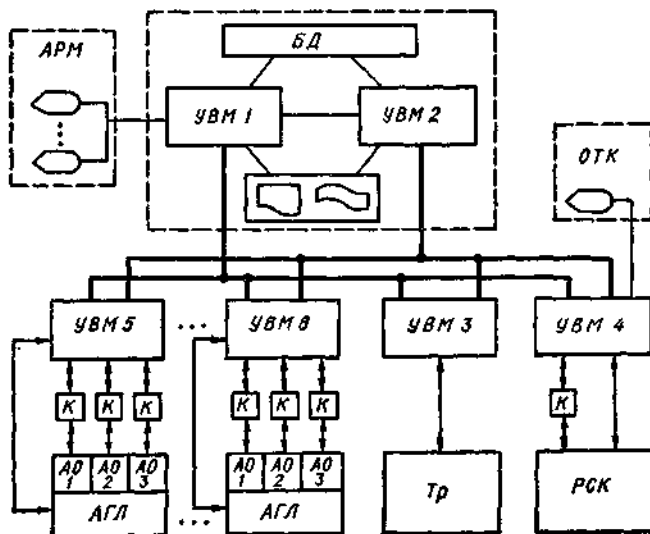


Рис 3. Система управления участком ГАПС гальванопокрытий: УВМ1 — автоматизированное проектирование и централизованное управление процессами гальванопокрытий, резервирование; УВМ2 — управление транспортно-складской системой, оперативно-календарное планирование и оперативное управление работой цеха, резервирование; УВМ3 — управление движением деталей в таре и подвесок на места монтажа-демонтажа, накоплением и передачей деталей на штангах; УВМ4 — управление движением штабелера, распределение загрузки, учет изделий; УВМ5 — УВМ8 — управление движением автооператоров АО автоматизированных гальванических пиний АГЛ и внутривенными процессами; К — микроконтроллеры; ОТК — отдел технического контроля; РСК — роботизированный склад.

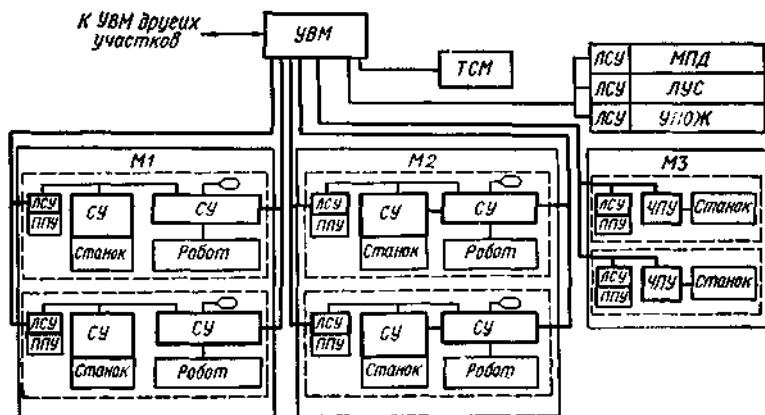


Рис. 4. Система управления участком токарной механической обработки цеха механической обработки ГАПС:
 ТСМ — транспортно-складской модуль, управляемый группой микро-ЭВМ; МПД — модуль подготовки деталей; ЛУС — линия удаления стружки; УПОЖ — устройства подачи охлаждающей жидкости; М1 — модуль из станков 16К20Т1; М2 — модуль из станков 16Б16Т1; М3 — модуль из станков МС 12-250; СУ — системы управления станками и роботами со встроенными микро-ЭВМ

Применение микропроцессоров, мини-и микро-ЭВМ привело к приближению этих средств к местам съема информации и выдачи управления. Агрегатно-модульный принцип компоновки ГАПС распространен и на облачную управляющую систему, когда управляющие вычислительные машины (УВМ) или процессоры закреплены за определенными технологическими агрегатами или даже встроены в них. Это привело к преобладанию распределенных микропроцессорных систем управления. Обмен в распределенных ОУВС обеспечивается по сетевой идеологии через кольцевую магистраль (рис. 1), на основе одно- или многосинных структур типа «общая шина» (рис. 2 и 3), радиальных структур связей (рис. 4) и т. д. Перестраиваемость автоматизированного производства предполагает реорганизацию алгоритмов управления операциями в ОУВС. Поэтому в рамках одной физической структуры необходима реализация различных функциональных (логических) структур управления. Гибкость функциональной структуры обеспечивается средствами настройки базовых операционных систем ОУВМ и специализированных операционных систем программного управления оборудованием. Программное управление логической структурной

управляющей вычислительной системы или сети требует унификации правил взаимодействия между ее элементами (протоколов).

Из анализа приведенных на рисунках структур видно, что наряду с распределенностью функций в ОУВС можно выделить иерархичность в их реализации. Так, в управляющем оборудовании (рис. 1) выделено четыре уровня управления: I — локальные системы управления (ЛСУ), встроенные в управляемое оборудование (сюда же относятся встроенные средства числового программного управления); II — управление технологическими процессами на однотипном оборудовании: роботизированными складами, транспортом, робототехнологическими комплексами (*УВМ 4 — УВМ 6*); III — координация управляющих процессов в пределах участка (*УВМ 2 и УВМ 3*); IV — обеспечение банка данных, САПР и оперативного персонала цеха (*УВМ 1*). Характерная особенность системы управления (рис. 1) состоит в использовании кольцевой магистрали, создающей условия для взаимодействия между любыми из УВМ, отнесенными к уровням II и III. Такая организации обменов свойственна локальным сетям, где эффективно решаются вопросы гибкого взаимодействия абонентов.

Распределение функций по уровням управления и УВМ. Можно видеть также в схемах систем на рис 2. Вопрос о выборе числа уровней управления, мощностей УВМ и микроконтроллеров, функциональных связей между ними решают на основе объема, сложности, связности и других характеристик задач управления.

ЭВМ верхнего уровня в приведенных структурах обеспечивают действия с банком данных управления и проектирования, подготовку и управление ходом производства. К ним подключают дисплейные рабочие места технологов и оперативного персонала ГАПС.

Видеотерминалы можно также подключать к УВМ отдельных агрегатов, создавая условия для автоматизированной отладки и обеспечивая автономное управление оборудованием. В системах управления (рис. 1 и 2) предусмотрены дисплеи диспетчера, складов, механика, энергетика, робототехнологических комплексов.

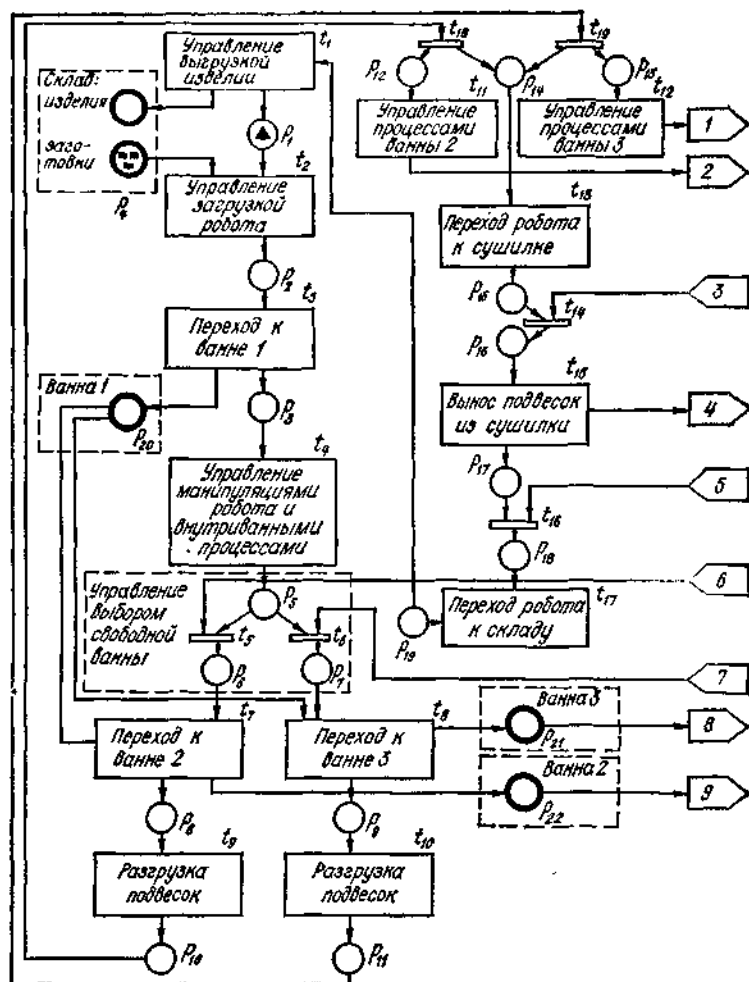
Печатающие устройства обслуживают складские операции, фиксируют аварийные ситуации.

Технологическая гибкость автоматизированных производств обеспечивается путем перестройки работы систем программного управления оборудованием. Перестройка технологии сопровождается изменениями в материальных, вычислительных и управляющих потоках ГАПС. Системный подход к организации программируемой технологии предполагает целостный охват всех элементов производства с их отношениями и связями, поскольку как бы хорошо

ни были спроектированы элементы сложной системы, ошибки в общей компоновке могут существенно сказаться на качестве и работоспособности системы.

Прежде чем сформулировать задачи системотехнического проектирования ОУВС, рассмотрим примеры модельного представления алгоритмов управления реальными ГАПС. Эти модели будут использоваться как для проектного анализа, так и в качестве программ управления оборудованием ГАПС. Подчеркнем, что управляющие модели будут объединять в единой форме действия ОУВС и объекта автоматизированного управления.

Например, для системы управления одного из участков гальванопокрытий (рис. 3) базовая часть управляющей модели процессов гальванических покрытий приведена на рис. 5.



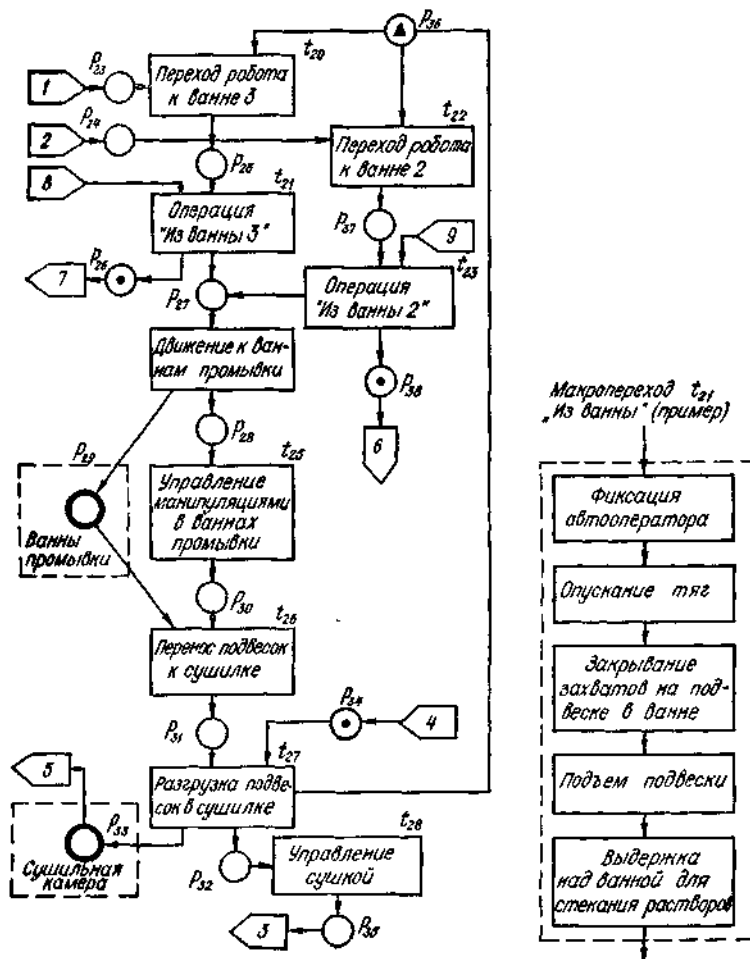


Рис. 5. Управляющая модель процессов гальванических покрытий

Модель представлена в виде структурной схемы, для изображения которой использован аппарат сетей Петри. Основными элементами сети Петри являются переходы t_j , обозначаемые на схеме прямоугольниками; позиции p_i , обозначаемые окружностями; стрелки; метки, изображаемые в позициях точками, треугольниками и т. п.

Графическая форма сети Петри подобна классическим структурным схемам последовательных алгоритмов, однако одна из ее

отличительных особенностей состоит в возможности отображения параллельных процессов в системах. Так же, как и в классических структурных схемах, в прямоугольных элементах сети (переходах t_j) описываются выполняемые операции. Например, на рис. 5 p_{20}, p_{21}, p_{22} и p_{29} — обозначают ванны анодной обработки, основного покрытия и промывки; p_1, p_2, p_3 и др. — фиксированные позиции роботов-автооператоров вдоль автоматической линии; p_4 — накопитель заготовок склада. Позициями на схеме интерпретируются также ячейки памяти для управляющих переменных (двоичных семафоров) программ управления операциями ОУВС. Так, если в позициях p_{26}, p_{38} записаны метки (интерпретируемые как семафорные единицы), то это свидетельствует о том, что ванны 3 и 2, соответственно, свободны. Наличие метки в p_{35} свидетельствует об окончании сушки. Распределение меток по позициям, называемое маркировкой сети, определяет состояние представляемой системы. Например, приведенное на рис. 5 маркирование указывает, что первый робот-автооператор находится в позиции p_1 (у склада), второй — в позиции p_{36} (у сушильной камеры); ванны и сушильная камера свободны, так как p_{26}, p_{34} и p_{36} помечены; заготовки находятся в складской позиции p_4 . Действия управляющих программ ОУВС сопоставлены переходам $t_2, t_4, t_9, t_{11}, t_{21}$ и пр. Функции этих программ указаны на рисунке в соответствующих прямоугольниках схемы. Перемещения роботов-автооператоров из одной позиции в другую обозначены переходами t_3, t_7, t_8 и т. д. Приведенная на рис. 5 схема управляющей модели является укрупненной. Поэтому многие из действий t_j представляют операции параллельных или последовательных управляющих программ. Например, действия t_{11} и t_{12} , реализующие управления процессами в ваннах 2 и 3, обеспечивают включение источников питания ванны, выставление опорного напряжения, измерение тока ванны, расчет общей площади погруженных изделий, расчет оптимальной плотности тока на изделиях и длительности электрохимического осаждения металла. В процессе осаждения производится периодический контроль параметров электролита и корректировка тока ванны. После окончания процессов покрытия выключаются источники тока. Завершение этих действий приводит к записи в семафорную позицию p_{23} или p_{24} единицы (метки).

Выполнение программ управления процессами в ваннах (действия $t_4, t_{11}, t_{12}, t_{25}$ и t_{28}) может производиться параллельно с помощью нескольких УВМ или же реализоваться в режиме мультипрограммирования на одной УВМ. Второй из приведенных способов управления допустим лишь в том случае, если отклонения в

периоде регулирования внутриванных процессов не ухудшают качество покрытий.

Динамическая интерпретация управляющей модели выполняется следующим образом. Действие t_j возбуждается в том случае, если во всех его входных позициях есть метки. Так, управление загрузкой выполняется, если робот находится в позиции p_1 и на складе p_4 есть заготовки. Выполнение совокупности действий t_2 приводит к изъятию меток из входных позиций и направлению метки по каждой из связей в выходные позиции. Завершение загрузки t_2 сопровождается появлением метки в p_2 , вызывающей действие t_3 , а отработка t_3 приводит к появлению метки в p_{20} (занятию ванны 1) и возбуждению операций t_4 . Переходами t_5 , t_6 , t_{14} , t_{18} описываются операции синхронизации процессов.

Динамическая интерпретация модели обеспечивает программное управление движением двух роботов (находящихся вначале в p_1 и p_{36}), а также вызов вычислительных и управляющих действий в соответствии с описанным алгоритмом. Для организации программного управления процессами ГАПС можно создать специализированную операционную систему, функция которой состоит в интерпретации управляющей модели и обеспечении взаимодействия технологического и управляющего оборудования автоматизированного производства. Переход к выпуску нового изделия при таком подходе к организации управления должен сопровождаться подготовкой соответствующей управляющей модели и стандартной сетевой структуры данных. Эти данные затем могут использоваться как алгоритм и интерпретироваться специальной операционной системой, организующей программное управление оборудованием и процессами автоматизированного производства.

Рассматриваемая модельная интерпретация позволяет управлять многопроцессными ГАПС, в которых объединена обработка нескольких типов изделий. Управление многопроцессными ГАПС становится аналогичным мультипрограммной обработке в вычислительных системах и может быть обеспечено соответствующими программными средствами.

Применение облачных сетевых управляющих моделей позволяет организовывать контроль, учет причин сбоев и аварийных ситуаций в ГАПС. Использование семафорных механизмов управления взаимодействием процессов как средств контрольной обратной связи позволяет устанавливать причины неисполнения шпускаемых действий t_j . Если в течение контрольного времени не происходит перевод управляющей модели в новое состояние, должна

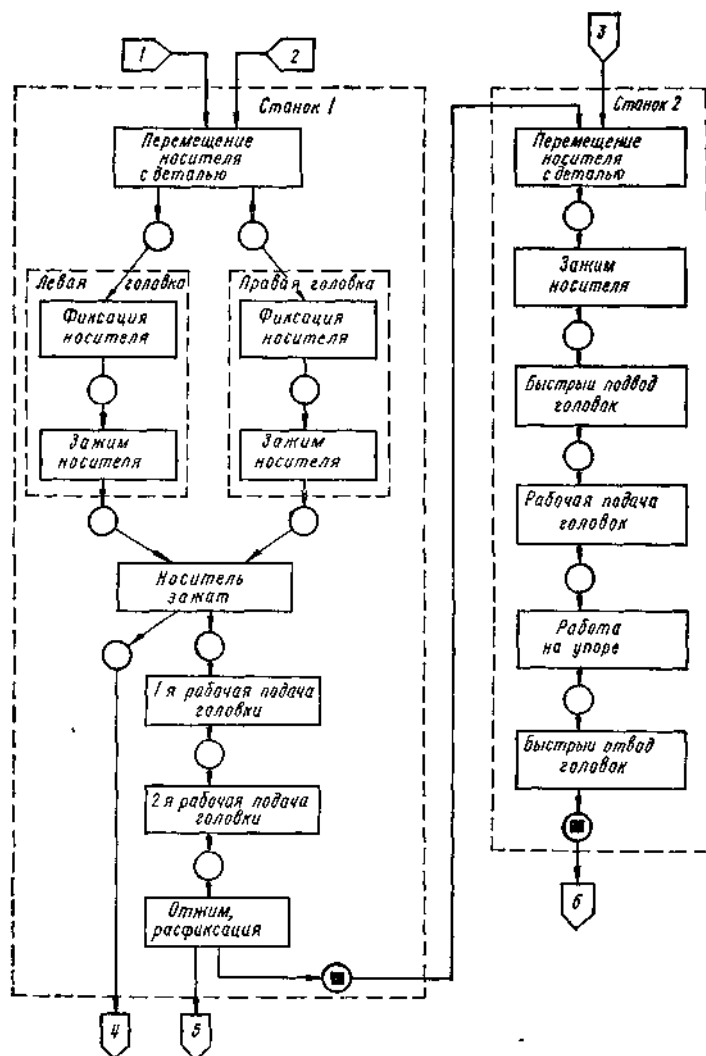
предусматриваться выдача сообщений операторам о сбойных ситуациях.

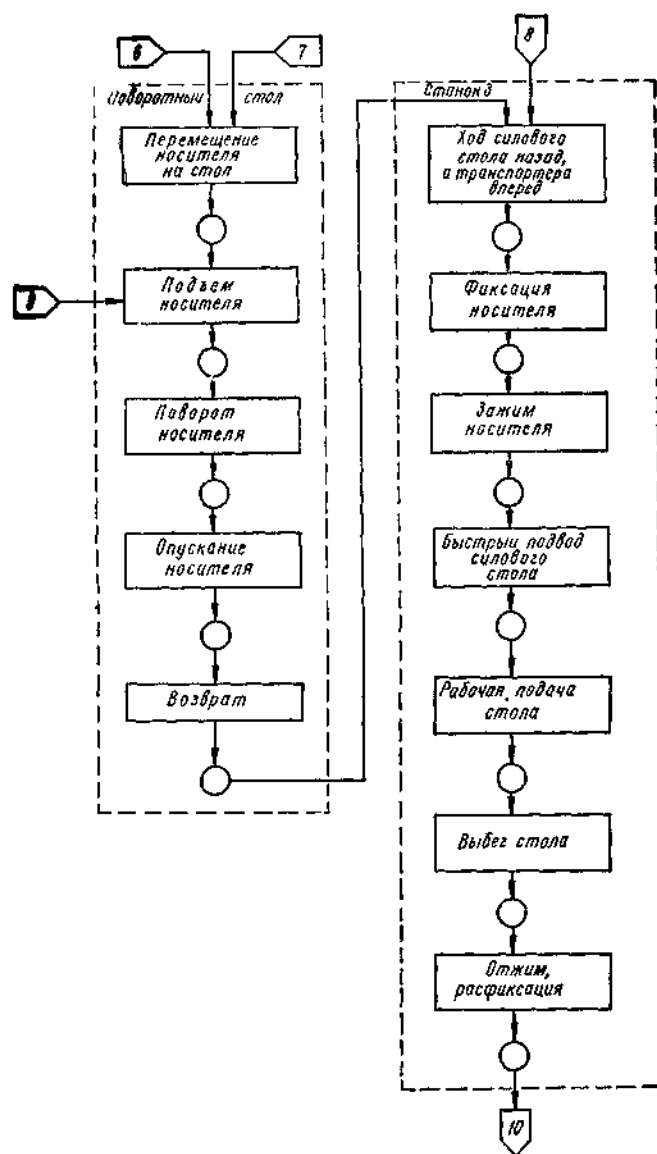
Например, «ток ванны не установлен», «подвеска не захвачена», «подвеска не отделяется от робота», «нет разгрузки у ванны 2», «робот не движется» и т. д.

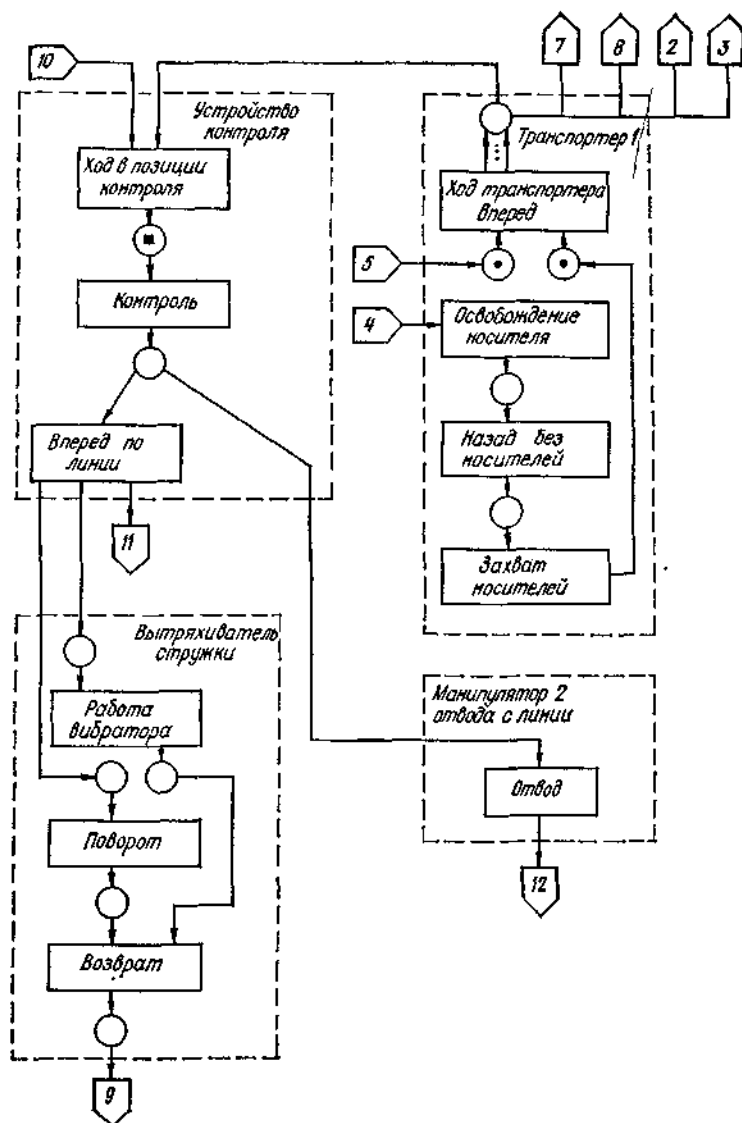
Реализация облачных управляющих моделей должна обеспечивать безопасность совместных действий роботов-автооператоров, исключаящих их столкновения на определенных позициях. Кроме того, необходимо обеспечить высокую производительность линии, которая зависит от временных диаграмм движения роботов-автооператоров (так называемых циклограмм). Решить эти вопросы можно на этапе проектирования облачных управляющих моделей путем их непосредственной интерпретации на специальных средствах имитационной отладки. Для имитационной отладки необходимо задать временные характеристики действий на переходах t_j .

Таким образом, облачная управляющая модель является результатом проектирования и способом организации облачных процессов управления, базой для поиска безопасных расписаний движения роботов, оптимизации технологии производства и технологии управления.

На рис. 6 изображена управляющая модель процессов на участке ГАПС обработки корпусных деталей.







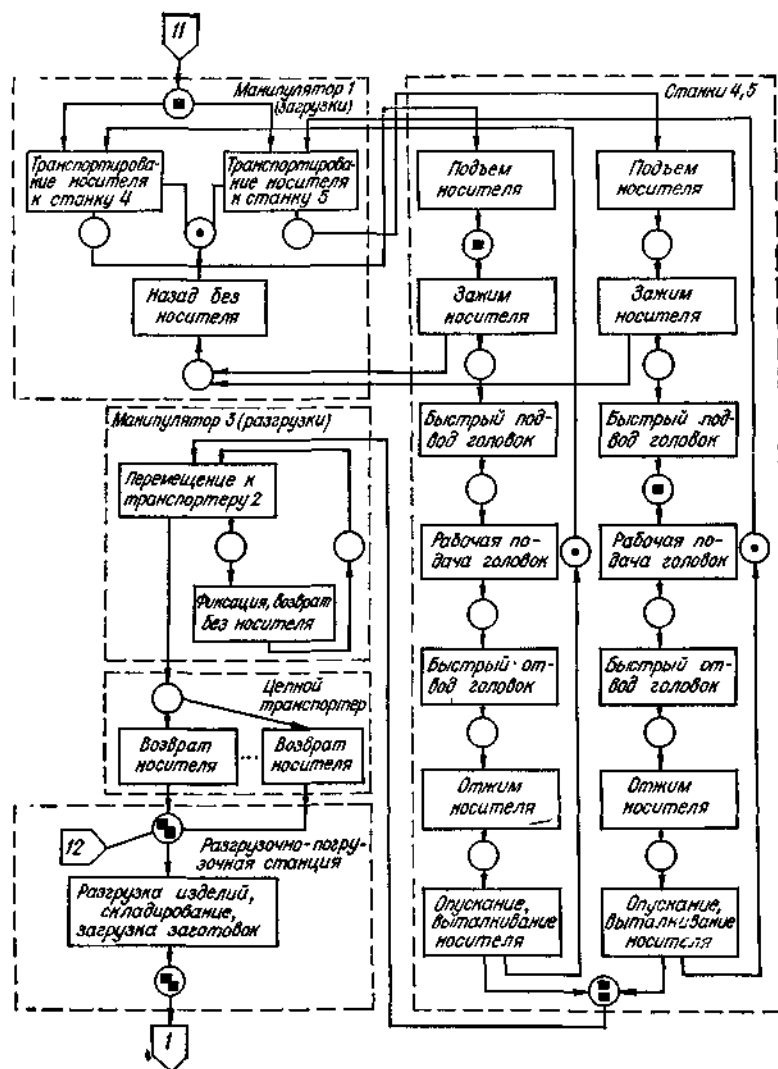


Рис. 6. Управляющая модель участка ГАПС обработки корпусных деталей

В отличие от приведенной на рис. 5 здесь дана структурированная модель, в которой выделены действия на станках 1- 5, поворотом стола, устройстве контроля, вытряхивателе стружки, манипуляторах 1—3, транспортерах 1-2 и на загрузочно-разгрузочной станции. Алгоритм управления представляет последовательность операций, определяющих маршрутную технологию или технологическую схему обработки детали, последовательность действий транспортно-накопительной системы, склада и т. д. Имитационная интерпретация управляющей модели позволяет строить и автоматически анализировать циклограммы, выявлять лимитирующие узлы (узкие места), оценивать производительность, решать задачи поиска эффективных расписаний. Если в составе ГАПС имеются станки с ЧПУ и средства подготовки к ним, то облачная управляющая модель становится программой управления второго уровня. Она определяет порядок взаимодействия подготовленных программ первого уровня, управляет распределением кадров программ и данных по станкам, а также динамикой изделий, материалов и инструментов. Для интерпретации облачных управляющих моделей, а также их имитационного анализа может создаваться единая, переносимая на различные УВМ операционная система (ОС) имитации и программного управления оборудованием. Организация управления ГАПС на основе использования эталонных облачных управляющих моделей показана на рис. 7.

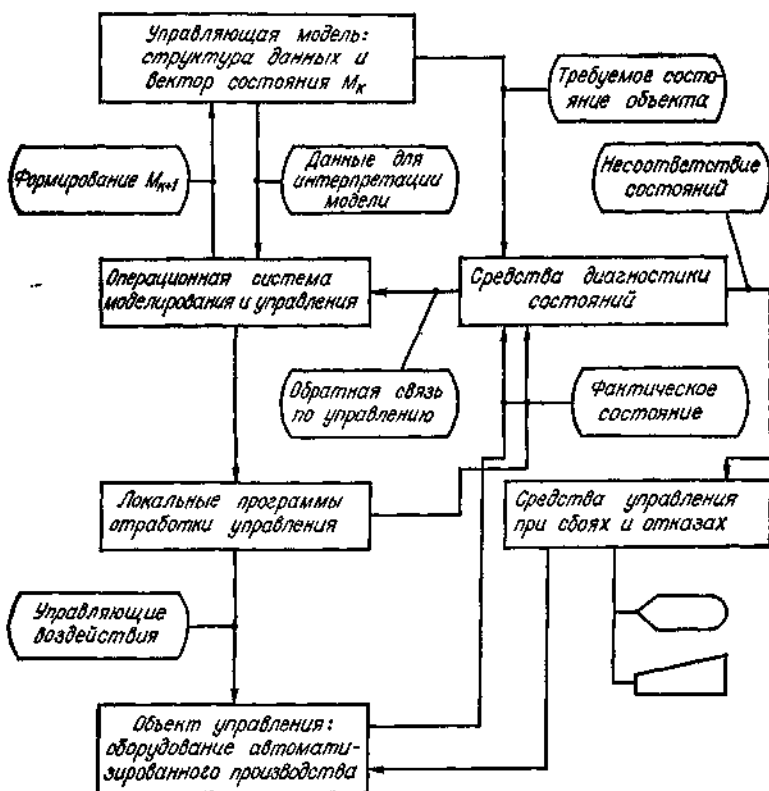


Рис. 7. Схема организации управления ГАПС на основе использования эталонных облачных управляющих моделей

Предложенный подход ориентирует на разработку средств облачного моделирования и управления, которые инвариантны к области применения гибкой автоматизации. При этом создаются условия для интегрирования базы данных и программного обеспечения проектирования и управления в единую систему, поддерживающую имитационную отладку, анализ и оптимизацию алгоритмов управления и выполняющую интерпретацию управляющей модели при программном управлении оборудованием автоматизированного производства и распределенной ОУВС.

9.1.2. Задачи системотехнического проектирования ОУВС ГАПС

Определим основные задачи, решение которых необходимо или целесообразно обеспечить при проектировании и перестройках ГАПС. **Гибкость облачной управляющей системы автоматизированного производства** обеспечивается разработкой облачных управляющих моделей представленных структурами данных, и их интерпретацией с помощью единой операционной системы моделирования и управления ГАПС. Математическое обеспечение программного управления строится путем реализации совокупности базовых облачных управляющих моделей, адаптируемых к конкретным облачным технологическим процессам и условиям производства на основе САПР. **Взаимодействие с объектом по жестким временным диаграммам и режим реального времени.** Управление автоматизированными производственными линиями, участками, цехами и другими подразделениями с помощью распределенной системы предполагает выполнение функций управления локальными производственными модулями, координацию их действий с действиями автоматической транспортной и складской систем, использование роботов. Взаимодействие с объектом обычно осуществляется по принципу прямого цифрового управления, реализуемого станками и иными средствами с числовым программным управлением. Запуск следующей операции на автоматическом оборудовании может выполняться по сигналам от службы времени облачного управляющего процессора в соответствии с последовательностью директивных сроков управления. В этом случае организуется действие на объект или взаимодействие с ним по жестким временным диаграммам. Строгое регламентирование диаграммы взаимодействия облачной управляющей системы и объекта возможно лишь при монопольной однопрограммной организации работы микропроцессора. Во взаимодействии по жесткой временной диаграмме режим реального времени присутствует в наиболее строгой своей форме. Более простая форма режима реального времени предполагает лимитирование времени ответа ОУВС на запрос от объекта. Ограничение на время реакции связывается в этом случае с выполнением периодических действий. В мультипрограммных системах реальный масштаб времени задается предельным временем пребывания заявок в системе, включающим время ожидания в очередях и обслуживания процессорами и другими устройствами. **Корректность системных действий и взаимодействий.** Переход от централизованных к распределенным системам породил высокий параллелизм и асинхронность информационно-алгоритмических процессов в ОУВС. Взаимодействующие процессы могут изменять

свое состояние под управлением средств операционной системы или при поступлении необходимых данных. Эффективность распределенной системы во многом зависит от способов синхронизации между процессорными элементами, работающими совместно над множеством задач. Асинхронный принцип управления не предписывает заранее, в какой последовательности должны активизироваться процессы. Последовательность определяется в ходе вычислений и формирования управлений. Это обеспечивает некоторую подстройку динамики облачной управляющей системы к реальному ходу облачных управляемых процессов автоматизированного производства. Источниками асинхронности могут быть непредсказуемые задержки в действиях управляемого оборудования, в коммутационных линиях, при использовании процессоров и т. д. Перестройка ГАПС сопровождается перестройкой алгоритмического и программного обеспечения, порядка использования ресурсов, протоколов локального взаимодействия процессов. В связи с такими качественными изменениями могут стать возможными нарушения системного взаимодействия как в пределах ОУВС, так и в пределах автоматизированной системы в целом. Проверка качественной корректности системных действий предполагает анализ следующих свойств процессов: живость, отсутствие тупиков (дедлоков), ограниченность, безопасность. Процесс (или оператор) называют *живым*, если существуют выполнимые условия его активизации. *Тупик* — это системная ситуация, связанная с взаимной блокировкой нескольких процессов, захвативших коммунальные системные ресурсы. В распределенных системах ресурсы выделяются несколькими контроллерами по запросам от процессов. Каждый контроллер в состоянии избежать локальных тупиков, но в силу параллелизма взаимодействующих процессов может возникнуть распределенный тупик. Предсказание тупика на этапе системного проектирования должно исключить тупиковую остановку процессов управления автоматизированным производством. Высокая ответственность облачной управляющей системы должна обеспечиваться также средствами обнаружения тупиков, возможность возникновения которых не была предсказана на этапе анализа корректности системных взаимодействий. Отрицательные последствия возникшего и обнаруженного тупика должны быть ликвидированы с минимальными потерями для производства. Специалист по технологии управления должен исключить возможность повторения такой тупиковой ситуации.

Термин *ограниченность* связывают с ограниченной емкостью накопителей изделий, системных очередей, буферов данных и т. п.

Если в системе невозможны ситуации, приводящие к переполнению системных очередей, буферов данных и других накопителей, то говорят, что условие ограниченности для системных действий выполнено. Нарушения ограниченности могут привести к потере сообщений или данных, переполнению накопителей изделий производства.

Термин *безопасность* связывают с устранением критических состояний системы. Безопасное движение роботов исключает их столкновение. Безопасность двух процессов ОУВС нарушается, если им одновременно удастся получить доступ к ресурсу, предназначенному для персонального использования.

Распараллеливание алгоритмов. Задача состоит в представлении алгоритмов управления в такой форме, которая обеспечивает быстрое и эффективное использование информации о взаимной зависимости операторов. Многогранность данной проблемы связана с разнообразием архитектур современных вычислительных систем, языков представления алгоритмов, способов преобразования алгоритмов из исходной формы в заданную.

Распределение работ между ресурсами распределенной ОУВС.

Распределение вычислительных работ между обрабатывающими средствами вычислительной системы или сети выполняется с целью обеспечения функций управления в режиме реального времени; повышения производительности обработки; эффективного использования процессорных мощностей системы. Планы распределения работ создают на основе использования информации о возможном параллелизме в выполнении операторов алгоритма. Задача распределения работ состоит в наилучшем приспособлении алгоритмов к заданной архитектуре ОУВС и представляет собой задачу синтеза распределенного алгоритма при заданной структуре (архитектуре) системы.

Поставленная задача может формулироваться также в виде задания на подбор смеси работ для данной сети процессоров и других ресурсов. Смеси подбирают на основе известного числа обращений к ресурсам, взаимного временного расположения и времен обслуживания запросов. Если к смешиваемым работам предъявляют различные требования по оперативности их обработки, то подбор смесей сопровождается выбором типа квантования или приоритетного обслуживания и распределением приоритетов по работам.

Комплексирование ОУВС. Состоит в приспособлении архитектуры ОУВС к целесообразной организации функций распределенного управления. Данная задача является обратной к сформулированной задаче распределения работ. По отношению к техническим средствам

ОУВС можно выделить два типа операций комплексирования: компоновка ОУВС — тактическая операция, связанная с существенной реорганизацией ее физической структуры при перестройках ГАПС; редактирование структуры ОУВС — оперативное действие, выполняемое в пределах имеющегося комплекса технических средств как правило автоматически. Задания на компоновку и редактирование формулируют путем подбора логической структуры системы, которую можно построить в пределах заданной физической структуры ОУВС, обеспечивая эффективное ведение процессов управления.

Математически данная задача ставится как задача синтеза структуры, наилучшим образом приспособленной к выполнению алгоритма, при заданных ограничениях на возможность комплексирования.

Оптимизация распределения файлов между запоминающими устройствами. Одной из критических задач проектирования распределенных систем является задача назначения файлов общей базы данных на устройства хранения информации. Устройства хранения могут территориально и логически принадлежать различным локальным процессорам либо могут составлять группы автономных ресурсов общего пользования. Можно выделить два источника задач назначения файлов. Первый источник связан с созданием распределенных баз данных управления. Распределенные базы данных обеспечивают эффективное использование носителей информации, создают условия для эффективного использования данных и повышения производительности системы при групповом управлении оборудованием, поскольку транспортирование управляющих программ и данных может быть по времени сопоставимым с их использованием. Вторым источником задач назначения файлов являются задачи планирования распределенной обработки. За распределением вычислительных операторов по процессорам должна следовать задача распределения наборов данных по ЗУ. Решение задачи о назначении файлов затем может повторно использоваться для улучшения результатов назначения операторов и т. д.

В большинстве случаев файлы желательно локализовать в пределах процессоров, которые их используют. Это предотвращает задержки на пересылки по линиям связи. Однако если много наборов данных разместить в одном устройстве памяти, характеристики ОУВС (или сети) могут ухудшиться из-за задержки в очереди к этому устройству хранения. Но, с другой стороны, назначив файлы на одно устройство, можно увеличить его использование.

Оптимизация назначения файлов может создавать условия для минимизации стоимостей хранения данных, обработки и обмена. Другие критерии учитывают производительность и могут обеспечивать

минимизацию времени ответа или максимизацию пропускной способности ОУВС или сети. Задача назначения файлов может предполагать выбор скоростей устройств и частоты их посещения. Эффект от решения задачи назначения файлов возникает только в том случае, когда она может выполняться в приемлемые сроки, обеспечивая получение результатов к моменту перестройки ОУВС гибкого производства.

Балансирование системы. Необходимы сбалансированность управляемой и управляющей систем, а также сбалансированность процессов и ресурсов систем. Сбалансированность характеризует термин *узкое место*. Узкое место — это такой ресурс системы (устройство или программа), относительное использование которого (загрузка) имеет наибольшее значение. Балансирование системы обеспечивает выравнивание загрузок аппаратных компонентов системы, в результате чего устраняется узкое место, и все узлы начинают действовать как ресурсы с одинаковой эффективностью использования. Балансирование системы можно выполнить подбором производительностей ресурсов ОУВС и объекта управления. Но возможен также и способ, состоящий в перераспределении маршрутов процессов между узлами системы. Эта процедура обеспечивает координацию маршрутов нескольких процессов таким образом, что гарантируется уменьшение взаимных помех между ними. В сложных технических системах несбалансированность нагрузки и производительности ресурсов допустима лишь для гарантирования надежности или высокого коэффициента готовности. Сбалансированность — необходимое условие эффективного использования ресурсов и высокой производительности системы.

Поддержание работоспособности. Сбои в системотехнических действиях могут обнаруживаться на основе анализа фактических состояний процессов и сопоставления их с допустимыми, которые определяются интерпретацией управляющей модели (см. рис. 7).

Недопустимые состояния должны корректироваться.

Обеспечение автоматизации оперативно-календарного планирования производства. Главная задача оперативно-календарного планирования — организация эффективного и согласованного во времени и пространстве движения изделий производства. Критерии качества планов: выполнение производственной программы при соблюдении директивных сроков выпуска продукции; рациональное использование оборудования, материальных и трудовых ресурсов; минимальное связывание оборотных средств и т. п. Эффективность производства может обеспечиваться разработкой точных подетально-

пооперационных планов-графиков (расписаний). Ручной способ построения графиков отличается сложностью и большой трудоемкостью.

Для обеспечения автоматизации отображения пространственно-временной картины производственного процесса календарный план-график можно получить способом динамической интерпретации сетевых управляющих моделей. Сетевая модель производства может быть основным машинным документом как для оперативного управления, так и для оперативного планирования хода производства. При таком подходе **календарные планы разрабатывают на алгоритмическом уровне воспроизведения процессов.**

Динамическая интерпретация сетевой модели позволяет автоматизировать построение подетально-пооперационных графиков движения изделий, графиков выполнения операций и загрузки рабочих мест, формализовать задачу оптимизации расписаний как задачу оптимизации параметров и структуры эталонной управляющей модели. Оперативное планирование заключается в оперативной корректировке описания управляющих моделей. Корректировка учитывает реальные условия, складывающиеся на объекте: состояние оборудования, изделий, технологических параметров. **Подсистемы оперативного планирования и управления ГАПС объединяются единой информационной базой и инструментальной системой программного обеспечения.** Планы-графики здесь — вторичные элементы, порождаемые из формальных, алгоритмических представлений хода производства. Подетально-пооперационные сетевые управляющие модели позволяют поднять оперативно-календарное планирование в ГАПС до уровня автоматизированных систем.

9.1.3. Распределенная облачная управляющая вычислительная система автоматизированного производства как объект системотехнического проектирования

Облачные управляющие системы — это множество универсальных и проблемно-ориентированных процессоров, соединенных шинами или коммуникационными сетями. При их использовании могут распределяться вычисления, данные, системное управление. Возможен очень широкий спектр типа распределенности и управления ею. Один из подходов к организации систем состоит в распределении действий между процессорами в соответствии с определенной функциональной

специализацией. В таких системах, например, выделяют процессоры: арифметико-логические, перерабатывающие информацию; управления базой данных, выполняющие операции коррективки, перезаписи, сжатия, создания баз данных и т. п.; обеспечивающие управление стыком с объектом (интерфейсные процессоры, преобразователи, процессоры ввода-вывода и т. п.); связные или коммуникационные, создающие условия для взаимодействия между другими процессорами. Кроме функционального распределения, в ОУВС организуется также распределение обработки в пределах каждой функции.

В распределенных управляющих системах, так же как и в централизованных ОУВС, используется режим мультипрограммной эксплуатации. Мультипрограммирование применяют в ОУВС в связи с необходимостью реализации принципа группового управления в автоматизированных производствах, а также для эффективного использования аппаратуры.

Мультипрограммная система представима в виде сети очередей и ресурсов, на которой перемещаются процессы.

Рассмотрим системотехнические особенности ресурсов, относя их к числу активных или пассивных. Один и тот же ресурс может быть отнесен к активным или пассивным в зависимости от способа его использования и точки зрения системотехника. К активным ресурсам относят центральный процессор, устройства ввода-вывода, другие средства, которые характеризуются определенным временем обслуживания; к пассивным шины и переключатели в мультипроцессорных системах, устройства управления (контроллеры), память, программы, данные. Типичным примером пассивного ресурса является основная память, которая сама не выполняет работу, но обеспечивает выполнение обработки процессором. Без активного ресурса невозможно использование пассивного. Время использования пассивного ресурса априорно не задается, а связывается с характеристиками активных ресурсов, к которым он придается. Отсутствие необходимых пассивных ресурсов может быть причиной приостановки процесса или блокировки использования активных ресурсов.

Рассмотрим несколько характерных примеров. На рис. 8 приведена модель системы группового управления, имеющая M объектов управления $ОУ$, обрабатывающий процессор $Пр$, основную память $ОП$, архивную память на дисках $АП$ и контроллеры дисков $К.Д$.

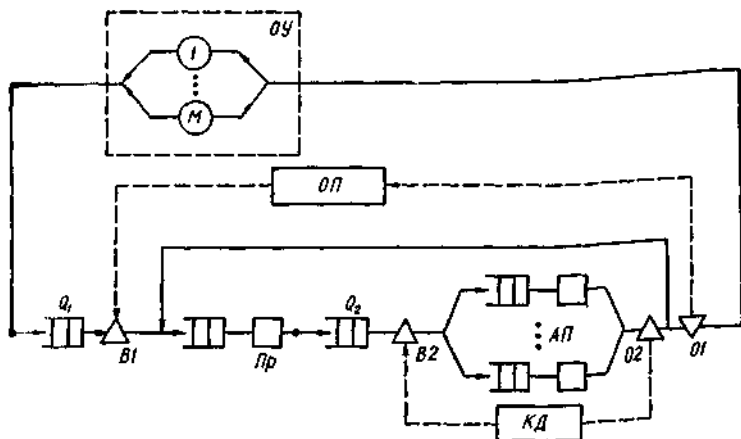


Рис. 8. Модель УВС группового управления с пассивными ресурсами

В модели отражены как аппаратные компоненты ОУВС, так и элементы операционной системы — очереди к ресурсам. Сплошные стрелки характеризуют пути движения и порядок обслуживания процессов. Обслуживание заявок выполняется в порядке их поступления в очередь. Для эффективного использования основной памяти организовано ее деление на разделы, число которых меньше коэффициента мультиобработки M . Выделение раздела памяти осуществляется по запросам, а освобождение — при переходе процесса из ОУВС группового управления к локальным управляемым объектам. Операции выделения и освобождения памяти обозначены штриховыми стрелками, а места этих действий в общей последовательности выполнения системных функций отмечены треугольниками $B1$ и $O1$. Если отсутствует свободный раздел, то запрос размещается в очередь Q_1 к памяти, как пассивному ресурсу. Время пребывания запроса в Q_1 определяется множеством характеристик тех элементов, которые на рисунке охвачены точками $B1$ и $O1$. Указать априорно время обслуживания в основной памяти в данном случае невозможно. Аналогично, если число контроллеров $АП$ меньше числа дисков, возникает очередь Q_2 к группе этих средств. Операции выделения и освобождения $КД$ помечены символами $B2$ и $O2$. В приведенном примере основная память и контроллеры архивной памяти — пассивные ресурсы, обеспечивающие работу активных ресурсов: процессора и дисков.

В процессе системотехнического проектирования распределенных систем необходимо учитывать эффекты блокировки использования ресурсов. Например, алгоритмы управления распределенной обработкой могут строиться таким образом, что происходит блокировка работы обрабатывающих процессоров, если возникает одна из следующих ситуаций: некоторая очередь или системный буфер заполнены до предела и не могут принять сообщение (в этом случае необходимо заблокировать работу процессора, который обслуживает заявки и направляет их в буфер); устройство может ожидать расписку о том, что сообщение невредимым поступило в накопитель, не выполняя новой работы (если за определенное время расписка не получена, необходима повторная передача сообщения).

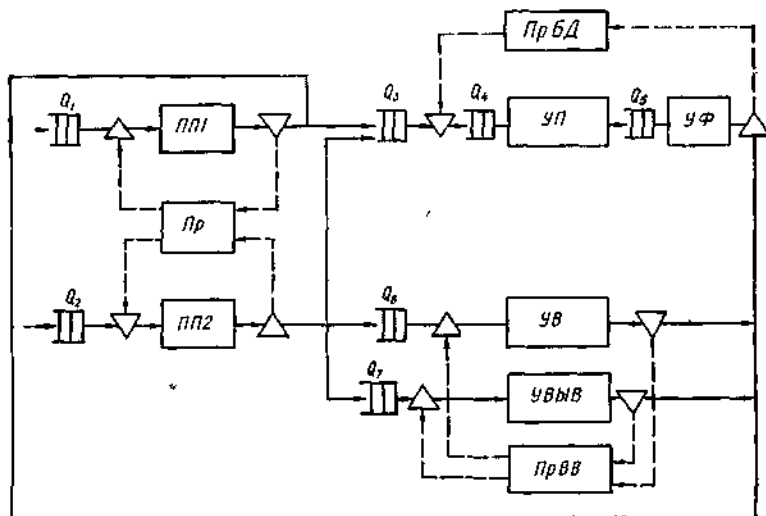
Рассматривая распределенную систему как объект системотехнического проектирования, необходимо учитывать многочисленные и разнообразные источники эффектов синхронизации процессов. Простейшая ситуация синхронизации возникает при совмещении операций на обрабатывающем процессоре и процессоре ввода-вывода, когда следующий цикл их параллельной работы может запуститься только после завершения предыдущих операций на обоих устройствах.

Источники синхронизации процессов могут находиться как в параллельных вычислительных алгоритмах прикладных задач, так и в структурах управляющих процессов.

Рассматривая системные элементы ОУВС, обычно исходят из того, что на аппаратной ресурсной среде происходит движение процессов, т. е. учитывают динамику программ. Однако при решении проектных задач возможен подход к системному анализу, имеющий основную ориентацию на анализ характеристик программных систем, которые здесь принимаются и качестве первичных элементов. Программные системы включают в себя операционную систему и прикладные программы. Их совокупность характеризуется производительностью (пропускной способностью), временем ответа или цикла и коэффициентами использования. Подобными характеристиками оценивают и аппаратные средства ВС. Однако элементами аппаратных систем являются процессоры, устройства ввода-вывода, а элементами программных систем — **программные модули**. Подобно различным способам организации межпроцессорных связей в ВС, разделения общей памяти вычислительных систем, ввода-вывода, можно рассматривать различные конфигурации программных систем, которые позволяют оценить их влияние на отношение стоимость-производительность или другую оптимизируемую характеристику. При анализе могут учитываться различные свойства языков программирования,

такие как использование процедур, наличие межмодульных связей и т. д. И, наконец, можно оценить, какая же секция является критической, узким местом программной системы.

Пример схемы прохождения процессов через программную систему приведен на рис. 9, где отражены как программные (или микропрограммные) ресурсы, так и вторичные по отношению к ним аппаратные ресурсы: обрабатывающий процессор *Пр*, процессор базы данных *ПрБД*, процессор ввода-вывода *ПрВВ*.



заданной точностью. ОУВС автоматизированного производства определяет действия автоматов, операторов автоматизированных и неавтоматизированных рабочих мест. Переход на выпуск новых изделий или изменение плановых заданий обеспечиваются перестройкой действий локальных систем управления, ОУВС и неавтоматизированных рабочих мест производства. В составе ОУВС можно выделить четыре типа функциональных элементов: процессоры (обрабатывающие средства); память (средства хранения информации); элементы коммутации и обмена; средства управления процессами и ресурсами (операционная система и аппаратные контроллеры). В составе объектов управления также можно выделить четыре типа элементов, функционально подобных элементам ОУВС: технологическое оборудование (обрабатывающие средства); промежуточные накопители и склады изделий (средства хранения); средства распределения и транспортировки; локальные (встроенные) системы управления. Таким образом, хотя ОУВС автоматизированных производств ориентирована на преобразование информации, а технологический процесс организует преобразование материи и энергии, их системные модели на уровне логико-время-количественных отношений могут строиться на основе функционально одинаковых элементов и, следовательно, единых принципов. В процессах технологии производства можно выделить те же особенности, которые свойственны процессам в ОУВС. Это — очереди, блокировки, синхронизация, активные и пассивные (первичные и вторичные) ресурсы, совместное использование и т. д. Например, для запуска процесса обработки нужен материал, инструмент и свободная обрабатывающая ячейка, этим средствам можно дать соответствующую ресурсно-процессную интерпретацию, аналогичную описанным интерпретациям ОУВС.

Модель системного уровня ГАПС как единого целого должна представлять динамику взаимодействующих информационно-алгоритмических процессов ОУВС и логико-время-количественных отношений в материальных потоках производства. Поскольку информационно-алгоритмические и материальные процессы взаимодействуют, синхронизируются, запускают друг друга, они могут интерпретироваться одинаковыми понятиями и отношениями, а анализ автоматизированного производства в целом и его компонентов (ОУВС и технологического объекта) на уровне логико-время-количественных отношений можно выполнять на базе единых математических и имитационных моделей системного уровня.

Естественной математической интерпретацией распределенных, асинхронных и мультипрограммных систем являются **сетевые модели**,

которые позволяют отражать распределенность структуры, сетевой характер взаимосвязей между процессами и ресурсами, а также между аппаратными и программными компонентами ОУВС, технологических процессов и ГАПС и целом. В связи с этим для решения задач системного проектирования ГАПС и ее ОУВС целесообразно привлечь сетевой лиализ.

Общность методов построения управляющих моделей ГАПС и моделей проектирования ОУВС, выражающаяся в использовании сетевых моделей, позволит сформировать единые средства автоматизированного проектирования и управления ГАПС.

9.1.4. Средства автоматизированного системотехнического проектирования ОУВС и их место в САПР ГАПС

Сложились два основных пути разработки АСУ технологическими процессами и САПР: индивидуальное проектирование и проектирование с использованием *типовых элементов*. В условиях массового применения предпочтителен второй из указанных способов. В гибких системах происходит слияние процессов производства и их проектирования. Объединение проектирования и производства целесообразно поддерживать *интегрированием* их технических средств, баз данных, персонала. Организационные проблемы и условия такого интегрирования весьма многообразны. Так, для улучшения условий технического обслуживания и эффективного использования средств автоматизации проектирования и ОУВС ГАПС аппаратный состав микропроцессорной техники САПР и ОУВС целесообразно реализовать на совместимых средствах, интегрированных в единый комплекс.

Единство информационной базы технологов производства, облачной управляющей системы и САПР призвано обеспечить описание характеристик технологии, извлечение из них информации о возможных алгоритмах управления технологией, представление информации о проектируемой ОУВС, создание программных средств управления на основе базы данных проекта, оценку эффективности и оптимизацию программно-технической реализации ОУВС и автоматизированного производства в целом. Существующие нормативы и традиции не всегда способствуют единству информационной базы описания технологии, базы данных управления ОУВС и базы данных проектирования САПР. Совмещение или создание условий для их эффективной интерпретации и

взаимодействия — важная задача системного подхода, призванного реализовать единую цель производства. Современное проектирование базируется на *модельном подходе*. Системотехническое проектирование автоматизированных производств требует отображения в модели функциональных, временных и пространственных взаимоотношений. Функциональные отношения описывают качественные и логические взаимосвязи и преобразования, развивающиеся во времени и пространстве. Цифровые управляющие системы и дискретные производства в модельном плане являются идентичными. Поэтому наряду с базами данных в САПР автоматизированных производств целесообразно выполнить интегрирование базовых моделирующих систем проектирования. Взаимосвязь основных подсистем САПР ГАПС с учетом принципов интегрируемости и специализации средств показана на рис. 10.

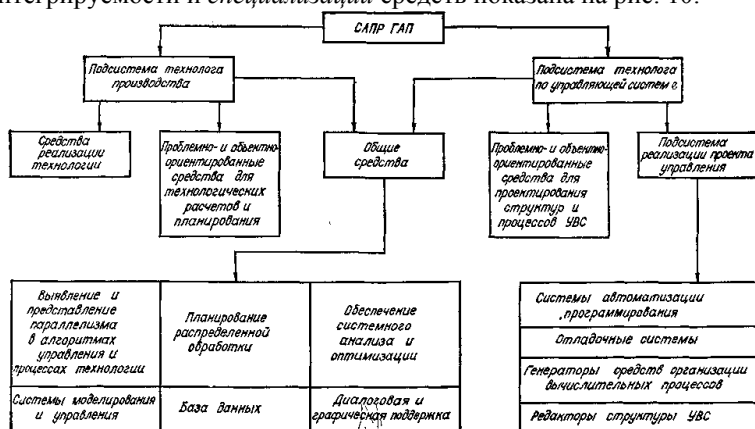


Рис. 10. Структурная схема элементов САПР, используемых при разработке управляющих моделей ГАПС

Специализация выражается в том, что здесь выделены такие группы потенциальных пользователей, как технолог производства и технолог по управляющей системе. Каждый из этих специалистов решает системотехнические задачи проектирования, однако ориентирует их на подготовку автоматизированного производства в пределах своего уровня компетенции — технологии производства и технологии управления соответственно. Профессиональные действия каждого из специалистов выполняются как для решения задач индивидуальной направленности, так и для достижения общей цели производства — экономичного и качественного выполнения плановых заданий. Технологи гибкого производства должны иметь в своем распоряжении:

проблемно- и объектно-ориентированные средства для технологических расчетов и планирования производства; проблемно -и объектно-ориентированные средства для проектирования структур и организации процессов ОУВС; общие средства. Продукты проектирования технологии производства и управления используются для настройки и эксплуатации объекта управления и управляющей системы. Продукты системного проектирования оформляются в заданной форме и передаются для реализации технологии и проекта управления. Реализация проекта управления обеспечивается программированием, отладкой и компоновкой проблемных управляющих программ, ориентированных на управление производством, и настройкой (генерированием) системных средств организации процессов в управляющей вычислительной системе. Перечисленные на рис. 10 действия объединены подсистемой реализации проекта управления и включают средства, обычно относимые к САПР программного обеспечения. В подсистему реализации проекта управления необходимо включить генератор средств организации вычислительных процессов, обеспечивающий настройку управляющих программ операционной системы распределенной ОУВС и позволяющий на стадии проектирования настраивать системное программное обеспечение на заданные режимы ОУВС с учетом конфигурации технических средств системы управления.

Рассмотрим организационную схему применения САПР. Технолог производства (цеха, участка) должен спланировать и отработать последовательность выполнения операций, сформировав технологическую цепочку из производственных агрегатов, промежуточных накопителей, средств транспортировки и распределения материальных потоков и потоков энергии. Выпуск конкретного изделия можно задать маршрутом (трассой) прохождения материальных потоков через технологическое оборудование производства. Технолог производства должен также задать режимные параметры оборудования, обеспечивающие выпуск изделия в соответствии с требованиями технического задания.

Технолог по управляющей системе ГАПС должен рассмотреть технологический процесс с позиций его взаимодействия с управляющим процессом и выработать технологию управления. Главная задача технолога по управляющей системе состоит в организации процессов управления в соответствии с логико-временными количественными условиями протекания управляемых процессов. Таким образом, организационная схема применения САПР должна предусматривать работу с ней технолога производства, затем

технолога по управляющей системе и завершаться исследованием системной работы совокупности «объект — управляющая система». Перестройка ГАПС должна предусматривать автоматизированный поиск (оптимизацию) решений по реализации управления, проверку его логической и технологической корректности, отсеивание нерациональных предложений.

Общие средства технологов производства и управления должны строиться на основе *инвариантов*, представленных математическими моделями, методами и алгоритмами для выполнения следующих функций: выявления и представления внутреннего параллелизма в алгоритмах управления и процессах технологии; планирования распределенной обработки (на производственном оборудовании и в распределенной ОУВС); обеспечения системного анализа и оптимизации взаимодействующих процессов и ресурсов.

Так как системотехническое проектирование ОУВС основывается на *сетевом анализе*, представление предметной области проектирования в базе данных САПР должно осуществляться на основе **сетевых структур данных**. В условиях САПР ГАПС сетевые структуры базы данных обеспечивают удобный для технологов производства графический формализм представления структуры производства и хода технологического процесса; удобный для технологов управления графический формализм представления алгоритмов, структур ОУВС, взаимодействий между ее процессами и ресурсами; применение современных графических дисплеев и графических методов в проектировании; эффективное распространение Р-технологии программирования на этап проектирования технико-программного комплекса ОУВС; широкие возможности в применении математических методов сетевого анализа и синтеза; сбалансированную ориентацию базы данных как на человека-проектировщика, так и на методы решения; сквозное использование унифицированных структур данных на всех этапах проектирования и непосредственно для организации управления и генерации программного обеспечения микропроцессорных элементов. Из различных средств обеспечения САПР на рис. 10 выделены средства диалоговой и графической поддержки (вопросы их организации рассмотрены дальше). Графический ввод в рассматриваемой системе проектирования реализует основные принципы Р-технологии.

В качестве *базового входного языка для описания проектных предложений и решений*, в САПР данной категории целесообразно выбрать язык, допускающий представление соответствующих частей проекта в графической форме, со спецификацией элементов

графической информации символами, числами и т. п. Дадим этому краткое обоснование.

Динамическое поведение распределенных систем реального времени является исключительно сложным. Взаимодействие параллельных процессов трудно предсказать в силу различных скоростей процессов в различных частях системы. Поэтому Фон-Неймановские последовательные модели вычислений, на которых базируются языки, во многом не соответствуют современным системам реального времени.

Именно поэтому разработан язык, допускающий как графическую, так и однозначно соответствующую ей реализацию в виде структур данных (см. раз. 9.2). Язык нагруженных графов использован в Р-технологии программирования. Введенный *язык нагруженных сетей* (ЯНС) обеспечивает представление параллелизма, синхронизации, конвейерности, распределенности процессов и ресурсов и т. д. Распределенные (параллельные) системы могут проектироваться **методами структурного проектирования**. Обеспечивается представление, документирование и анализ распределенных программно-аппаратных систем в интегрированных системах проектирования и управления.

Сетевое представление прочно укоренилось в электротехнике, электронике и проектировании ЭВМ для описания схмотехнических аспектов. Графическое представление существенно облегчает обмен решениями и идеями между людьми различных профессий и квалификации.

В отличие, например, от сетей Петри и других моделей, изображающих только потоки управления, **нагруженные сети** изображают также потоки данных, они хорошо структурируются и могут легко интерпретироваться сетью микропроцессоров. Вместе с тем языком нагруженных сетей легко интерпретируются параллельные структурные схемы, сети Петри, схемы потоков данных, сети очередей.

Использование ЯНС порождает новую методологию архитектурного проектирования распределенных систем реального времени методом моделирования.

Имитационное, численное и аналитическое моделирование сделали компьютерную науку теоретической и экспериментальной. Именно модели обеспечивают проведение экспериментов на основе собранных данных о явлении. Поэтому обязательным компонентом систем проектирования должны быть *средства измерений*, выполняющие сбор данных о тех характеристиках процессов и ресурсов, которые сложно оценить априорно. Измерительные средства целесообразно включить в работу для

идентификации параметров тех компонентов ОУВС и ГАПС в целом, значения которых вызывают несоответствие результатов имитационных экспериментов и натурных действий. Система идентификации моделей ОУВС и производства должна быть одной из составных частей САПР и реализоваться на основе программно-технических решений.

Важнейшей составной частью САПР является математическое и программное обеспечение моделирующих систем. Значительное число возможных конфигураций ОУВС, областей их применения, алгоритмов управления вычислительными процессами порождает большое число вариантов построения ОУВС. Чтобы решить задачу анализа или синтеза структуры вычислительных процессов, архитектуры ВС, необходимо располагать моделью ее функционирования. Модель может отображать процессы с различным уровнем детализации. В зависимости от требований системного анализа она может строиться в виде моделей массового обслуживания, сети Петри, логической автоматной структуры и т. д. Методика исследований и получаемые результаты определяются подходом к моделированию.

При проектировании ОУВС и для их перестройки в процессе эксплуатации возможно применение имитационных и аналитических моделей. Имитационные модели строят на базе алгоритмического представления процессов, аналитические — в виде систем алгебраических и дифференциальных уравнений, но и те и другие можно отнести к математическим моделям.

Успешное решение задач анализа и оптимизации без применения программных средств ограничивают лишь моделями, состоящими из единственного обслуживающего прибора, или несложными сетями массового обслуживания. Для исследования современных и перспективных мультипроцессорных систем управления и обработки информации необходимы средства программного обеспечения, пригодные для реализации математических моделей различного типа и уровня детализации процессов. **Только объединение этих средств в единую систему дает аналитику эффективный рабочий инструмент проектирования и исследований.**

САПР основаны на человеко-машинном способе выбора рациональных вариантов решений. Функции человека в САПР должны быть связаны с творческой работой и ответственностью за выбранные решения. На ЭВМ возлагается фудоемкая и рутинная работа. Компонуя моделирующие средства САПР ОУВС, необходимо учитывать, что ни один из способов модельных расчетов не является универсальным в применении. Имея набор нескольких методов, пользователь получает

свободу в использовании наиболее подходящего. Одни методы могут использоваться совместно с другими для преодоления сомнений или подтверждения результатов. Методы решения могут использоваться совместно (и гибридным способом).

Методологические принципы, учитываемые при разработке *моделирующих средств системы проектирования*, можно суммировать в виде следующих требований: простоты понимания на инженерном уровне; гибкости модельного формализма; структурируемости модели в соответствии с логической структурой управляемой системы; интегральности охвата моделируемой системы (аппаратуры, программных компонент, данных, динамики процессов); возможности отображения различных частей системы на различном уровне абстракции; единства и однородности базы данных; возможности комплексирования подмоделей различного типа в пределах одной модели; целесообразности того, чтобы модель, создаваемая для проектного анализа, могла использоваться и как управляющая модель, т. е. как средство организации управления автоматизированным производством; избыточности пакета моделирующих систем; полноты охвата задач проектирования и организации управления в избранной предметной области; использования единого входного языка высокого уровня, допускающего работу как за графическим, так и за алфавитно-цифровым дисплеем, и являющегося базой для интерпретации менее мощных языков, конструктивно порождаемых из него; возможности эффективного сбора результатов моделирования, наглядного изображения данных моделирования и динамики функционирования моделируемой системы; модульности, расширяемости .

Базовые составы средств аналитического и численного моделирования САПР перечислены на рис. 11, а.

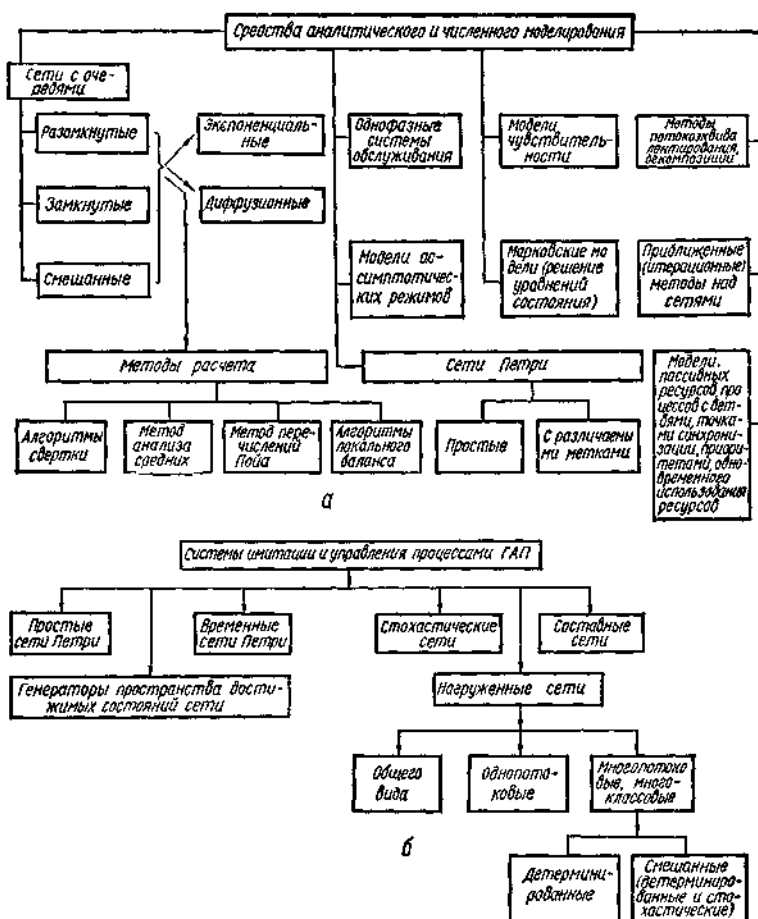


Рис. 11. Средства построения моделей проектирования и управления

Возможности моделей, особенности их реализации и применения рассмотрены дальше. Эти средства используют в основном на этапе автоматизированного проектирования. Системы имитации и управления (СИУ) (рис. 11, б) реализуются таким образом, чтобы они были инструментом как проектирования, так и программного управления ГАПС (см. раз. 9.2, 9.4). СИУ совместимы по входным данным, имеют как универсальные, так и функционально специализированные средства, создавая условия для эффективного использования времени и памяти САПР и ОУВС. Принципы

построения моделирующих систем позволяют создавать на их базе гибридные, имитационно-аналитические модели для проектирования и управления, формировать типовые модели конкретных ОУВС и базовых технологических режимов ГАПС.

В составе САПР должны быть средства автоматизированного *распараллеливания алгоритмов управления*, применять которые необходимо, например, для обеспечения режима реального времени в управляющей системе, а также при реализации распределенного управления. Необходимо также разработать средства *распределения (размещения) управляющих алгоритмов* (процессов). Эти средства должны предусматривать решение вопросов о комплексировании структуры ОУВС, планировании и организации управления. Названные средства по могут разрабатываться в отрыве от конкретных линий ОУВС, особенностей алгоритмов и критериев управления. Подготовка программ для распределенных вычислений может осуществляться на базе языковых средств параллельного программирования либо путем компиляции программ, подготовленных на традиционных последовательных языках. В настоящее время отсутствуют автоматические системы подготовки алгоритмов и программ для распределенных вычислений и управления. В составе рассматриваемой системы проектирования реализуются средства записи алгоритмов распределенных вычислений на языке нагруженных сетей и их обработки путем *автоматизированного распараллеливания* с участием человека-оператора. При создании распределенных и параллельных программ используют разнообразные правила выявления параллелизма и его использования. В их числе правила, основанные на методах распараллеливания линейных и циклических вычислений, организации макроконвейера, распараллеливания рекуррентных циклов и т. д. Для улучшения решений, предложенных проектировщиком, в САПР предусмотрено использование средств *планирования распределенных вычислений* и управления на основе некоторых приближенных методов и локальной дискретной оптимизации. Методы планирования распределенных вычислений должны обеспечивать выбор направления движения потоков информации, подбор смесей потоков в каналах обмена и смесей работ в процессорах, выбор производительности узлов, анализ чувствительности и т. д.

9.2. Сетевые модели проектирования облачных управляющих вычислительных систем

9.2.1. Изобразительные возможности сетевых моделей

Графические методы в проектировании и программировании получили широкое распространение в связи с широкими возможностями графических терминалов. Их внедрение позволяет преодолеть языковой барьер между пользователями и ЭВМ, вызванный различными аспектами, связанными со способами организации классических ЭВМ и средств их программирования. Установлено, что пользователи сопротивляются переходу на применение новых языков программирования. Перенос языковых и технологических средств программирования в практику системотехнических проектных работ часто не только не ускорял, но, наоборот, замедлял реализацию проектов. В связи с этим все более подкрепляется тенденция к развитию профессиональных компьютеров с графическими средствами общения. Переход от чисто текстового общения к общению с ЭВМ на базе графики и текста позволяет использовать те возможности проектантов, которые не реализовались при старых формах общения. Речь идет об обогащении зрительных образов общения и использовании потенциальных возможностей человека по их восприятию и обработке. Графические, структурные или диаграммные представления широко используются в технике. В изобразительных средствах для описания схем алгоритмов число разнообразных элементов чертежа минимально. Графика последовательных вычислительных процессов объединяет в себя чертеж и текстовые предложения. Такой же характер представления документов принят в электротехнике, электронике и других областях. Какой бы сложной ни была схема, ее целесообразно видеть. Алфавитно-цифровая спецификация дает дополнительную информацию к структуре системы или процессов. Такой же характер описания, объединяющий структуру и ее спецификацию, принят для представления процессов и структур ОУВС и ГАПС. При определении множества графических образов необходимо учитывать преемственность в соответствующих предметных областях.

Графические образы электротехники и электроники являются сетями-структурами. Графический образ алгоритма — это сеть-процесс. Проектирование ОУВС требует исследования как сетей-структур, так и сетей-процессов, причем отражение работы программно-аппаратных средств требует их объединения.

Рассмотрим модельные подходы сетевого анализа и условия их объединения на основе единого графического языка. Каждый из типов моделей охарактеризуем на примерах.

Многоклассовые детерминированные сети очередей. На рис. 12 приведена структура управляющих вычислительных систем.

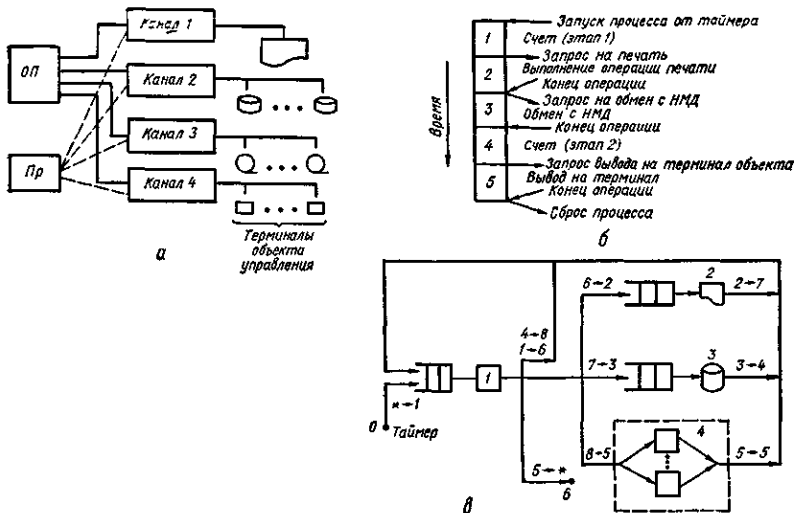


Рис. 12. Физическая структура ОУВС с каналами прямого доступа к памяти (а); схема выполнения программы (б) и модель (логическая структура) прохождения процессов через ОУВС (в)

Она характеризуется наличием канала прямого доступа в память со стороны накопителей, а также со стороны терминалов объекта управления. Процессор *Пр* может запустить операции в каналах, освободившись для выполнения других задач. Каналы обеспечивают отработку операций управления в накопителях, а также пересылку данных между ними и основной памятью. Запросы от терминалов объекта управления и обращения к ним со стороны ВС обслуживаются каналом 4. Глубина мультипрограммирования в ВС равна количеству терминалов объекта. Внутреннее управление системой реализуется аппаратными средствами (каналами, устройствами управления дисками и т. д.) и программными средствами операционной системы. Одной из ее функций является организация и обработка очередей к ресурсам системы. Очереди формируются к *Пр*, каналам, каждому из накопителей.

Почти все управляющие ЭВМ снабжены мультипрограммными ОС, поддерживающими одновременную обработку заданий путем разделения между ними системных ресурсов. Одна из функций мультипрограммных ОС состоит и максимизации совмещения прикладных задач пользователей при чередовании обработки на центральном процессе и периферийных средствах. Однако ОС требуются также системные ресурсы для выполнения своих функций. Такими ресурсами прежде всего являются основная и внешняя памяти, хранящие программные модули ОС, а также процессор, выполняющий системные программы. Эти часто непроизводительные по отношению к индивидуальным пользователям затраты могут быть существенными и составлять десятки процентов производительности центрального обрабатывающего устройства (ЦОУ). В большинстве систем функции управления отделяются от пользователя и объединяются в ОС. В связи с тем, что исполняемых функций достаточно много, ОС может быть большой, потребовать значительного расхода мощности системных ресурсов. Следовательно, важной задачей проектанта ОУВС является предсказание характеристик ВС с учетом затрат мощности ЦОУ и других средств на реализацию системного обслуживания.

Построим модель мультипрограммной ОУВС, отразив в ней взаимодействие системных и прикладных программ. Вычислительную систему будем моделировать сетью обслуживающих узлов, каждый из которых соответствует определенной функции ресурсов системы.

Процесс решения задачи (или просто процесс) можно представить его маршрутом через различные узлы сети. Если все задачи мультипрограммной смеси одинаковы и не взаимодействуют между собой, то процессы движутся между ресурсами по одинаковым маршрутам (цепям). Структура маршрута определяется макроструктурной прикладной программы. На рис. 12, б изображена схема действий процесса, рассматриваемая с позиций проблемного программиста. Действия пронумерованы цифрами 1—5, указывая на следующую их последовательность: счет 1 → печать → обмен с накопителем → счет 2 → вывод управления на объект. Макрокоманды запроса к печати, обмена с накопителем и объектом управления интерпретируются управляющими программами ОС, затем направляются к соответствующим каналам. Следовательно, выполнение прикладных и системных действий чередуется, и в очереди к процессору размещаются запросы на обслуживание различного типа.

Действия системы, функции программ и принадлежность их к *классам (типам)* обслуживания представим следующим списком:

1 — работа проблемной программы, этап счета 1;

2 — печать данных; 3 — обмен с накопителем; 4 — работа проблемной программы, этап счета 2; 5 — выдача информации на объект управления; 6—выполнение программы управления печатью (действия ОС), 7 — выполнение программы управления накопителем (действия ОС); 8 — управляющие действия на терминале объекта. Графическое представление путей процесса на сети очередей и ресурсов определяется с помощью двух ключевых управляющих элементов: правил перемещения в разветвлениях и межклассовых переходов на дугах, соединяющих обслуживающие центры (рис. 12, в).

Опишем правило перемещения в разветвлениях. Если процесс после выхода из обслуживающего узла принадлежит к классу r , то после обслуживания в узле i он направится к узлу j при условии, что дуга, соединяющая узлы i и j , помечена числом r . Если только одна дуга, исходящая из узла i , помечена признаком r , то дальнейший путь процесса детерминирован. Если в узле i невозможно обслуживание процесса классом r , то пометка r у выходящей из узла дуги может не встречаться.

Опишем правило задания межклассовых переходов. Структуру межклассовых переходов определяют из схемы действий на маршруте процесса. Если при переходе процесса между средствами системы происходит изменение класса обслуживания заявки, то у соответствующей связи (между узлами i и j) записывают, например, $r \rightarrow s$, обозначая переход из класса действий r в класс действий s .

Поясним описанное примером, представленным на рис. 12, в.

Изображенная здесь сеть узлов 1—4 и очередей к узлам 1—3 соединена дугами. Направление стрелок указывает возможные пути движения процессов. Процесс может занимать обслуживающий прибор или находиться в очереди к нему. Число приборов в узле 4 равно числу процессов в ОУВС, поэтому очередь здесь не возникает. Поскольку процесс (рис. 12, б) создается по запросу от таймера, в модель введен нулевой узел, являющийся источником заявок. На дуге, соединяющей узлы 0 и 1, изображена запись $* \rightarrow 1$, которая указывает на формирование начального значения атрибута-класса, который становится равным 1. Обслужившись в узле 1 (этап счета 1), процесс создает запрос на печать, обращаясь к действию класса 6. Оно выполняется процессором. Поэтому у дуги, соединяющей выход узла 1 и вход в очередь к нему, сделана пометка $1 \rightarrow 6$. Эта пара указывает как дальнейший путь процесса класса 1, так и новое значение его атрибута-класса. Если процесс класса 6 обслужен в узле 1, то согласно схеме действий процесса (см. рис. 12, б) он далее должен уйти в узел 2. Формирование этой части маршрута процесса производится с помощью пометки $6 \rightarrow 2$ у дуги между узлами 1 и 2.

Весь маршрут процесса можно представить последовательностью пар (i, r) , указывающих на его состояния в динамике $(0, *)$, $(1, 1)$, $(1, 6)$, $(2, 2)$, $(1, 7)$, $(3, 3)$, $(1, 4)$, $(1, 8)$, $(4, 5)$, $(1, 5)$, $(6, *)$. Первое и последнее состояния соответствуют источнику и стоку модели.

Если действия различных классов выполняются на одном и том же средстве, то им могут соответствовать различные значения времени обслуживания. В связи с этим временная спецификация модели должна представляться множеством значений $v = |v_{ir}|$, указывающих время выполнения действия типа r в узле i . В данном случае время первого этапа счета на процессоре v_{11} , время выполнения программы управления печатью на процессоре v_{12} и т. п. Некоторые элементы матрицы v здесь нулевые. Например $v_{15} = 0$, поскольку операция пятого типа выполняется только в узле 4 (операция взаимодействия с объектом управления реализуется в терминалах объекта).

Подводя итог, укажем, что модель процессов в данной ОУВС можно представить графической частью, спецификацией межклассовых переходов на дугах между узлами ($\rightarrow s$) и матрицей времен $v = |v_{ir}|$, где $i = 1, 2, \dots, N$; $r = 1, 2, \dots, R$. Внутреннее представление графической части в памяти ЭВМ предполагает хранение матрицы межузловых связей и описателя типов обслуживания в узлах. Поскольку эта матрица разрежена, целесообразно использовать эффективные способы ее хранения. На рис. 12, в приведены два типа узлов: узлы 1, 2 и 3 — с обслуживанием в порядке поступления (FIFO) и узел 4 — типа «задержка» (D). Возможны узлы и других типов. Таким образом, детерминированные многоклассовые модели обеспечивают естественное реалистичное представление взаимодействующих проблемных процессов, ресурсов и операционной системы (программ и очередей) на системном уровне отображения.

Стохастические сети с очередями. Сеть-структуру здесь описывают так же, как и при представлении детерминированных процессов.

Порядок выполнения этапов обслуживания вдоль трассы процесса не указывают, аппарат классов и межклассовых переходов отсутствует. Время обслуживания определяется в пределах ресурса и не дает представления о трудоемкости на разных этапах обслуживания проблемных программ. В такой модели невозможен прямой учет управляющих программ ОС.

Графическая часть модели специфицируется векторами времен обслуживания в узлах, типами обслуживающих приборов, а также стохастической матрицей $P = |p_{ij}|$, описывающей вероятности межузловых переходов процесса. К описываемой группе моделей относятся сети с очередями, процессы в которых содержат лишь элементы случайности. В естественном виде такие сети возникают,

если только часть маршрута процесса предопределена, т. е. когда элементы случайности присутствуют в алгоритмах управления или обработки, либо, например, задаются: вероятность отказа какого-либо ресурса системы; вероятность некоторого состояния объекта управления или управляющей системы, определяющая порядок использования ресурсов.

Вообще элемент случайности возникает из-за неопределенности в характере использования ресурсов либо вследствие грубости модельного представления.

Рассмотрим многоклассовую сетевую модель ВС с элементами случайности (рис. 13).

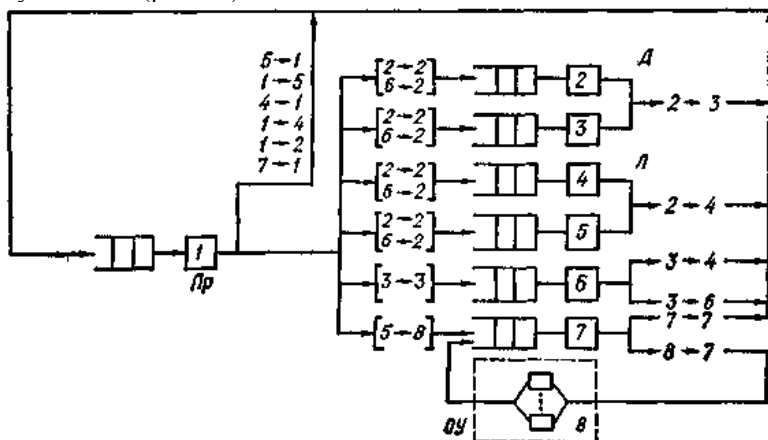


Рис. 13. Многоклассовая сетевая модель с элементами случайности: 1 — выполнение счета проблемной задачи; 2 — работа супервизора ввода-вывода, подготовка накопителя информации к вводу-выводу, 3 — подготовка и передача данных через канал, 4 — действия диспетчера центрального процессора, 5 — прочие системные операции, 6 — действия супервизора памяти; 7 — создание задачи управления, 8 — вывод информации на объект, действие объекта управления

Характеристики программ и режимов обслуживания пронумерованы числами.

Модель процессора представлена узлом 1. Модели ввода-вывода данных с накопителями Д описаны двумя последовательными этапами: подвод и поиск информации представлены узлами 2 и 3, а обмен — через канал — узлом 6. В моделях Л 4 и 5 учтено время подвода,

которое значительно больше времени обмена. Канал Л представлен узлом 7, а объект управления — узлом 8.

Рассматриваемая сетевая модель *замкнута* относительно множества процессов, в отличие от сети на рис. 12. Опишем ее подробнее. Пусть в результате действий объекта сформирован запрос. Дождавшись обслуживания в канале 7 и затратив на него время v_{17} , запрос направляется в очередь к *Пр*, где после его обслуживания в течение времени v_{17} создается задача управления. Создание задачи завершается ее запуском, что представляется переходом из класса 7 в класс 1 и возвратом процесса в конец очереди к *Пр*. Если при выполнении программы возникает запрос к базе данных, то процесс переходит и системный класс 2, и после получения *Пр* может быть направлен к одному из накопителей информации. Поэтому на путях от узла 1 к узлам 2—5 необходимо указать вероятности перехода для заявок класса 2. Это и есть тот элемент случайности, который складывается в результате динамики процессов взаимодействия с базой данных. Детализируем обработку запроса к супервизору ввода-вывода, обслуживающему диски. Когда запрос достигнет начала очереди к диску, он получит обслуживание, связанное с позиционированием головок чтения-записи на необходимые цилиндр и сектор. После выполнения этих действий процесс перейдет из системного класса 2 в системный класс 3, возвращаясь в *Пр*, которое обеспечивает создание данных, передаваемых канальной программе. Во время обслуживания классом 3 на *Пр* включается время диспетчеризации центрального процессора, а также время создания канальной программы чтения-записи. Завершив эти системные функции, системное задание класса 3 следует в канал и получает обслуживание, затрачивая время, необходимое для полного ввода данных с диска в основную память. Далее задание переходит в системный класс 4 и возвращается в *Пр*, где получает обслуживание, необходимое для выхода из прерывания по обращению к канальной программе и планирования возврата к проблемной программе. Завершив выполнение системных функций класса 4, задание возвращается на центральный процессор в класс обслуживания с номером 1. Таким образом завершается цикл выполнения системных операций по организации обмена данными между основной памятью и диском

В рассматриваемой системе возможна ситуация, когда задание класса 3, завершив обслуживание в канале 6, вызывает программу распределения памяти. Этот путь возникает при обмене данными, организованных цепочкой. После обслуживания в *Пр* процесс изменяет класс ($6 \rightarrow 2$) и имитирует операцию ввода данных, завершающую обмен между основной и внешней памятью. Уровень детализации

процессов в создаваемой модели во многом определяется исходными данными, которыми располагает конструктор, и глубиной проникновения в системные процессы.

Рассматривая межклассовые переходы процесса на дугах из узла 6 в узел 1, можно видеть еще один элемент неопределенности трассы процесса, описываемый вероятностными характеристиками. Если в результате ввода установлено, что необходимый массив данных введен полностью, реализуется межклассовый переход $3 \rightarrow 4$, иначе следует переход $3 \rightarrow 6$ и вызов супервизора памяти. Элемент случайности необходимо также описать в циклическом пути, охватывающем узел 1. Таким образом, для разрешения маршрутных неопределенностей в многоклассовых сетях необходимо использовать вероятностные условия, привязываемые не только к паре узлов (i, j) , как это имеет место в стохастических сетях, но и к классам заявок на этих связях.

Сеть Петри. Классические сети очередей не используются для описания и точного математического анализа процесса с точками синхронизации и ветвления. Сети Петри предложены как инструмент для описания асинхронных взаимодействий в процессах и структурах именно с такими эффектами. Структура сети Петри представляется ориентированным двудольным графом (биграфом). Множество вершин V двудольного графа разбивают на два непересекающихся подмножества (доли) T и P , такие что $V = T \cup P, T \cap P = \emptyset$.

Вершины из T и P соединены между собой ориентированными ребрами (дугами). Поскольку ребрами соединены вершины из P с вершинами из T и наоборот, все простые циклы биграфа имеют четную длину. Для представления процессов, являющихся динамическими объектами систем, с графом связываются подвижные элементы, которые называют метками (марками). Распределение марок по вершинам сети называют ее маркированием.

Приведем некоторые определения и обозначения теории сетей Петри, используемые в данном разделе. Сеть Петри — это пятерка $N = \{T, P, I, O, M_0\}$, где P — конечное множество позиций; $P = \{p_i\}$, $i = 1, 2, \dots, n$; T — конечное множество переходов, $T = \{t_j\}$, $j = 1, 2, \dots, m$; $I : T \times P \rightarrow \{0, 1\}$ — функция следования; $O : P \times T \rightarrow \{0, 1\}$ — функция предшествования; $M_0 : P \rightarrow Z_0$ — начальное маркирование, задающее начальное распределение меток по позициям сети; Z_0 — множество неотрицательных целых чисел.

Функции предшествования и следования задают множества одинарных дуг $(t_j p_i)$ и $(p_i t_j)$, связывающих вершины биграфа. Дуги, предшествующие позиции p_i , содержатся в множестве

$$I(p_i) = \{(t_j p_i) \mid I(t_j p_i) = 1\},$$

а дуги, предшествующие переходу t_j , — в множестве

$$I(t_j) = \{(p_i t_j) | O(p_i t_j) = 1\}.$$

Здесь значение $I(t_j p_i)$ указывает на наличие ребра, соединяющего переход t_j и позицию p_i , и, аналогично, значение $O(p_i t_j) = 1$ указывает на дугу, направленную от позиции p_i к переходу t_j . Подобно, дуги, следующие из p_i и t_j , представим множествами

$$O(p_i) = \{(p_i t_j) | O(p_i t_j) = 1\};$$

$$I(t_j) = \{(t_j p_i) | I(t_j p_i) = 1\}.$$

Входные позиции перехода t_j объединяются в множества его предшественников

$$PRE(t_j) = \{p_i \in P | O(p_i t_j) = 1\},$$

а выходные позиции — в множества позиций — последователей

$$POST(t_j) = \{p_i \in P | I(t_j p_i) = 1\}.$$

Аналогично, предшественниками и последователями позиций p_i являются переходы, объединяемые множествами:

$$PRE(p_i) = \{t_j \in T | I(t_j p_i) = 1\};$$

$$POST(p_i) = \{t_j \in T | O(p_i t_j) = 1\}.$$

Все перечисленные множества можно найти из отображений I и O . Маркирование сети представляется вектором

$$M = \begin{pmatrix} m(p_1) \\ m(p_2) \\ \vdots \\ m(p_n) \end{pmatrix},$$

где $m(p_i)$ — число меток в позиции p_i .

Говорят, что переход t_j возбуждается маркированием M и может сработать, если все входные позиции перехода t_j имеют метки.

Формальная запись условия возбуждения

$$\forall p_i \in P [m(p_i) - O(p_i t_j) \geq 0]$$

сравнивает число меток в p_i с числом дуг между p_i и t_j . Если меток $m(p_i)$ больше числа дуг $(p_i t_j)$, то $m(p_i) - O(p_i t_j) \geq 0$. Срабатывание перехода t_j приводит к тому, что его позиции-предшественники $PRE(t_j)$ теряют по одной метке, а каждая из позиций $POST(t_j)$ получает по одной метке. Если в сети задано начальное маркирование, при котором

хотя бы один переход возбужден, то в ней начинается движение меток. Множество достижимых маркирований обозначим $R(N)$. Если в сети N одновременно возбуждено несколько переходов, то порядок их срабатывания не определен, и, следовательно, может быть представлено несколько последовательностей срабатывающих переходов t .

Языком сети, или языком, порождаемым сетью $L(N)$, называется множество всех конечных последовательностей срабатывающих переходов при заданном начальном маркировании. Язык $L(N)$ представляет множество всех возможных последовательностей действий, выполняемых параллельными процессами.

Семафорные механизмы широко используются для организации взаимодействия программ ОС. Они используются для запрещения доступа процессов в критические секции, учета ресурсов и синхронизации процессов

Рассмотрим пример описания взаимодействующих процессов с помощью сети Петри. Пусть в области общей памяти системы выделен участок длиной N единиц, который используется для хранения запросов на выполнение задач. Процессы записи в очередь и выбора запросов из нее могут происходить независимо (асинхронно).

Описатель очереди содержит четыре элемента: n и l — число заполненных и свободных ячеек очереди; α и β — адреса начального и конечного элементов очереди. Алгоритмы записи в очередь и выборки запросов R из очереди приведены на рис. 14.

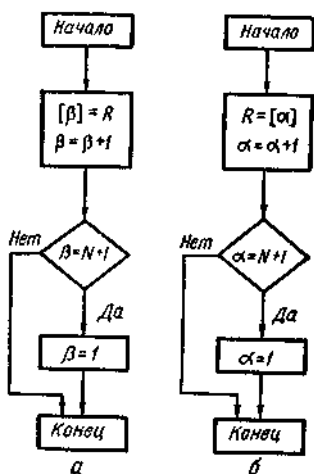


Рис. 14. Алгоритмы постановки и выбора из очереди: *а* — в очередь, *б* — из очереди

Запись вида $[\beta] = R$ представляет операцию размещения запроса в ячейку очереди с номером β , а запись вида $R = [\alpha]$ — считывание содержимого ячейки α и использование его в качестве информации о запросе. Если с очередью может работать несколько программ В ОЧЕРЕДЬ и ИЗ ОЧЕРЕДИ, то для организации корректного ее использования необходимо исключить их одновременный доступ к области хранения запросов. В противном случае становятся возможными ошибки. Например, два потребителя могут выбрать один и тот же запрос, не выбрав из очереди следующий за ним. Для недопущения подобной ситуации параллельные процессы В ОЧЕРЕДЬ и ИЗ ОЧЕРЕДИ могут синхронизироваться с помощью семафорных примитивов P и V . Пусть $S = \{0,1\}$ — семафор, ограничивающий доступ процесса в некоторую область действий. Операции над S : $V(S)$ — переменная S увеличивается на 1, если $S = 0$; $P(S)$ — если $S = 1$, то S уменьшается на 1, и процесс переходит к выполнению следующего действия. В противном случае процесс блокируется, ожидая состояния семафора $S = 1$. Примитив $P(S)$ запрещает доступ процесса в критическую область, а $V(S)$ — разблокирует.

Подобно определяются операции над считающими семафорами l и n : $P(l)$ — если $l > 0$, то l уменьшается на 1 и процесс продолжается, иначе он блокируется в текущей точке; $V(l)$ — если $l < N$, то l увеличивается на 1 и процесс продолжается.

Программа, запускающая параллельные процессы записи-чтения в очередь, имеет вид:

НАЧАЛО

$l = N; \quad n = 0; \quad S = 1.$

ПАРАЛЛЕЛЬНО ВЫПОЛНИТЬ M_1, M_2 ;

M_1 : **НАЧАЛО** P : **СОЗДАНИЕ ЗАПРОСА**;

$P(l); \quad P(S); \quad \text{В ОЧЕРЕДЬ};$

$V(S); \quad V(n); \quad \text{ПЕРЕЙТИ К } P$;

КОНЕЦ

M_2 : **НАЧАЛО** R : $P(n); \quad P(S); \quad \text{ИЗ ОЧЕРЕДИ};$

$V(S); \quad V(l); \quad \text{ВЫПОЛНЕНИЕ ЗАПРОСА};$

ПЕРЕЙТИ К R ;

КОНЕЦ

КОНЕЦ

Сеть Петри, описывающая эти параллельные процессы, приведена на рис. 15.

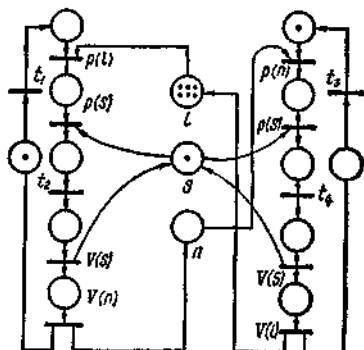


Рис. 15. Сеть Петри, описывающая параллельные процессы управления буферированием:

t_1 — СОЗДАНИЕ ЗАПРОСА; t_2 — В ОЧЕРЕДЬ;

t_3 — ВЫПОЛНЕНИЕ ЗАПРОСА; t_4 — ИЗ ОЧЕРЕДИ

Содержательный смысл позиций и переходов отражен у этих элементов сети.

На рис. 16 изображена модель действий системы, имеющей процессор, устройство ввода-вывода (УВВ) и накопитель.

В этой модели метки во входной очереди представляют запросы.

Переход t_x срабатывает, если есть запрос, свободный процессор и лента с данными. Действия по использованию процессора и УВВ могут выполняться параллельно, что отражено параллельными ветвями с t_2 и t_3 . Счет 2 может выполняться по за першению счета 1 и окончанию ввода-вывода, что определяется условием возбуждения перехода t_{im} . Освобождение процессора, накопителя и уход одной метки из сети происходят при срабатывании t_i . После перемотки ленты (переход t_6) может приняться для обслуживания следующая заявка, представленная меткой во входной очереди.

Теория сетей Петри служит одной из составляющих общей теории сетевых моделей. К недостаткам сетей Петри носится отсутствие времени в определении ее динамического функционирования.

Большинство теоретических исследований по сетям Петри исключает из рассмотрения время.

На рис. 16 изображена модель действий системы, имеющей процессор, устройство ввода-вывода (УВВ) и НМЛ.

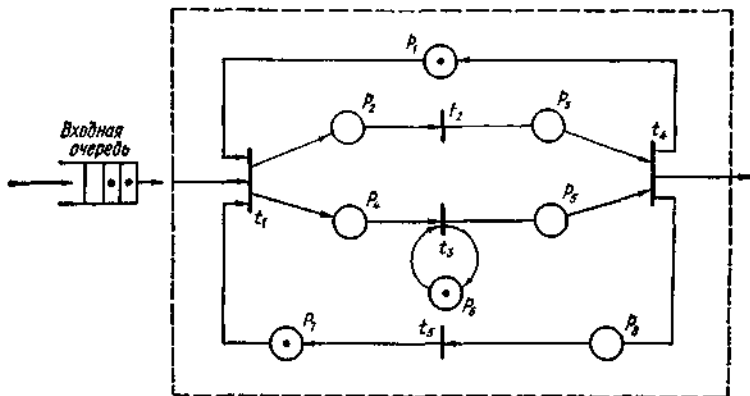


Рис. 16. Модель системных взаимодействий в виде сети Петри
 t_j — действия системы, t_1 — ввод с накопителя, t_2 — счет 1; t_3 — выполнение ввода-вывода; t_4 — счет 2; t_5 — перемотка ленты; p_i — управляющие переменные ОС; p_1 — процессор свободен, p_2 — запрос счета, p_3 — конец счета; p_4 — запрос ввода; p_5 — конец ввода; p_6 — УВВ свободно; p_7 — подвод ленты выполнен; p_8 — запрос к накопителю

В этой модели метки во входной очереди представляют запросы. Переход t_1 срабатывает, если есть запрос, свободный процессор и лента с данными. Действия по использованию процессора и УВВ могут выполняться параллельно, что отражено параллельными ветвями с t_2 и t_3 . Счет 2 может выполняться по завершению счета 1 и окончанию ввода-вывода, что определяется условием возбуждения перехода t_4 . Освобождение процессора, накопителя и уход одной метки из сети происходят при срабатывании t_4 . После перемотки ленты (переход t_6) может приняться для обслуживания следующая заявка, представленная меткой во входной очереди.

Теория сетей Петри служит одной из составляющих общей теории сетевых моделей. К недостаткам сетей Петри относится отсутствие времени в определении ее динамического функционирования. Большинство теоретических исследований по сетям Петри исключает из рассмотрения время. Практика требует учета времени, хотя при этом происходит существенное усложнение в методах анализа сетей и как бы переход от алгебраических уравнений — к дифференциальным. Сети Петри, приведенные в примерах, описывали: параллелизм и синхронизацию процессов; взаимодействие процессов и ресурсов, представленных метками; независимые действия, представленные

переходами t ; накопители информации (ячейки и буферы) и состояния процессов, представленные позициями. Существуют два базисных метода анализа сетей Петри. Один из них основан на использовании графа достижимых маркировок (диаграммы состояний сети), другой — на применении уравнения состояний и методов линейной алгебры.

Временная сеть Петри позволяет более реалистично отражать процессы и системы. Во временных сетях каждому переходу сопоставляется время τ_i . Если переход возбуждается, то метки, вызвавшие запуск перехода, покидают входные позиции $PRE(t_i)$. Порождение меток в выходных позициях $POST(t_i)$ происходит через время τ_i . Формальное определение временной сети: $TN = \{N, \tau\}$, где N — сеть Петри; $\tau: T \rightarrow R_0$ — функция времен срабатывания, сопоставляющая каждому переходу постоянное время срабатывания; R_0 — множество неотрицательных рациональных чисел.

Сеть с приоритетами переходов. Формальное определение: $PRN = \{N, PR\}$, где N — сеть Петри; PR — отношение приоритетности (порядка), задаваемое на множестве переходов T и определяющее порядок потребления меток возбужденными переходами в условиях конфликта за метку.

Временная сеть с приоритетами переходов объединяет моменты, описанные в рассмотренных выше классах сетей. Формальное определение: $PRTN = \{N, \tau, PR\}$.

Рассмотрим пример. На рис. 17 изображена модель взаимодействия между узлами сети ЭВМ, описывающая протокол связи с исправлением ошибки.

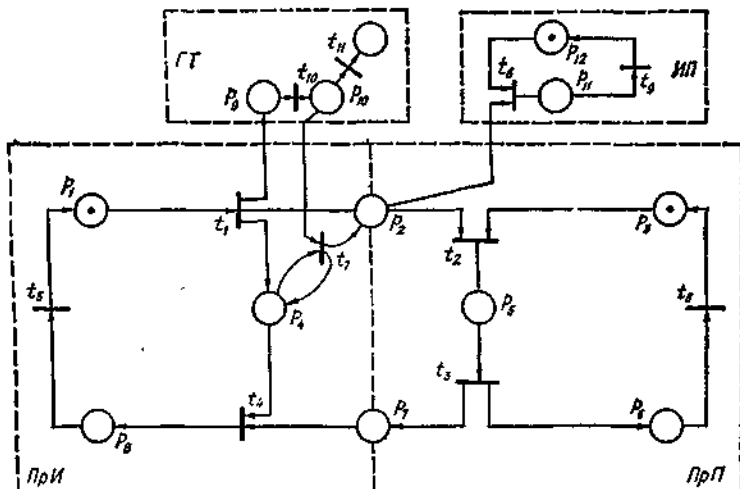


Рис. 17. Модель взаимодействия между двумя процессорами (протокол с исправлением ошибки обмена):

ПрИ — процессор-источник; *ПрП* — процессор-приемник; *ГТ* — генератор тайм-аута, *ИП* — источник помех; t_1 — передача сообщения в буфер обмена; t_2 — прием сообщения, t_3 — посылка подтверждения о приеме данных; t_4 — прием подтверждения, t_5 — обработка данных в *ПрИ*; t_6 — обработка в *ПрП*; t_7 — повторение передачи, t_8, t_9 — переходы модели источника помех; t_{10}, t_{11} — элементы *ГТ*; p_1 — конец обработки в *ПрИ* и запрос действия t_1 , p_2 — буфер сообщения; p_3 — *ПрП* готов принять сообщение и запрашивает t_2 ; p_4 — *ПрИ* ожидает подтверждения, p_5 — завершение действия t_2 и запуск t_3 ; p_6 — запрос t_6 ; p_7 — буфер подтверждения; p_8 — запрос t_8 ; p_9, p_{10} — позиции *ГТ*; p_{11}, p_{12} — позиции *ИП*

Протокол предусматривает подтверждение приема посланного сообщения, повторение передачи при потере сообщения и соответствует упрощенной версии реального протокола. Содержательный смысл, соответствующий состояниям и переходам сети Петри, имеет определенные значения.

Переходам t_i сопоставлены времена τ_i , отражающие длительность выполнения соответствующих действий в системе обработки и обмена, генераторах помех и тайм-аута. Отношения приоритетности введены для двух пар переходов — $(t_2 t_8)$ и $(t_7 t_{11})$, поскольку только по отношению к ним может возникнуть конфликт за метку. Они имеют вид: $PR(t_8) > PR(t_2)$ и $PR(t_7) > PR(t_{11})$.

Передача сообщения процессором-источником порождает буферирование копии сообщения и запуск генератора тайм-аута, послыску сообщения в канал, а также формирование условия подтверждения о приеме, что отражается появлением меток в p_2, p_4, p_9 . Если сообщение в канале p_3 исчезает благодаря действию источника помех (что отражается возбуждением t_8), то не приходит подтверждение приема (метка в p_7 не появится), и через время тайм-аута (связанное с t_{10}) произойдет повторная выдача сообщения в канал (что отражается срабатыванием t_7). С t_9 связан интервал времени между потерей в канале очередного сообщения и возникновением условия для потери следующего сообщения. С t_{10} связано время тайм-аута, через которое организуется повторная посылка потерянного сообщения. Если через t_{10} в p_4 нет метки, то не создадутся условия возбуждения t_7 , и метка из p_{10} покинет генератор тайм-аута.

На рис. 18 изображен граф допустимых маркирований элементов модели процессора-приемника и процессора-источника.

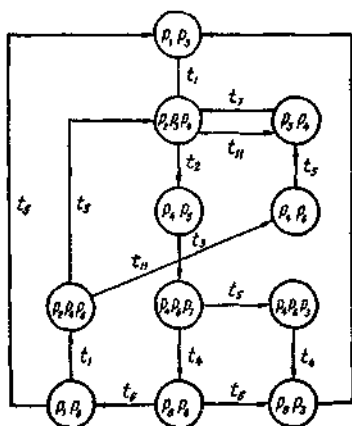


Рис. 18. Граф достижимых состояний (допустимых маркирований) с теряемой меткой

В вершинах графа состояний перечислены наборы позиций, в которых одновременно находятся метки-сообщения (при начальном маркировании M_0 , изображенном на рис. 17). Ориентированные дуги указывают переходы между состояниями системы, а пометки у дуг указывают на переходы t , срабатывание которых вызывает данное изменение состояния.

Приведенный пример показывает, что вместо длинных словесных описаний протокола взаимодействия достаточно иметь графическое изображение модели, спецификацию времен и приоритетов,

обеспечивая тем самым всю информацию о возможных путях и свойствах процессов.

Пренебречь временными свойствами сети возможно только в том случае, если исключение τ не отразится на структуре переходов диаграммы состояний сети. В модели на рис. 17 невозможно пренебречь временными характеристиками, не исказив представление возможных свойств объекта.

Раскрашенные сети являются естественной интерпретацией реальных систем, аналогичной многоклассовым сетям с очередями. Однако эти многоклассовые сети не допускают представления эффектов размножения и синхронизации, отражаемых с помощью переходов в сетях Петри.

Меткам раскрашенной сети Петри приписывают атрибуты, которые называют цветами. Правила возбуждения переходов дополняются условиями, предполагающими выбор меток определенных цветов из позиций $PRE(t_j)$. Срабатывание переходов сопровождается посылкой в полиции $POST(t_j)$ меток с задаваемыми значениями цвета. На рис. 19 приведена модель процессов в двухстаночной системе, обслуживаемой роботом, из которой множество атрибутов-цветов, связываемых с метками и дугами сети, принято $C = \{\lambda, c_1, c_2, c_3, c_4\}$.

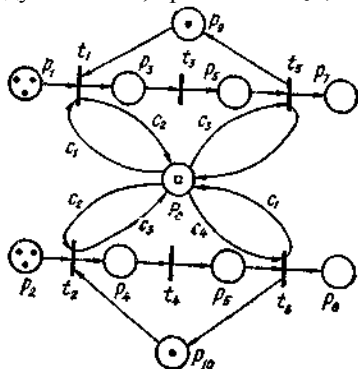


Рис. 19. Модель процессов в двухстаночной роботизированной системе, представленная раскрашенной сетью:

t_1, t_2 — операции загрузки станков, t_3, t_4 — обработка деталей; t_5, t_6 — разгрузка станков; p_0 — позиция робота, выполняющего операции загрузки-разгрузки станков; p_1, p_2 — бункеры для заготовок; p_3, p_4 — конец загрузки, запуск обработки; p_5, p_6 — конец обработки, запрос разгрузки; p_7, p_8 — бункеры для готовых изделий; p_9, p_{10} — двоичные семафоры, исключающие попытки загрузить еще не разгруженные станки

В сети раскрашена метка, имитирующая работу робота, и дуги на маршруте робота. Прочие дуги, а также метки-изделия и метки-управления не раскрашены. Начальный цвет метки в p_0 равен c_1 и дуга $(p_0 t_1)$ окрашена в c_1 . Поэтому условие возбуждения выполнимо для перехода t_1 . После срабатывания t_1 в p_0 метка возвращается, уже имея цвет c_2 , поскольку дуга $(t_1 p_0)$ раскрашена в цвет c_2 . Раскрашивание дуг обеспечивает возбуждение только одного из переходов, следующих за p_0 , при любом состоянии системы. При данной раскраске маршрут робота представлен последовательностью срабатывающих переходов $t_1 t_2 t_5 t_6$, т. е. происходит загрузка первого станка, затем загрузка второго, после чего следуют их разгрузки, и цикл повторяется. На рис. 20 приведена сеть, описывающая движение совокупности транспортных средств роботизированного комплекса.

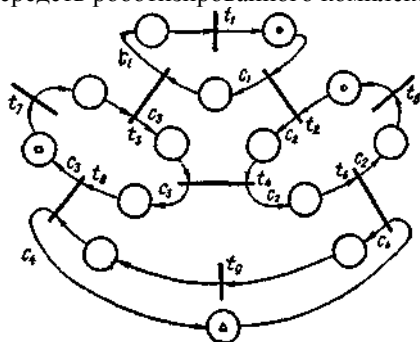


Рис. 20. Модель синхронизируемого движения транспортных средств роботизированного комплекса

В местах пересечения маршрутов роботы ожидают прибытия других, чтобы выполнить передачу груза. Множество цветов задано: $C = \{\lambda, c_1, c_2, c_3, c_4\}$. Цвет метки на рисунке обозначается одной из форм $\{\bullet, \circ, \square, \triangle\}$ и соответствует конкретному роботу. Раскраска дуг, следующих за переходами, фиксирует маршруты множества меток-роботов. При данной на рис. 20 раскраске представляется движение роботов по своим замкнутым маршрутам, проходящим через непересекающиеся множества позиций. На рис. 21, в раскрашенная сеть использована для представления мультипрограммного выполнения алгоритма с ветвями и точкой синхронизации в ВС с процессором и УВВ.

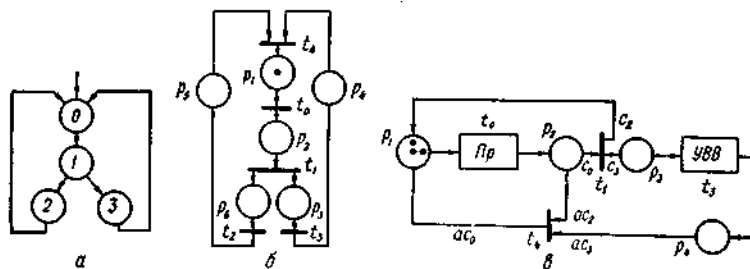


Рис. 21. Граф алгоритма (а); биграф алгоритма (б) и модель мультипрограммного выполнения алгоритмов, представленная раскрашенной сетью (в)

На *Пр* и *УВВ* параллельно могут выполняться операторы 2 и 3, завершение которых является условием выполнения на *Пр* оператора 0 (что видно из ярусной формы графа алгоритма на рис. 21, а). Переходы $t_0 — t_3$ биграфа алгоритма (рис. 21, б) соответствуют вершинам графа на рис. 21, а. Для отображения точки входа в алгоритм на его биграфе введена маркированная позиция p_1 , поэтому нулевой вершине графа поставлены в соответствие переходы t_0 и t_4 биграфа. При построении сети рис. 12, в учтено, что время выполнения операций t_1 и t_4 мало по сравнению с остальными. Учитывая также, что действия t_0 и t_2 выполняются на процессоре, позиции p_1 и p_5 , а также p_2 и p_6 биграфа совмещены на рис. 12, в и обозначены как p_1 и p_2 . Процессы выполнения алгоритмов представимы движением меток. Раскраска метки двухэлементная, определяющая номер процесса $a = \{1, 2, 3\}$ и номер этапа обработки $C = \{c_0, c_1, c_2, c_3\}$.

Из рис. 21, б видно, что некоторые из дуг раскрашены одним цветом (c_i), другие — парой цветов (ac_i), прочие не раскрашены вовсе, что учитывается при управлении возбуждением переходов и изменении цвета меток. Заметим, что при раскрашивании парой цветов все эти признаки анализируются или изменяются в комплексе. Например, входные дуги перехода t_4 помечены парами ac_2 и ac_3 , что определяет выбор для синхронизации на t_4 процессов с одним и тем же номером a и завершенными этапами обработки c_2 и c_3 соответственно.

На рис. 22 показаны изобразительные возможности раскрашенных сетей по разделению путей данных и управления в УВВ. Данные от двух источников *Ист* помечены квадратами и p_1 и p_3 , а сигналы запроса на обработку — кружками в p_2 и p_4 . Запросы принимаются (переходы t_1, t_2), накапливаются в регистре (позиция p_5), выбираются из нее (переход t_3) и запускают пересылку данных от источников (p_1 при

p_3) в буфер p_7 . Метка-треугольник в позиции p_8 отражает управляющий процесс, разрешающий выполнение только одной из двух операции считывания t_4 или t_5 .

Таким образом, процедура раскраски сети Петри — это модельный способ трактовки различных по характеру реальных элементов: в алгоритмах — номера оператора или этапа обработки; в мультипрограммных системах — номера задачи, ее приоритета и других атрибутов; для ресурсов — их типа, способа использования и т. д.

Приоритетные раскрашенные сети. В таких сетях приоритетность необходимо определять и по отношению к меткам. Для этого задают отношения приоритетности на множестве цветов и сопоставляют их определенным дугам ($p_i t_j$). Правила возбуждения перехода t_j реализуют приоритетный выбор метки для возбуждения перехода на основе значения соответствующего атрибута-приоритета (цвета) метки. Приоритетные раскрашенные сети адекватно представляют приоритетное обслуживание процессов.

Например, спецификация правил приоритетного обслуживания для модели (см. рис. 21) включает определение отношения порядка (приоритетности) на множестве номеров задач $a = \{1, 2, 3\}$. Сопоставление этого правила дугам ($p_1 t_0$) и ($p_3 t_3$) указывает на приоритетность выбора процесса для выполнения процессором и объектом управления. Аналогично задается отношение порядка на множестве признаков $\{\bullet, \circ\}$, представленных на рис. 22 графически.

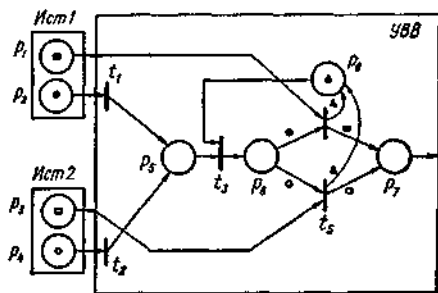


Рис. 22. Схема разделения путей данных и управления раскрашиванием сети

Правило выбора сопоставляется дуге ($p_5 t_3$) этой модели. В приведенных примерах внимание акцентировано на моделирование различных элементов ОУВС. Полностью аналогичны им особенности

моделей дискретных производственных систем, где необходимо представлять потоки материалов, инструмента, управляющих данных и их взаимодействия с ресурсами управляемого объекта (см. рис. 5, 6).

9.2.2. Нагруженная сеть как графический язык высокого уровня

Сложность дискретных облачных управляющих систем и производств требует применения разнообразных методов моделирования для эффективного проектирования структур и планирования операций в системах. Необходимы различные уровни детализации и модельное представление различных аспектов процессов. Отражать все их в единой модели на основе общецелевой системы моделирования нецелесообразно, поскольку это ведет к значительным затратам времени расчетов. Важным требованием к моделям САПР является гибкость и непосредственная интерпретация входных данных, объем которых может быть значительным. Ранее были рассмотрены различные типы сетевых моделей, представляющих те или иные стороны объектов. Реальные системы могут содержать такие особенности, которые несовместимы в частных моделях. В связи с этим для описания различных и многообразных свойств систем на основе единого базового языка необходимо выбрать такие средства, которые легко адаптируются для представления различных частных классов сетевых моделей, и при необходимости объединить эти частные свойства в едином описании.

По аналогии с языком нагруженных графов, используемым в Р-технологии проектирования последовательных программ, язык для представления моделей распределенных структур процессов, ресурсов, алгоритмов и программ построен на основе нагруженных сетей.

Основные элементы языка нагруженных сетей (ЯНС) — графическая часть и ее спецификация. В качестве базовых компонентов графической части использовано минимальное множество элементов, которые обеспечивают построение частных типов моделей (позиция, переход, ориентированная дуга (ребро), метка). Для представления динамического поведения систем этим элементам сопоставляются нагружающие атрибуты и отношения.

Объединение биграфа и меток с атрибутами и отношениями спецификации порождает нагруженную сеть, предназначенную для адекватного представления взаимодействующих процессов, ресурсов и структур данных. Главные особенности нагруженных сетей: правила управления сетью учитывают нагружающие отображения; биграфом представляются как потоки данных, так и потоки управления; в

единообразной форме представимы статические данные и ресурсы; различные интерпретации нагруженной сети позволяют выделить сети для потоков управления и потоков процессов на ресурсной среде, обеспечивая применимость в проектировании имитационного моделирования и аналитического аппарата, разработанного для изучения свойств сетей. Дадим формальное описание и содержательную трактовку ЯНС.

Пусть $N = (T, P, I, O)$ — ориентированный биграф, в котором $T = \{t_j\}$ — конечное множество вершин, называемых переходами; $P = \{p_i\}$ — конечное множество вершин, называемых позициями; $I : T \times P \rightarrow \{0,1\}$ — функция следования; $O : P \times T \rightarrow \{0,1\}$ — функция предшествования; $a_{ij} \in A_1$ и $a_{ji} \in A_2$ — дуги биграфа; $A = A_1 \cup A_2$, причем $a_{ij} = (p_i t_j)$ и $a_{ji} = (t_j p_i)$ — дуги, следующие соответственно от позиции к переходу и наоборот. Примем μ — множество меток; $\mu_s \in \mu$ — s -я метка ими. Каждой метке μ_s сопоставляется описатель $D_s = (d_1, d_2, \dots, d_N)_s$, который позволяет различать атрибуты d_k , связанные с ней, $k = 1, 2, \dots, N$. Атрибут d_k может предназначаться для описания типа и свойств объектов системы, представляемых метками. Например, $d_1 = \{\text{сигнал, процесс, команда, данные, устройство, процессор}\}$; $d_2 = \{\text{номер, имя объекта}\}$ и т. д. Если метка предназначена для представления данных, т. е. значение $d_1 = \text{данные}$, то d_2 может представлять их тип, а $(d_3, d_4, \dots)$ — текущее значение. Если $d_1 = \text{команда}$, то атрибут d_2 может представлять ее формат, а d_3, d_4 — различные поля, предусмотренные в команде данного формата. Пусть $D = \{D_s\}$ — множество описателей меток. Маркирование сети сопоставляет каждой из позиций множество меток из μ , а каждой из меток — описатель, со сформированными значениями атрибутов. В каждом из состояний сети маркирование задается набором отображений: $M : (P \rightarrow \mu \rightarrow D)$. Начальное маркирование сети, обозначаемое M_0 , задает начальное риопределение меток по позициям и начальные значения атрибутов $d_k \in D_s$. Для адекватного представления свойств систем на множествах значений атрибутов d_k определяются бинарные отношения r_k . Типичными r -отношениями на множествах значений d являются отношения порядка. Они могут задаваться решеткой, матрицей, орграфом. Управление процессами и ресурсами на сети выполняется путем сопоставления дугам биграфа N управляющих параметров. Каждой дуге a_{ij} , направленной от позиции p_i к переходу t_j , сопоставляется значение $c_{ij}^k \in X_k$, определяющее условие возбуждения перехода t_j по входящей в него дуге a_{ij} . Здесь X_k — множество допустимых значений k -го атрибута метки. Если в p_i размещается метка m_s , k -й атрибут которой $d_k(m_s)$ в составе пары $(d_k(m_s), c_{ij}^k)$ удовлетворяет отношению r_k , то m_s может участвовать в

возбуждении перехода t_j на дуге a_{ij} , т. е. если $(d_k(m_s), c_{ij}^k) \in r_k$, то метка m_s , находясь в p_i , является кандидатом на возбуждение t_j . Если не исключено появление нескольких переходов-кандидатов на возбуждение, в которых участвует одна и та же метка, то может потребоваться задание вторичного отношения, обеспечивающего определенность выбора. При описании сети могут накладываться ограничения на правила построения сети, исключающие недетерминизм выбора.

Условие возбуждения перехода t_j представляется функцией

$$U(t_j) = \forall p_i \in PRE(t_j) \exists (m_s \in \mu) [(d_k(m_s), c_{ij}^k) \in r_k],$$

причем если $U(t_j) = 1$, то t_j возбуждается.

Время между моментами возбуждения переходов t_j и их срабатыванием задается отображением $\tau : T \times d_t \rightarrow R_0$, где $d_t \in D_s$ — атрибуты метки, управляющие временными параметрами сети.

В интервале между возбуждением и срабатыванием перехода метки закрепляются за ним, находясь в «недействительном» состоянии по отношению ко всем элементам сети, кроме данного перехода. При срабатывании переходов осуществляется пересылка меток в позиции $POST(t_j)$, следующие за t_j . Дугам a_{ji} , следующим из переходов в позиции, присваиваются функции φ_{ji} , определяющие операции над параметрами описателей D_s тех меток, которые участвуют в возбуждении t_j . Значения этих функций определяют, например, следующие действия: направляемая по дуге a_{ji} метка m_s является потомком метки, пришедшей по дуге a_{ij} , а некоторый ее атрибут d_k модифицируется в соответствии с операциями, заданными отображением $\varphi_{ji} \in \Phi_2$. Если срабатывание перехода t_j связано с выполнением действий над некоторой структурой данных, то операционная функция φ_{ji} задает вид обработки атрибутов меток-данных, участвующих в возбуждении.

Нагруженная сеть — это объект, задаваемый набором

$$НС = \{N, D, F, \tau, M_0\}, \quad (2.1)$$

где N — ориентированный биграф; D — множество описателей меток; F — нагружающее отображение, в котором

$$F_1 : A_1 \rightarrow \Phi_1; \quad F_2 : A_2 \rightarrow \Phi_2; \quad F = F_1 \cup F_2;$$

$\Phi_1 = \{r_i\}$ — множество управляющих функций; $\Phi_2 = \{\varphi_{ji}\}$ — множество операционных функций; $\tau : T \times d_t \rightarrow R_0$ — функция, определяющая время срабатывания переходов; d_t — атрибут метки m_s , управляющий временем; M_0 — начальное маркирование сети.

Для обозначения факта отсутствия операции управления ϕ_1 , или операции обработки ϕ_2 на дугах $a \in A$ используем символ λ . При графическом изображении сетей этот символ будем опускать. Обладая высокой изобразительной способностью, класс нагруженных сетей может порождать более простые их типы. Так, если $D = F = \tau = \emptyset$, то маркированная нагруженная сеть (2.1) лишается нагружающих отображений и переходит в класс сетей Петри. Если $D = F = \emptyset$, получается временная сеть Петри, переходы которой нагружены временем срабатывания. В сетях с очередями управляющая функция ϕ_1 учитывает значение временного атрибута d , указывающего время входа метки в позицию. Подклассами нагруженных сетей являются многоклассовые, многопоточные, цветные и другие разновидности сетей с различаемыми метками, в которых соответствующим образом определены D , F и τ .

Описанная модификация сетей Петри приспособлена для представления потоков данных и управления, позволяет отображать операции над статическими распределенными структурами данных, в единообразной форме представлять операции над ресурсами и процессами. Спецификация вида управляющих функций и их аргументов обеспечивает реализацию не только тех классов сетей, которые перечислялись в 9.2.1, но и других, удобных для конкретного пользователя. Изобразительные возможности нагружений сети как средства описания взаимодействующих процессов и ресурсов объединяют изобразительные мощности частных видов сетей, представимых нагруженной сетью.

Построение НС-модели распределенных программно-аппаратных систем следует выполнять в следующей последовательности. На основе анализа зависимостей по данным и управлению, отраженным в алгоритме распределенной обработки, находят структуру сетевой модели алгоритма, оценивают временные свойства операторов, определяют порядок наделения ресурсов, строят биграф, описывающий взаимодействие процессов и ресурсов, формируют функции, управляющие динамикой сети, и определяют ее начальное маркирование.

На рис. 23 приведен пример представления распределенных данных, хранимых на пассивных ресурсах и операций над ними с помощью активного ресурса.

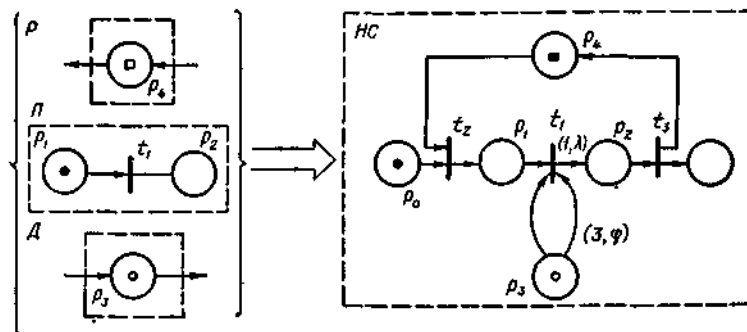


Рис. 23. Схема получения нагруженной сети $НС$ путем связывания процесса $П$, ресурса $Р$ и данных $Д$

Метка, обозначенная кружком в позиции p_3 , вместе со своим описателем D представляет некоторую структуру данных (пассивный ресурс). Поток управления задают меткой, движущейся из p_1 . Ресурсная метки обозначена квадратом в p_4 .

Приведенная на рисунке нагруженная сеть получена в результате связывания процесса (алгоритма обработки t_1), процессора (активного ресурса) и данных. Формат операционных функций определим в виде пары (x, y) , где y — идентификатор функции; x — ссылка на позицию, атрибуты меток которой используются как аргументы в y . Надпись $(3, \varphi)$ возле дуги $(t_1 p_3)$ указывает на обработку меток-данных из позиции p_3 с помощью функционального преобразования φ . А надпись $(1, \lambda)$ возле дуги $(t_1 p_2)$ указывает на отсутствие операции над меткой из p_1 при пересылке ее в p_2 . Поскольку остальные дуги не нагружены соответствующими надписями, операции над атрибутами меток на них не выполняются. Такая же пара (x, y) может связываться и с описателем времени срабатывания t_1 перехода t_1 , определяя его величину в зависимости от значения атрибутов меток-данных из определенных позиций.

9.2.3. Уравнения состояний, динамика сети Петри и анализ качественных характеристик систем

Сети Петри отражают логическую последовательность событий, позволяют проследивать потоки информации, отражают взаимодействие параллельных процессов. Наиболее многообещающая область теории сетей Петри — спецификация и анализ взаимодействующих процессов. Преимущество этого типа моделей

состоит в простоте понимания на инженерном уровне и быстроте чтения графических образов. Внедрение графики в моделирование предоставляет возможности в представлении динамики функционирования сети. Воспроизведение динамики меток на экране дисплея позволяет проследить ход и качественные характеристики процессов, а использование цветного графического дисплея — эффективно интерпретировать динамику раскрашенных сетей. Моделирование в ГАПС — не только инструмент анализа, но и средство отладки дискретных систем управления, диагностики качественных характеристик процессов и источников вредных функциональных эффектов.

Рассмотрим сеть Петри как математическую систему. Статические свойства системы определяет графовая часть сети Петри, а динамические — начальное маркирование и правила возбуждения (моделирования). Для того чтобы обеспечить изложение вопросов динамики некоторых классов сетей Петри (базовых N -сетей, временных и раскрашенных сетей), дадим определения основных характеристик сетей, а затем представим уравнения динамики. Маркирование M называют достижимым из маркирования M_0 , если существует зажигающая последовательность (последовательность срабатывающих переходов) $W = t_1 t_2 \dots t_n$, переводящая сеть N из состояния M_0 в M . Отношение следования маркирований, возбуждаемых действием последовательности

$$W: M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{t_3} \dots \rightarrow M$$

Переход t_j называют достижимым в N из маркирования M , если существует такое маркирование M' , достижимое из M , при котором произойдет срабатывание перехода t_j .

Переход t_j называют живым, если он достижим в N из любого $M \in R(N)$, где $R(N)$ — множество достижимых маркирований сети.

Сеть N — живая, если все ее переходы живые. Доказав, что сеть является живой, гарантируем, что в соответствующей ей системе выполним все элементарные действия процессов при их развитии. Позицию p_i называют n -ограниченной, если существует неотрицательное целое n , такое что $\forall M \in R(N) : m(p_i) \leq n$, где $m(p_i)$ — число меток в позиции p_i .

Сеть N — ограниченная, если все ее позиции ограничены, т. е. для каждой позиции сети существует предельное число меток, которые могут быть там одновременно. Если метки используются для представления промежуточных данных, созданных системой, то, проанализировав ограниченность сети, можно сделать вывод о возможности потери информации в ВС при ее буферировании.

Сеть N — безопасная, если $\forall M \in R(N) : m(p_i) \leq 1$, т. е. при всех достижимых маркированиях ее позиции не могут иметь более одной метки. Анализ безопасности проводят для исключения условий одновременного использования ресурса несколькими процессами. Например, если не могут два сообщения одновременно передаваться по каналу или два вычислительных процесса не могут одновременно находиться в критической программной секции, имеет место безопасность маркирования в соответствующих позициях сетевой модели.

Маркировку $M \in R(N)$ называют тупиковой, если в этом состоянии сети ни один из переходов не может сработать.

Сеть N обладает терминальностью (завершаемостью) процессов, если при всех допустимых начальных маркированиях завершение процессов в ней приводит к тому, что метки покидают сеть, или собираются в некоторой терминальной (конечной) позиции. Терминальность гарантирует работу систем при следующей инициализации без всяких побочных эффектов. Терминальность процессов связана с результативностью алгоритмов — их важнейшим свойством. Два перехода, связанные с общей входной позицией p_i , являются конфликтными, если они возбуждаются в результате появления метки в p_i . Конфликт за метку из p_i приводит к случайному (не-детерминированному) характеру процессов в сети.

Сеть — непротиворечивая, а процессы в ней — детерминированные, если для всех достижимых маркирований множество переходов $E(M)$, возбуждаемых маркированием M , равно множеству $S(M)$ переходов, срабатывающих при M , т. е. $\forall M \in R(N) : E(M) = S(M)$.

Две сети N_1 и N_2 эквивалентны, если $L(N_1) = L(N_2)$, т. е. языки, порожаемые ими, совпадают.

Последовательность позиций и переходов $p_1 t_1 p_2 t_2 \dots p_n$ называют маршрутом от p_1 к p_n , если переход t_i является входным к позиции p_{i+1} и выходным к p_i для $1 \leq i \leq n - 1$.

Замкнутый маршрут, т. е. маршрут, где $p_1 = p_n$, называют циклом.

Простым циклом называют такой замкнутый маршрут, в котором все вершины различны (кроме первой и последней).

Сеть называют сильно связной, если каждая пара ее позиций взаимно достижима.

Опишем сформулированные свойства для сети N (рис. 24, а).

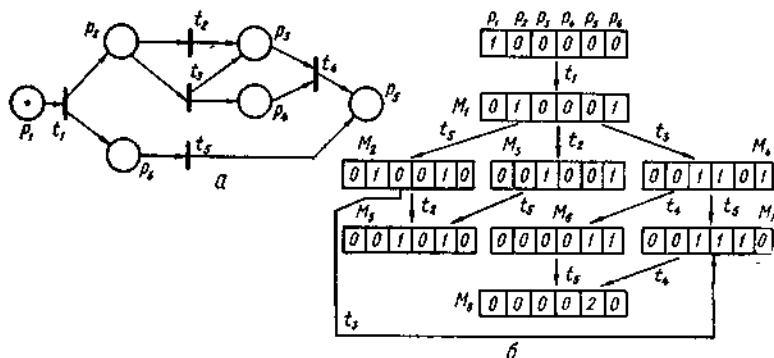


Рис. 24. Разомкнутая сеть (а) и граф достижимых маркировок $R(M_0)$ этой сети (б): у дуг указаны срабатывающие переходы

Ее начальное маркирование равно $M^*_0 = |100\ 000|$; * — операция транспонирования вектора-столбца. Язык, порождаемый сетью с данным M_0 , представлен следующим множеством последовательностей срабатывающих переходов

$$L = \{t_1 t_5 t_2, t_1 t_2 t_5, t_1 t_3 t_4 t_5, t_1 t_3 t_5 t_4, t_1 t_5 t_3 t_4\}.$$

Множество достижимых маркировок

$$R = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right\}.$$

Сеть при данном M_0 — живая, ограниченная, но небезопасная, поскольку пятая позиция восьмой маркировки имеет дм« метки: $m_8(p_5) = 2$. Если начальное маркирование взять равным M_5 , то в ней ни один из переходов не может сработать. Поэтому сеть с этой начальной маркировкой не обладает живостью.

Анализируя маркирования сети, можно увидеть, что если $m(p_2) = 1$, то происходит возбуждение переходов t_2 и t_3 . Срабатывание этих переходов предполагает изъятие каждым из них по одной метке из p_2 . Но в p_2 имеется всего лишь одна метка, поэтому только один из возбужденных переходов может сработать. Следовательно, данная сеть не является детерминированной. Как видно из рис. 24, б, сеть имеет тупиковое маркирование — состояние M_8 .

При описании процессов сетями Петри всякое маркирование сети ассоциируется с определенным состоянием моделируемой системы. Срабатывание любого перехода t_i сети N связано с изменением ее состояния (выполнением элементарного действия в системе). Свойство ограниченности сети соответствует тому факту, что ограничено число меток во всех позициях, отсюда следует, что число ее состояний также ограничено.

Проверка свойств сетей может выполняться путем построения и анализа множества достижимых состояний системы. Однако в случае большого количества состояний такой способ затруднителен. Другой способ представляет собой структурный анализ сети на основе заданной матрицы инцидентности и начального маркирования. Этот способ рассмотрим ниже.

Динамику сети в пространстве состояний можно описать следующим рекуррентным уравнением:

$$M_k = M_{k-1} + A^*U_k; \quad k = 1, 2, \dots, \quad (2.2)$$

где M_k — состояние, которое следует после состояния M_{k-1} в результате k -го воздействия U_k ; M_0 — начальное маркирование; A^* — матрица, полученная транспонированием матрицы инцидентности позиций и переходов, причем ее элементы a_{ji} равны n , $-n$ или 0 , когда, соответственно, переход j имеет n исходящих дуг к позиции i , n входящих дуг в позицию i или не имеет связи с позицией i ; U_k — управляющий вектор, компоненты которого $U_{jk} = \{0, 1\}$. Если $U_{jk} = 1$, то в k -й момент асинхронного времени происходит срабатывание перехода t_j , если же $U_{jk} = 0$, то срабатывание t_j не происходит. Элементы a_{ji} матрицы A определяют воздействия, оказываемые процессом на состояние системы, указывая число меток, добавляемое в позиции p_i при срабатывании переходов t_i . Матрицу A можно вычислить на основе операций над матрицами I и O , задающими соответственно число дуг, следующих за переходами, и позициями $A = I - O^*$. Элементы матрицы O задают число меток, которые должны быть изъяты из позиций p_i при срабатывании перехода t_i . Элементы строки t_j матрицы I определяют число меток, направляемых в позиции сети при срабатывании перехода t_j .

Для вычисления управляющего вектора U_k необходимо определить те переходы t_j , которые срабатывают при маркировании M_{k-1} . Если t_j срабатывает, то $u_j = 1$ и произведение A^*U_k дает число меток, добавляемых в позицию p_i в k -м такте. Из анализа матрицы O можно установить те переходы t_j , которые возбуждаются при заданном маркировании M . Условие срабатывания перехода t_j и выработки $u_j = 1$ представляется логическим уравнением

$$\forall_i [m(p_i) \geq I(t_i p_i)] \Rightarrow (u_i = 1), \quad (2.3)$$

которое управляет действиями t_j .

Для генерирования диаграммы состояний сети Петри достаточно использовать рекуррентное уравнение состояний (2.2) и логическое уравнение, формирующее управляющий вектор U_k .

В теории сетей Петри иногда исходят из того, что не указывают порядок срабатывания переходов, если вектор управления U задает возбуждение нескольких из них в одном и том же такте k . Так, в сети N (рис. 25) в состоянии M_0 вектор управления $U^*_1 = |101|$ что указывает на возбуждение как первого, так и третьего переходов.

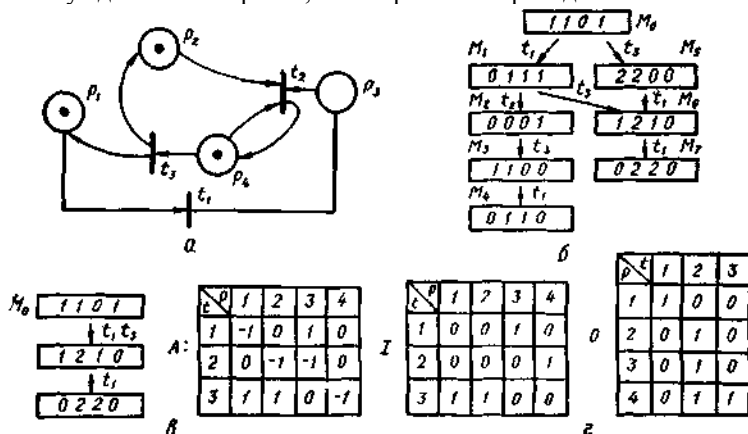


Рис. 25. Замкнутая сеть (а), диаграммы смены ее состояний (б, в) и матричное представление структуры сети (г)

Следуя сформулированной посылке, можно предположить срабатывание или t_1 или t_3 , тогда будет переход в состояние M_1 или в M_5 соответственно. Однако, если следовать уравнению (2.2), то диаграмма состояний сети имеет вид, приведенный на рис. 25, в. Итак, если полагать, что все возбужденные переходы срабатывают вместе, то динамику сети можно представить уравнением (2.2). Если исходить из того, что возбужденные переходы могут срабатывать только порознь, то уравнение состояний сети будет иметь другой вид.

Соответствующие этим управляющим посылкам диаграммы маркирований приведены на рис. 25, в и рис. 25, б. Совмещение первого и второго способов срабатывания одновременно возбуждаемых переходов даст третью диаграмму состояний, являющуюся наложением двух приведенных на рисунке.

Одну из трех возможных альтернатив в способе переключения переходов выбирают в соответствии с постановкой задачи практики. В определении сети Петри не содержится время, поэтому и возникает такое многообразие вариантов управления сменой состояний. В реальных распределенных объектах имеется вполне определенный способ управления сменой состояний и присутствует такой фактор как время, что позволяет уточнить уравнение (2.3).

Важным в практических приложениях является класс сетей Петри, в которых переходам t_j сопоставлены интервалы времени τ между моментами возбуждения и срабатывания переходов. Вектор времен τ описывает длительности элементарных активностей, входящих в процессы в определенных отношениях следования. Если времена срабатывания переходов одинаковы, то динамика системы моделируется уравнениями (2.2), (2.3), поскольку такая сеть является синхронной. Этих уравнений недостаточно для моделирования асинхронных процессов, для которых характерно различие элементов τ . И этом случае процедуру продвижения меток в сети следует производить в асинхронном времени, введя в модель часы. Если в синхронных системах достаточно вести отсчет тактов, то в асинхронных необходимо в каждом из тактов определять момент наступления очередного. Интервал времени между тактами k и $(k + 1)$ определяется временем наступления ближайшего события в сети. Здесь под событием понимается одновременное срабатывание одного или нескольких переходов. Уравнение (2.2) отражает свободное движение сети из начального состояния M_0 . Вынужденное движение сети определяется также вектором-функцией $W(t)$, компоненты которой $W_i(t)$ описывают приход меток в позиции p_i во времени; $W(t)$ может задаваться в моменты дискретного времени k , наступление которых устанавливается моментами переключения состояния сети или внутренними часами. Уравнение вынужденного движения сети Петри имеет вид

$$M_k = M_{k-1} + A * U_k + W_k. \quad (2.4)$$

Таким образом, введенная здесь последовательность W_k описывает внешний источник меток, представленных входной лентой.

Перемещение ленты для считывания следующей группы меток и ввода их в соответствующие позиции выполняется в асинхронном времени по командам от часов или по сигналам перемещения ленты из самой системы.

Последовательность W_k предназначена для отображения взаимодействий в структурированных сетях, она может отображать последовательные наборы данных, управляющих команд и т. д.

Очевидно, что в любом состоянии системы компоненты вектора маркировки не могут быть отрицательными. Они могут принимать только нулевые или положительные целочисленные значения. Это условие в краткой записи имеет вид $M_k \geq 0, \forall k$. Учитывая последнее, из уравнения состояний (2.2) получаем $M_{k-1} + A^*U_k \geq 0$. Последовательность маркирований (2.2) можно выразить через начальную маркировку M_0 и вектор счета срабатываний

$$S = \sum_k U_k; \quad S = |s_j|; \quad j = \overline{1, m}. \quad (2.5)$$

Элемент s_j указывает, какое число раз срабатывает переход t в последовательности маркирований, ведущей от M_0 к M_k .

Учитывая уравнения (2.5), из выражения (2.2) получаем

$$M_k = M_0 + A^*S \quad (2.6)$$

Введя вектор изменения маркирования $\Delta M = M_k - M_0$, из уравнения (2.6) получим уравнение

$$A^*S = \Delta M. \quad (2.7)$$

К нахождению решений данного уравнения сводятся различные задачи практики. Все его возможные решения можно получить одним из методов решения задач целочисленного программирования.

Рассмотрим использование методов линейной алгебры в решении рекуррентных уравнений смены состояний сетевой модели. Они позволяют на основе математического исследования структуры биграфа сети и начального маркирования M_0 оценить такие качественные характеристики сети как ограниченность, живость и т. д. Целочисленный вектор $X = |x_i|, i = \overline{1, n}$, являющийся решением линейной системы

$$AX = 0, \quad (2.8)$$

называется p -инвариантом или p -циклом

Рассмотрим уравнение (2.2), обе части которого умножим на X^* :

$$X^*M = X^*M_0 + X^*A^*S. \quad (2.9)$$

Известно, что $AX = X^*A^*$, поэтому из выражения (2.9) с учетом системы (2.8) получим

$$X^*M = X^*M_0. \quad (2.10)$$

из которого следует, что любой p -инвариант характеризует все достижимые маркирования сети с позиции сохранения некоторых свойств процессов. Если обозначить $X^*M_0 = K_0$, где K_0 — постоянная величина, то свойство инвариантности маркирований сети представимо в виде отношения-равенства

$$X^*M = K_0 = \text{const}. \quad (2.11)$$

Вектор X называют p -инвариантом, поскольку он определяет некоторое из свойств распределения меток по позициям p_i .

Фундаментальной системой решений системы линейных однородных уравнений (2.8) называют такую их совокупность, через которую линейно выражаются все остальные решения. Если ранг матрицы A равен числу неизвестных ($r=n$), то система (2.8) имеет только нулевое решение. Если $r \leq n$, то система (2.8) помимо нулевого имеет бесконечное множество других решений, причем фундаментальная система состоит из $(n - r)$ векторов X . Ранг $n \times m$ -матрицы $A = \|a_{ij}\|$ равен наивысшему порядку отличного от нуля определителя, полученного вычеркиванием $n - r$ столбцов и $m - r$ строк из матрицы A .

Таким образом, все инварианты X для маркирований сети можно получить из $n - r$ базисных решений. Объединив записанные в виде векторов-строк решения фундаментальной системы, получим матрицу инвариантов или базисных решений B . Тогда для любого достижимого маркирования аналогично равенству (2.11)

$$BM = BM_0 = K_0. \quad (2.12)$$

Если все компоненты p -цикла неотрицательны, его называют p -цепью. Полная p -цепь — это p -цепь, все компоненты которой положительны.

Сеть Петри инвариантна, если для нее существует полная p -цепь

Полная p -цепь включает в себя все позиции сети

Инвариантная сеть Петри является ограниченной. Докажем это. Пусть X — полная p -цепь. Тогда $X^*M = K_0$, т. е. взвешенная сумма марок по

$$\sum_i x_i m_i = K_0$$

всем позициям — ограниченная. А поскольку x_i положительны и вся сумма ограничена, то и маркирования всех позиций сети ограничены.

Рассмотрим следующую группу характеристик, получение которых основывается на вычислении t -инвариантов (или t -циклов)

Целочисленный вектор Y называется t -инвариантом, если он является решением линейной системы

$$A^*Y = 0 \quad (2.13)$$

Если значение t -инварианта подставить в уравнение (2.6) вместо вектора счета срабатываний S , то окажется, что $M = M_0 + A^*Y = M_0$.

Отсюда следует, что если $Y \neq 0$, то сеть устойчива, т. е. после ряда переключений она возвращается в исходное состояние M_0 .

Устойчивость процессов связана с их циклической повторяемостью, начиная с состояния M_0 . Заметим, что среди решений системы (2.13) могут быть такие векторы Y , компоненты y_j которых отрицательны

Полная t -цепь — это t -цикл, все компоненты которого положительны. Полная t -цепь включает в себя все переходы сети. Если сеть имеет полную t -цепь, то она жива при любом начальном маркировании, поскольку в цепи маркирований от M_0 до $M = M_0$ срабатывают все переходы

Таким образом, по устойчивости процессов и значению t -инварианта можно определить живость сети, если сеть жива и ограничена, то она инвариантна и устойчива

Решим вопрос обнаружения тупиковых состояний

Используя уравнение (2. 3), условие возбуждения перехода t_j можно представить в виде

$$\sum_{i=1}^n l(i, p_i) m(p_i) \geq \sum_{i=1}^n l(t_j, p_i),$$

или учитывая, что $l(i, p_i) = O^*(p_i, t_i)$,

$$\sum_{i=1}^n O^*(p_i, t_j) m(p_i) \geq \sum_{i=1}^n O^*(p_i, t_j). \quad (2.14)$$

Для краткости будем использовать векторную запись условия возбуждения переходов $O^*M \geq O^*E$, где E — единичный вектор

Условие (2. 14) трактуется следующим образом: если некоторая компонента j левой части при данном маркировании M равна соответствующей компоненте правой части или больше ее, то переход t_j возбуждается.

Поскольку множество достижимых маркирований $R(N)$ должно удовлетворять условию (2. 11), то отсутствие возбуждаемых переходов для $M \in R(N)$ следует определять из решения системы

$$\begin{cases} BM = BM_0; \\ O^*M = O^*E - E. \end{cases} \quad (2.15)$$

Если эта система имеет решения, то некоторое ее маркирование является тупиком. Заметим, что отношению (2. 12) могут удовлетворять и недопустимые маркирования Поэтому решением системы (2. 15) также могут быть недопустимые маркирования, хотя все допустимые тупики удовлетворяют ему. Таким образом, система (2.15)—достаточное условие того что маркирование M тупиково.

Для анализа тупиков можно также использовать систему

$$\begin{cases} M = M_0 + A^*S \geq 0; \\ O^*M \leq O^*E - E, \end{cases}$$

или

$$\begin{cases} M_0 + A^*S \geq 0 \\ O^*AS \leq O^*(E - M_0) - E, \end{cases} \quad (2. 16.)$$

если в число исходных данных входит известный вектор счета срабатываний S . Таким образом, несовместность системы (2.15) или (2.16) свидетельствует об отсутствии маркирования M или вектора счета срабатываний S , которые ведут к тупику из M_0 .

Рассмотрим на простых примерах возможности указанного математического подхода. Для анализа более сложных сетей необходимо применять ЭВМ, разработав для этого соответствующее программное обеспечение.

Например, для сети N (см. рис. 25) согласно уравнению (2.8) p -инвариант X является целочисленным решением системы

$$-x_1 + x_3 = x_2 + x_3 = x_1 + x_2 - x_4 = 0.$$

Ранг матрицы A системы $AX = 0$ равен трем, поэтому данная система имеет одно целочисленное базисное решение, вектор-строка которого $X^* = | 1 \ -1 \ 1 \ 0 |$. Подставив X и M_0 в равенство (2.12), получим условия верные для любого достижимого маркирования

$M = | m_1 \ m_2 \ m_3 \ m_4 | : m_1 - m_2 + m_3 = 0$ или $m_1 + m_3 = m_2$. Из последнего условия виден смысл инварианта: сумма числа меток в позициях p_1 и p_3 и число меток в позиции p_2 равны для любого достижимого маркирования сети. Этот инвариант верен здесь для диаграмм состояний рис 25, б и 25, в, соответствующих разным правилам управления срабатыванием переходов. Вывод об ограниченности сети по данному значению X сделать нельзя

Вычислим t -инварианты Y . Из системы (2.13) получим систему уравнений $-y_1 + y_3 = -y_2 + y_3 = y_1 - y_2 = -y_3 = 0$. Система имеет только нулевое решение $Y^* = | 0 \ 0 \ 0 |$, поэтому в данной сети не существует маркирований, связанных циклами на диаграмме состояний.

Маркирование M совпадает с начальным, только если не сработает ни один из переходов сети, поскольку все компоненты вектора Y нулевые. Иначе говоря, поскольку $Y = 0$, то последовательность состояний сети не имеет возвратов, что подтверждается диаграммами состояний на рис. 25.

В связи с полученными особенностями целесообразно определить, имеет ли тупик указанная невозвратная последовательность маркировок. Система уравнений (2.15) имеет вид

$$m_1 - m_2 + m_3 = m_1 = m_4 = 0; \quad m_2 + m_3 + m_4 = 2.$$

Система совместна, и ей удовлетворяют любые маркирования, которые имеют $m_1 = 0; m_2 = m_3; m_4 = 0$. Тупиковыми, например, являются маркирования $M = | 0 \ 1 \ 1 \ 0 |$; $M = | 0 \ 2 \ 2 \ 0 |$ и т. д. Таким образом, сеть N_2 не является беступиковой при $M_0 = | 1 \ 1 \ 0 \ 1 |$. Рассмотрим еще один пример анализа.

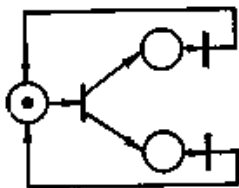


Ранг матрицы A равен трем, поэтому $n - t = 1$, т. е. в фундаментальную систему решений однородной системы $AX = 0$ входит один вектор X . Распишем систему $AX = 0$: $x_1 - x_3 = x_2 - x_1 = x_3 - x_2 = 0$. Взяв ее целочисленное решение $X^* = |1\ 1\ 1|$, получим условие инвариантности для позиций: $X^*M = X^*M_0 = m_1 + m_2 + m_3 = 1$. Интерпретация полученного результата состоит в следующем: сеть ограничена, поскольку $\sum_i m_i = 1$, и, более того,—безопасна, поскольку суммарное число меток

в сети не превышает 1. Для вычисления t -инварианта рассмотрим систему $y_1 - y_2 = y_2 - y_3 = y_3 - y_1 = 0$, из которой найдем $Y^* = |1\ 1\ 1|$. Поскольку $Y \neq 0$ — сеть устойчива, т. е. процессы в ней периодически повторяются. Поскольку t -инвариант Y охватывает все переходы, так как все $y_i = 1$, то сеть жива. Система уравнений (2.15) для данной сети

$$\begin{cases} m_1 + m_2 + m_3 = 1; \\ m_1 \leq 0; \quad m_2 \leq 0; \quad m_3 \leq 0. \end{cases}$$

Поскольку она несовместна, делаем вывод об отсутствии тупиков в данной гати. Применив описанный подход к анализу следующей сети:



можно убедиться в том, что она живая, неограниченная и беступиковая.

Для исследования количественных характеристик сетей можно использовать различные методы теории графов и классического сетевого анализа. В качестве примера рассмотрим решение задачи нахождения всех простых путей между начальным и конечным состояниями и связанную с ней задачу определения вектора счета срабатывания S для переходов сети. Все простые пути из M_0 в M можно найти, используя уравнение (2.7). Если предположить, что в сети в каждый момент дискретного времени срабатывает только один переход, то алгоритм нахождения простых путей можно построить следующим образом: выполняют перебор возможных значений вектора S , и те из них, которые удовлетворяют уравнению (2.7), выбирают в качестве

источников определения простых путей. Ненулевые компоненты вектора S дают число срабатываний и номера переходов, создающих путь из M_0 в M .

Если с каждым переходом связано время τ_i , то для конкретного пути от M_0 к M можно найти время его реализации.

$$T_p = \sum_{i=1}^m \tau_i s_i,$$

В случае параллелизма в срабатывании переходов оценка времени перехода от M_0 к M усложняется.

Оценка времени на пути из M_0 в M позволяет изучить свойства системы путем сравнения их с заданными, обеспечение которых необходимо, например, из условий режима реального времени. Общее решение задачи об эквивалентности сетей Петри и временных сетей неизвестно. Отношение эквивалентности выражает в некотором смысле тождество строения. Эквивалентные — «то «одинаково устроенные» сети. Исключение из рассмотрения времени и переход к анализу невременных сетей имеют двоякие следствия. С одной стороны, исключение времени как параметра существенно усложняет диаграмму состояний и увеличивает множество допустимых маркировок. Диаграмма состояний временной сети вкладывается в диаграмму для невременной сети. В связи с этим при исследовании сетей на основе изучения множества их состояний задача анализа может усложняться. Но с другой стороны, если невременная сеть беступиковая, живая и т. д., то гарантировано сохранение этих свойств и во временной сети Петри. Таким образом, идя на усложнение постановки задачи путем исключения времени, можно все же обеспечить использование методов линейной алгебры и уравнений состояния для изучения качественных свойств систем.

Перейдем к изучению свойств раскрашенных сетей.

Раскрашенная сеть, как частный вариант нагруженных сетей, это объект, который характеризуется следующим множеством элементов:

$$CN = \{N, C, F, CM_0\},$$

где $N = \{T, P, I, O\}$ — биграф; $C = \{X, R\}$; $X = \{\lambda, c_1, c_2, \dots, c_l\} = \{c_r\}$ — множество раскрашивающих цветов или других признаков включающее элемент X , обозначающий «отсутствие цвета»; R — бинарна отношение на множестве цветов X , (чаще всего предполагаемое как отношение эквивалентности или равенства цветов); $F : A \rightarrow X$ — отображение множества дуг биграфа N на множество цветов, дающее раскраску дуг c_{ij}^s , — цвет s -й дуги a_{ij}^s , биграфа, направленной из вершины i в вершину j , CM_0 — начальное цветное маркирование сети. Здесь раскрашиваются также метки сети.

Если μ — множество цветных меток, то цветное маркирование сети характеризуется парой отображений

$$CM : \begin{cases} P \rightarrow \mu; \\ \mu \rightarrow X, \end{cases}$$

первое из которых указывает на размещение меток по позициям, а второе на цвет этих меток. Факт размещения метки m в позиции p_i обозначим $m \in p_i$.

Условимся, что метка m , имеющая цвет $c(m)$, может принять участи в возбуждении перехода t_j , если пара $(c(m), c_{ij}^s)$ удовлетворяет заданному R -отношению на множестве цветов. Если R —отношение равенства, то метка $m \in p_i$ может участвовать в возбуждении t_j при условии что $c(m) = c_{ij}^s$ (т. е. когда цвет метки равен цвету дуги a_{ij}^s , направленной из позиции p_i к переходу t_j). Для возбуждения перехода t_j необходимо, чтобы по всем входящим дугам могли пройти метки соответствующего цвета из позиций p_i , являющихся предшественниками перехода t_j . Логическое уравнение, описывающее условие возбуждения перехода t_j , имеет следующий вид:

$$\forall p_i \in PRE(t_j) \exists m \in p_i [R(c(m), c_{ij}^s) = 1] \Rightarrow u(t_j) = 1,$$

иначе $u(t_j) = 0$. Итак, если $u(t_j) = 1$, то переход возбужден, и осуществляется его срабатывание. Возбуждавшие метки из позиций $PRE(t_j)$ при этом изымаются, а позиции $POST(t_j)$ пополняются метками, цвет которых полагается равным цвету дуг, проходящих из t_j в позиции $POST(t_j)$.

Для отображения цветного маркирования числами условимся представлять его в следующем виде: $CM = |m_1...m_i...m_n|*$, где $*$ — операция транспонирования; $m_i = |m_i^1 m_i^2...m_i^l|$, причем m_i^r — число меток r -го цвета в позиции p_i .

Динамика меток в раскрашенной сети характеризуется матрицей инцидентности D , элементы которой d_{ij} представим в виде векторов-строк длиной l :

$$d_{ij} = |d_{ij}^1 d_{ij}^2 \dots d_{ij}^l|, \quad (2.17)$$

где l — число раскрашивающих цветов.

Значения d_{ij}^r задаются следующим образом. Если метка цвета r потребляется переходом t_j из позиции p_i при срабатывании t_j , то $d_{ij}^r = -1$; Если метка цвета r поступает в p_i из t_j , то $d_{ij}^r = 1$, и, наконец, в отсутствие этих действий $d_{ij}^r = 0$.

Эти же условия можно сформулировать следующим образом. Если позиция p_i имеет исходящие дуги цвета r в переход t_j , то $d_{ij}^r = -1$; если имеет входящие дуги цвета r , то $d_{ij}^r = 1$, и если не имеет инцидентных дуг цвета r , то $d_{ij}^r = 0$.

Динамику продвижения меток по раскрашенной сети можно описать шением состояния:

$$CM_k = CM_{k-1} + D^*U_k, \quad (2-18)$$

CM_k — состояние, в которое переходит раскрашенная сеть из CM_{k-1} под действием управляющего вектора $U_k = |u_{jk}|$, компоненты которого

$$u_{jk} = \begin{cases} 1, & \text{если в } k\text{-й момент срабатывает переход } t_j; \\ 0, & \text{иначе.} \end{cases}$$

Введем в рассмотрение инварианты для раскрашенной сети, подобно тому как это сделано для сети Петри; p -инвариант найдем из решения линейной системы

$$DV = 0, \quad (2.19)$$

где $V = |V_i|$ — целочисленный вектор-столбец, компоненты которого имеют вид

$$V_i^* = |v_i^1 v_i^2 \dots v_i^l|.$$

Учитывая уравнение (2.2), находим

$$V^*CM = V^*CM_0. \quad (2.20)$$

Итак, подобно уравнению (2.10) для сети Петри, уравнение (2.20) для раскрашенной сети описывает инвариантность в ее маркировках.

Фундаментальная система решений однородной системы линейных уравнений имеет $n - rl$ решений, где n — число позиций сети; r — ранг матрицы D ; l — мощность множества цветов X . Объединив их в матрицу CB , получим уравнение, аналогичное уравнению (2.12):

$$CB CM = CB CM_0 = K_0.$$

Решение линейной системы

$$D^*W = 0 \quad (2.21)$$

дает t -инварианты W , где $W = |W_j|$ — целочисленный вектор-столбец, в котором $W_j = |W_j^1 W_j^2 \dots W_j^l|$. Условие существования тупиков в раскрашенной сети находится из анализа системы

$$\begin{cases} CB CM = CB CM_0; \\ CO^*CM \leq CO E' - E, \end{cases}$$

$$(2.22)$$

Подобно тому как это выполняется для сети Петри. Здесь CO^* — матрица аналогичная O^* , расширенная в пределах каждого элемента по принципу (2.17):

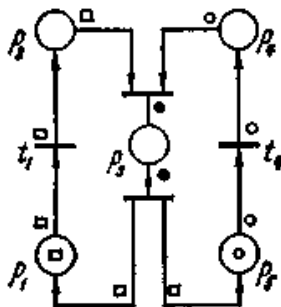
$$CO_{ij}^r = \begin{cases} 1, & \text{если существует дуга из } t_j \text{ в } p_i, \text{ раскрашенная цветом } r; \\ 0, & \text{иначе,} \end{cases}$$

$$CO_{ij}^* = |CO_{ij}^1 CO_{ij}^2 \dots CO_{ij}^l|,$$

E' — единичный клеточный вектор, компоненты которого e'_j представлены наборами из l единиц.

Итак, с математических позиций анализ раскрашенной сети и базовой сети Петри подобен. В случае раскрашенной сети, однако, действия выполняются на уровне клеточных матриц и векторов.

Рассмотрим простой пример для уяснения методики расчета динамических свойств раскрашенных сетей.



В изображенной сети, описывающей взаимодействие двух процессов, обозначенных метками $\{\square, \circ\}$, протокол взаимодействия предусматривает порождение нового процесса $\{\bullet\}$, а затем возврат к исходным. Каждому цвету из множества $X = \{\square, \circ, \bullet\}$ сопоставим числа $r = 1, 2, 3$. Тогда элементы матрицы $D = |d_{ij}|$, где $i = 1, 2, 3, 4$, $j = 1, 2, 3, 4, 5$, равны $d_{11} = d_{22} = |-1 \ 0 \ 0|$;

$$d_{13} = d_{31} = |1 \ 0 \ 0|; \quad d_{35} = d_{44} = |0 \ 1 \ 0|; \quad d_{23} = |0 \ 0 \ 1|; \quad d_{33} = |00 \ -1|;$$

$$d_{24} = d_{45} = |0 \ -1 \ 0|, \text{ а прочие } d_{ij} = |0 \ 0 \ 0|.$$

Вектор начального маркирования $CM_0 = |m_{0i}|$ представлен компонентами: $m_{01} = |1 \ 0 \ 0|$; $m_{02i} = m_{03} = m_{04} = |0 \ 0 \ 0|$; $m_{05} = |0 \ 1 \ 0|$. Уравнение состояния (2.18) здесь имеет вид

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix}_k = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix}_{k-1} + \begin{bmatrix} d_{11} & d_{21} & d_{31} & d_{41} \\ d_{12} & d_{22} & d_{32} & d_{42} \\ d_{13} & d_{23} & d_{33} & d_{43} \\ d_{14} & d_{24} & d_{34} & d_{44} \\ d_{15} & d_{25} & d_{35} & d_{45} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}_k.$$

Его рекуррентное решение позволяет моделировать динамику состояний сети.

Найдем p -инвариант сети $V^* = |V_1 V_2 V_3 V_4 V_5|$, где $V^*_i = |v^1_i v^2_i v^3_i|$. Для чего необходимо решить систему (2.19). Ранг матрицы D равен единице, следовательно, фундаментальная система имеет одно решение, так как $n - rl = 1$. Распишем систему уравнений (2.19) для данной сети:

$$\begin{cases} v_1^1 = v_2^1 \\ v_3^3 = v_2^1 + v_4^2 \\ v_5^3 = v_1^1 + v_5^2. \end{cases}$$

В качестве инварианта можно взять клеточный вектор-столбец с компонентами $V^*_1 = V^*_2 = |1\ 0\ 0|$; $V^*_3 = |0\ 0\ 3|$; $V^*_4 = V^*_5 = |0\ 2\ 0|$, удовлетворяющий приведенной однородной системе линейных уравнений. Затем вычислим произведение

$$V^*CM_0 = |v_1^*v_2^*v_3^*v_4^*v_5^*| |m_1^*m_2^*m_3^*m_4^*m_5^*|_0 = 3$$

и условие инвариантности:

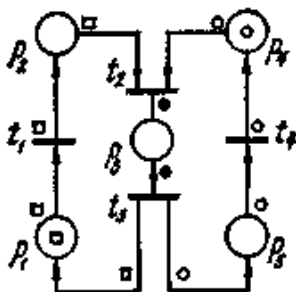
$$VCM = |v_i^*| |m_i^*|_* = \sum_{i=1}^5 V_i^*m_i^* = m_1^1 + m_2^1 + 3m_3^3 + 2m_4^2 + 2m_5^2 = 3.$$

Верхний индекс здесь указывает цвет метки, а нижний — позицию. В рассматриваемом примере возможны следующие варианты распределения меток по позициям и их цветности: $m_1^1 + 2m_4^2 = 3$; $m_1^1 + 2m_5^2 = 3$; $m_1^1 + m_2^2 = 3$; $m_1^1 + 2m_5^2 = 3$; $3m_3^3 = 3$. Из условия инвариантности, содержащего в правой части константу 3, следует ограниченность сети. Решив систему (2.21), можно установить, что данная сеть живая и устойчивая. Элементы матрицы CO равны: $CO_{42} = CO_{54} = |1\ 0\ 0|$; $CO_{42} = CO_{54} = |0\ 1\ 0|$; $CO_{33} = |0\ 0\ 1|$; а прочие $CO_{ij} = |0\ 0\ 0|$. Система (2.22) здесь имеет вид:

$$\begin{cases} m_1^1 + m_2^1 + 3m_3^3 + 2m_4^2 + 2m_5^2 = 3; \\ m_1^1 \leq 0; \quad m_2^1 + m_4^2 \leq 1; \quad m_3^3 \leq 0; \quad m_5^2 \leq 0. \end{cases}$$

Данная система несовместна, а, следовательно, исследуемая раскрашенная сеть не содержит тупиков.

В описанной раскрашенной сети изменим раскраску на дуге (t_3p_5) и примем следующее начальное маркирование:



На основе вычислений можно установить, что приведенная сеть — ограниченная, живая, однако тупиковая, поскольку система

$$\begin{cases} m_1^1 + m_2^1 + 3m_3^3 + 2m_4^2 + 2m_5^1 + 2m_6^2 = 3; \\ m_1^1 \leq 0; \quad m_3^3 \leq 0; \\ m_2^1 + m_2^2 \leq 1; \quad m_6^2 \leq 0 \end{cases}$$

совместна. Тупиковое маркирование описывается вектором

$$CM = [(0 \ 0 \ 0) (1 \ 0 \ 0) (0 \ 0 \ 0) (1 \ 0 \ 0) (0 \ 0 \ 0)].$$

«Обесцвечивание» сети, т. е. переход от раскрашенной сети к базовой сети Петри может привести к потере и искажению информации о протекании процессов в сетевой модели. Так, если выполнить «обесцвечивание» приведенной сети, то расчетами можно установить, что ее инвариант станет равным $m_1 + m_2 + 3m_3 + 2m_4 + 2m_5 = 3$, т. е. произошло как бы «обесцвечивание» инварианта сети, приведенной на с. 72, а не на с. 73. Но самое важное состоит в том, что «обесцвеченная» сеть — беступиковая, хотя исходная раскрашенная сеть имеет тупик. Искажение такого важного свойства при упрощении модели указывает, что для обеспечения достоверности проектного анализа при исследовании раскрашенных сетей недопущим переход к «обесцвеченной» сети.

9.2.4. Операции над сетями и их назначение

Системотехническое проектирование базируется на математической модельной интерпретации управляющих и обрабатывающих структур и процессов. Для обеспечения проектного анализа строят модель объекта проектирования, которую затем используют для оценки эффективности и работоспособности предлагаемого проектантом решения. Получение сетевой модели — это только первый этап проектирования, позволяющий получить формальное представление асинхронных параллельных процессов и структур. Следующие этапы могут предусматривать преобразования одной сети или совместно нескольких. Операции над сетями выполняют на основе приемов декомпозиции сложных систем и построения объединенной структуры из простых. Средства композиции и декомпозиции призваны обеспечить построение корректных простых и эффективных структур процессов и систем.

Разнообразные действия над сетями Петри описаны в литературе, где введены операции наложения, разметки, итераций присоединения и исключения для базовых N-сетей Петри. Практические задачи анализа и синтеза параллельных систем требуют использования таких операций, которые порождают нагруженные сети из простых. При этом, вместе с операциям над их статическими элементами

(позициями, переходами, дугами биграфа), должны быть введены определенные операции над маркировками и нагружающими (в частности раскрывающими) функциями.

Конструкторы сложных дискретных автоматов давно не используют подход к синтезу, который основан на построении частных граф-схем алгоритмов и объединении их в единую граф-схему, являющуюся базой для реализации многофункционального устройства. Таким образом, например, синтезируются управляющие микропрограммные автоматы многофункционального назначения. Аналогичные задачи возникают при синтезе параллельных систем. Решение их требует, чтобы сеть-результат переходила из класса простых сетей в класс нагруженных. Опишем две наиболее необходимые для практики операции над сетями.

1. Операция, предназначенная обеспечивать такое композиционное представление сети из более простых, которое создает условия для выполнения на сети-композиции функций той или иной из исходных сетей по выбору. Такая операция порождает многофункциональную сеть, объединяющую в себе возможности исходных. Эту операцию будем называть *функциональным наложением сетей*. Поскольку действия реальной системы в сетевой модели представляются переходами t_i , то функциональное наложение связано, главным образом, с совмещением этих действий-переходов. Операция функционального наложения обеспечивает централизацию обработки по различным алгоритмам в пределах одного средства, работающего в режиме разделения времени.

2. Операция, связанная с конструированием распределенных систем, использующих часть общих ресурсов по мере их освобождения. Эта операция (*ресурсное наложение сетей*) связывается с разделением ресурсов между несколькими процессами, и может отражать различные способы управления взаимодействием, такие как семафоры или мониторы.

Опишем операцию функционального наложения элементарных участков временных сетей Петри, сводящуюся к наложению переходов и инцидентных им позиций и дуг (рис. 26).

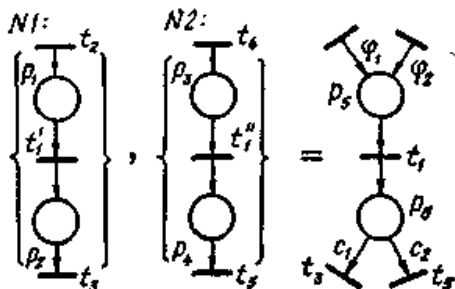


Рис. 26. К объяснению операции наложения элементарных сетей, порождающей нагруженную сеть:
накладываемые участки $N1$ и $N2$ отмечены фигурными скобками

Порождаемый участок сети $HC = (N1, N2)$ должен функционировать как сеть $N1$ или $N2$ в зависимости от того, из какого источника в него попадает метка. В этих условиях при слиянии переходов t'_1 и t''_1 необходимо также выполнять слияние инцидентных им позиций. Операция наложения функций переходов t'_1 и t''_1 на рисунке выполняется в следующей последовательности: с сетями-операндами связываются значения различающего атрибута $x \in X = \{c1, c2\}$ и на мно-жестве X определяется R -отношение равенства; выполняются операции слияния соответствующих статических элементов заданных участков биграфов $p_5 = p_1 \cup p_3$; $p_6 = p_2 \cup p_4$, $t_1 = t'_1 \cup t''_1$ и в результате порождаются дуги (p_5, t_1) и $(t_1 p_6)$. В порожденной сети p_5 — хвостовая, p_6 — головная позиция; переходу t_1 сопоставляется нагружающее его время

$$\tau = \begin{cases} \tau'_1 & \text{при } x = c_1; \\ \tau''_1 & \text{при } x = c_2; \end{cases}$$

дугам, входящим в хвостовую позицию p_5 из t_2 и t_4 . сопоставляются операционные функции φ_1 и φ_2 , ($\varphi_1: x: = c_1$; $\varphi_2: x: = c_2$), присваивающие атрибуту меток x соответствующие значения признака; дугам, следующим из головной позиции p_6 , сопоставляются значения c_1 и c_2 , используемые функцией R для управления потоками меток; для сети-результата формируется начальное маркирование CM_0 , равное объединению маркирований сетей-операндов, в котором метки сети $N1$ раскрашиваются цветом c_1 , а сети $N2$ — цветом c_2 .

Если необходимо выполнить функциональное наложение нагруженных сетей, то при слиянии дуг необходимо сделать конъюнктивное

объединение нагружающих функций исходных сетей с нагружающими элементами, вводимыми в процессе наложения переходов. Если сливаемые переходы имеют по несколько позиций-предшественников или преемников, то для каждой из этих позиций необходимо указать свою позицию-пару во второй сети, участвующую в операции наложения.

Операция функционального наложения сетей $HC = (N1, N2)$ применима при порождении сети, язык которой $L(HC)$ можно настроить на язык одной из сетей операндов — $L(N1)$ или $L(N2)$.

На рис. 27 приведен пример функционального наложения двух заданных сетей $N1$ и $N2$.

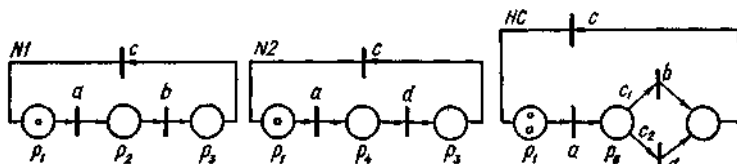


Рис. 27. Функциональное наложение сетей $N1$ и $N2$, порождающих нагруженную сеть $HC = (N1, N2)$

Порождаемая сеть $HC = (N1, N2)$ содержит всевозможные переходы, имеющиеся в сетях $N1$ и $N2$, причем в HC нет двух переходов, помеченных одинаково. Здесь множества переходов сетей $N1$ и $N2$ равны соответственно: $T_1 = \{a, b, c\}$ и $T_2 = \{a, c, d\}$. Наложение порождает новую сеть HC , с множеством переходов $T = \{a, bb, c, d\}$. Для наглядности метка сети $N1$ на рис. 27 представлена кружком, а метка сети $N2$ — квадратом. Обе эти метки введены в позицию p_1 сети HC . Из рисунка можно видеть, что метки первого и второго типов в сети HC , проходя по замкнутым маршрутам $W_1 = (abc...)$ и $W_2 = (adc...)$, соответственно, ожидают выполнения общих действий a и c и разветвляются в позиции p_6 для выполнения индивидуальных действий b или d . Хотя приведенные примеры достаточно просты, они подробно рассмотрены в связи с тем, что соответствующие возможности должен знать пользователь-проектировщик.

Укажем также на то, что выполнение описанной операции может выполняться в автоматизированном режиме с использованием ЭВМ. Директивы проектировщика могут определять степень наложения функций сетей. Возможен режим наложения всех одинаково помеченных переходов, или части их, определяемой оператором ЭВМ. Функциональное наложение можно использовать для упрощения сетей со сложной структурой взаимодействия между процессами и ресурсами, а также для устранения некорректных построений в сетях.

Так, в N -сети на рис. 28, а можно выполнить наложение подсетей S_1 и S_2 , обведенных штриховыми линиями, в результате чего получится нагруженная сеть (рис. 28, б), в которой обеспечена корректность предоставления ресурса R_1 процессам.

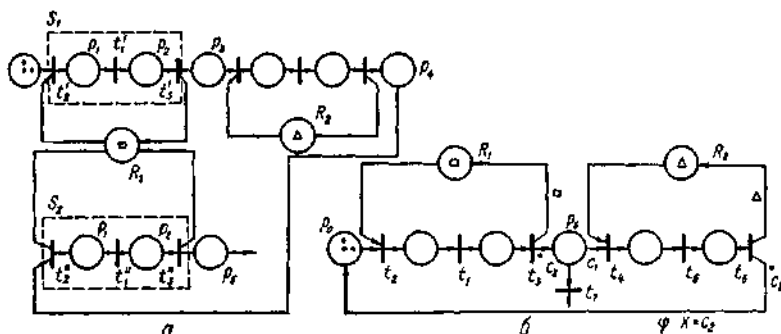


Рис. 28. Сеть с конфликтной позицией R_1 (а) и нагруженная сеть без конфликтных позиций (б)

Если в исходной сети возможно одновременное возбуждение переходов t'_2 и t''_2 , что создает конфликт заметку-ресурс из позиции R_1 , то в сети на рис. 28, б конфликтные позиции отсутствуют. Сеть на рис. 28, б функционально аналогична сети на рис. 28, а, однако она не имеет некорректных построений, связанных с конфликтами за метки-ресурсы. При выполнении операции ресурсного наложения сетей производится наложение ресурсных позиций сетей. Эти позиции предназначены для отображения хранилищ меток-ресурсов, которые закрепляются за метками-процессами на время выполнения работ. Именно такая операция определяет связывание взаимодействующих программ с ресурсами в мультипрограммном режиме эксплуатации вычислительной системы, создавая их мультипрограммный взаимосвязанный набор. На рис. 29 показан результат операции наложения двух сетей, имеющих общие ресурсные позиции R_1 и R_2 .

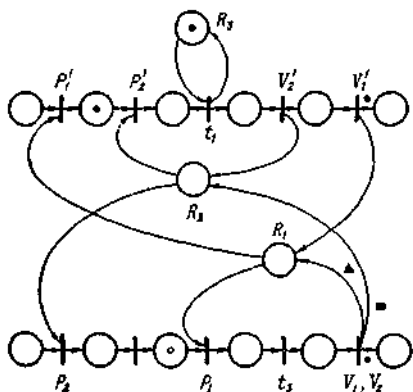


Рис. 29. К объяснению операции ресурсного наложения сетей, использующих общие ресурсные позиции

Исходные сети представлены верхней и нижней линейными последовательностями позиций и переходов. Прimitives, осуществляющие выделение и освобождение ресурсов, обозначены символами P и V , и приписаны соответствующим переходам. При наложении могут порождаться сети с существенно новыми свойствами, что следует учитывать при их изучении. Так, хотя каждая из исходных сетей является живой, их наложение может порождать сеть с тупиками. Именно такой случай показан на рис. 29. Заметим, что возникновение тупика возможно, если в биграфе существует замкнутый цикл, проходящий через две или более ресурсные позиции. Таким образом, анализ позволяет предсказать не только сам факт возможности тупика, но и обеспечивает локализацию тупика, а также нахождение состояния сети, ведущего к нему. А это означает, что можно принять определенные предохранительные меры по устранению самого тупика или его последствий. Указанный здесь метод поиска тупиков может использоваться наряду с рассматривавшимися в предыдущем параграфе методами уравнений состояния. Так же как в базовых сетях Петри, в структурированных нагруженных сетях могут использоваться операции последовательного присоединения, инерации (замыкания входа и выхода) связывания мониторами и т. д. Упомянутые операции не имеют каких-либо других особенностей, отличных от описанных и представленных на рисунках. Практическая важность рассмотренных операций над сетями состоит в обеспечении проектировщика инструментом автоматизированного структурного

анализа и преобразования процессов и систем, а также их моделей на основе методов композиции и декомпозиции. Операции над сетями опосредованно выражают операции над алгоритмами, аппаратными структурами и их объединением, обеспечивая экономическую эффективность или реализуемость жесткого временного режима ОУВС.

9.2.5. Сетевое представление и автоматизация подготовки алгоритмов распределенной обработки и управления

Особенность систем распределенной обработки и управления состоит в том, что действия алгоритмов и программ разворачиваются не только во времени, но и в пространстве. В добавление к физической распределенности процессорных средств распределяются данные и системное управление. В связи с децентрализацией управления обрабатывающие узлы становятся высокоавтономными.

Взаимодействие элементов системы организуется на основе семафоров, мониторов, асинхронных прерываний. Хотя уже предложены различные механизмы управления распределенной обработкой, все еще отсутствует общий подход к проектированию параллельных и распределенных программных систем, поддающийся автоматизации. Алгоритмизация параллельной и распределенной обработки базируется на оригинальных и общих методах параллельных вычислений, моделях распределенных алгоритмов, средствах распараллеливания программ и вычислений. Методы, модели и средства распараллеливания ориентируются на конкретный класс ВС. Так, достигнут большой прогресс в вопросах параллельных вычислений и программирования для векторных и конвейерных ВС. Эти системы в основном ориентированы на решение вычислительных задач и обработку больших массивов данных. В системах управления производственными процессами преобладают распределенные микропроцессорные сети и системы с классическим способом организации работы локальных узлов по принципу «одиночный поток команд — одиночный поток данных».

Под подготовкой алгоритмов распределенной обработки и управления подразумевается представление их в такой форме, которая обеспечит решение задачи о размещении распределенного алгоритма на технических средствах, даст возможность изучить пропускную способность (производительность) алгоритма, временной режим его использования. Разумеется, что представление распределенного алгоритма должно обеспечиваться соответствующими средствами анализа. Так как методы проектирования и анализа сетевых систем

применимы также к анализу и проектированию распределенных алгоритмов и распределенных программных систем, в алгоритмизации параллельной обработки целесообразна ориентация на графические подходы и сетевой анализ.

Большая часть моделей параллельных и распределенных программных систем является базой для реализации соответствующих языковых средств. Эти модели не предназначены для проектирования структур распределенной обработки. Нагруженные сети обладают алгоритмической полнотой в представлении распределенных алгоритмов, поэтому это тот класс моделей, на который можно ориентировать проектирование программно-технических средств управляющих вычислительных систем гибких производств. На рис. 21—23, 26—29 приведены модели программных систем и алгоритмов, интерпретируемых нагруженными сетями.

Интерпретация алгоритма с помощью НС-сети является естественной. Метки и их нагружающие атрибуты могут представлять команды со всеми их полями, данные произвольной структуры с их описаниями, управляющие переменные. Операции над данными в естественном виде представляются управляющими функциями НС-сети.

Альтернативные условия ветвления алгоритмов и циклы представимы с помощью раскрашенных меток и управляющих функций, приписываемых дугам. Параллельные ветви алгоритма и точки синхронизации естественно представляются с помощью переходов сети. Выполнение операторов алгоритма может связываться с действием перехода или дуги. Срабатывание перехода порождает в выходной позиции метку, атрибуты которой вырабатываются операционной функцией, нагружающей исходящую из перехода дугу. Интерпретация операции над метками-данными, выполняемыми под управлением процесса, показана на рис. 23. При выполнении операций над данными может порождаться результат, направляемый для выполнения следующего оператора (динамические данные) и одновременно могут изменяться характеристики и значения данных в памяти (статические данные). Такая ситуация также легко представима сетью.

Время выполнения элементарных операторов алгоритма можно оценить на основе одного из подходов, описанных в литературе. Временные свойства операторов связываются с переходами. Анализ корректности распределенного алгоритма и его потенциальных временных свойств можно выполнить на основе комплекса средств сетевого анализа, рассматриваемого в книге. Очевидно, что последующее вложение в конкретную структуру ОУВС ухудшит временные свойства реализаций алгоритма. Факт нарушения требований по оперативности распределенной обработки может

обнаружиться достаточно поздно, уже на этапе отладки алгоритма, после затраты значительных сил и средств. Во избежание этой неприятной ситуации необходимо изучение потенциальных возможностей распределенного алгоритма.

Алгоритм, описанный моделью-биграфом, позволяет оценить объемы данных, которые передаются для выполнения оператора из статической памяти, а также от другого оператора через канал связи, регистр или другое динамическое хранилище информации. Таким образом, НС-модель алгоритма позволяет найти объемы статических и динамических элементов распределенной памяти, потенциально необходимые для хранения информации в условиях его выполнения. Для конкретного класса распределенных алгоритмов по их НС-моделям можно построить наилучшую проблемно-ориентированную схему их реализации. Алгоритм распределенной обработки строят на основе анализа информационной связи между первичными фрагментами алгоритма. Эти первичные объекты можно представить в форме модулей, что позволит их легко понимать и модифицировать. При распараллеливании алгоритма необходимо специфицировать множество модулей (операторов) алгоритма и описать информационную зависимость между операторами. Эти задачи можно решить разными способами для различных алгоритмов. Некоторые из алгоритмов могут быть более приспособлены для конвейерной (или макроконвейерной) обработки, другие — для обработки на векторных процессорах. В алгоритмах распределенных систем управления чаще всего реализуется распараллеливание на уровне модулей, и реже — на уровне макроконвейера и векторов.

Алгоритмы для распределенных вычислений можно разрабатывать двумя способами: компиляцией алгоритмов, подготовленных на традиционных последовательных языках, либо на основе их прямого представления нагруженными сетевыми биграфами.

Первый способ предполагает, что начальное представление алгоритма выполняется на основе одного из языков программирования высокого уровня. Этот способ позволяет использовать ранее разработанные тексты программ для однопроцессорной обработки, извлекая из них алгоритм для распределенной системы.

Второй способ предполагает, что рабочим языком описания алгоритма является ЯНС (см. гл. 9.2.2).

Алгоритмы, описанные на последовательных языках высокого уровня, содержат следующие основные типы операторов: **присваивающие, условные, передачи управления, повторения или цикла. Этих четырех типов операторов достаточно, чтобы выразить основные концепции, требующиеся для автоматизации распараллеливания**

алгоритмов и программ. Так, операторы ввода-вывода можно заменить присваивающими, операторы вызова процедур можно заменить телом процедуры. Модули мультипрограммного использования представимы элементом обслуживания с очередью. Использование реентерабельных модулей представляется путем «раскрашивания» вызывающих процессов признаками-номерами, обеспечивающими возврат в точку вызова реентерабельной процедуры. Если в алгоритмическом языке используются операторы типа FORK и JOIN, они непосредственно интерпретируются переходами нагруженной сети.

Представление параллельного алгоритма требует выявления информации о взаимодействии элементарных фрагментов алгоритма. Взаимодействие фрагментов осуществляется путем передачи данных между ними. В связи с тем что человеку свойствен последовательный характер мышления, анализируемые им исходные фрагменты можно считать располагающимися последовательно. Информационную зависимость между последовательно анализируемыми фрагментами S и T устанавливают путем сопоставления множеств входных и выходных данных этих фрагментов алгоритма.

Обозначим: $I(S)$ — множество входных переменных оператора (фрагмента) S ; $O(S)$ — множество выходных переменных, создаваемых оператором S . Если при последовательном просмотре действий алгоритма фрагмент S предшествует фрагменту T , то будем писать $S < T$, указывая отношение следования в их анализе. Различают четыре типа информационных зависимостей между двумя фрагментами S и T , такими что S предшествует T в последовательности просмотра.

1. Фрагмент T зависим по данным от S , если $O(S) \cap I(T) \neq \emptyset$. Пересечение этих множеств дает те простые переменные, создаваемые S , которые используются T . Этот тип информационных связей называют также прямой ссылкой.
2. Если $I(S) \cap O(T) \neq \emptyset$, то пересечение этих множеств содержит такие переменные, которые используются S и вырабатываются T . Этот тип информационной связи можно назвать обратной ссылкой.
3. Фрагменты S и T зависимы по выходным данным (по выходу), если существует хотя бы одна переменная, которой присваивают значения как S , так и T ; зависимость по выходу имеет место, если $O(S) \cap O(T) \neq \emptyset$.
4. Фрагменты S и T связаны по управлению, если существует управляющая переменная, вырабатываемая фрагментом S и используемая фрагментом T .

Необходимо также выполнять анализ информационной связи для одного и того же фрагмента S .

Информационные зависимости позволяют установить направления ориентированных дуг между всеми фрагментами алгоритма и обеспечить построение его модели в графической сетевой форме.

На рис. 30 в качестве примера дано алгоритмическое описание

процесса вычислений суммы парных произведений $x_n = \sum_{i=1}^n a_i b_i$.

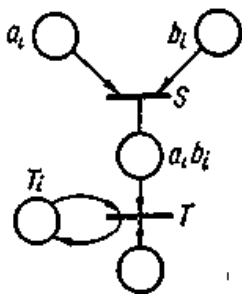


Рис. 30. Сетевая модель алгоритма вычисления суммы парных произведений

По условиям работы алгоритма предполагается, что операнды $a_i b_i$ поступают последовательно из внешней среды для выполнения действий очередного шага алгоритма.

Фрагменты (операторы) вычисления произведений $a_i b_i$ обозначены буквой S_i , а накопленные суммы парных произведений —

$T_i = T_{i-1} + a_i b_i$, $x_n = T_n$. Для данного циклического процесса

необходимо установить информационные связи между всеми

операторами S_i и T_i . Очевидно, что $I(T_i) = O(S_i)$;

$I(T_i) = O(T_i)$, что устанавливает направления дуг между позициями S_i и

T_i модели алгоритма. Из рис. 30 видно, что в данных условиях

возможен параллелизм в действиях S_{i+1} и T_i . Если не выполнится S_i

(не сработает переход S), то не создаются условия для выполнения действия T_i (для возбуждения перехода T).

Алгоритмическое представление процесса на рис. 30 допускает реализацию управления вычислениями по готовности данных.

Приведенная на рис. 23 модель предполагала использование

статических данных, размещаемых в памяти. Сеть на рис. 30

интерпретируется уравнением (2.4) возмущенного движения сети, в котором внешняя последовательность W_k — это последовательность

пар входных данных $a_i b_i$. Сетевая форма алгоритма обеспечивает представление конвейерности, параллелизма, управления потоками данных и команд. НС-форма алгоритма обеспечивает естественный переход к структурной реализации устройств для выполнения распределенных вычислений. Структурная реализация дает средство максимальной производительности для выполнения данных вычислений.

При структурной реализации алгоритма переходам сопоставляются процессорные устройства, позициям — элементы памяти — регистры, триггеры или запоминающие устройства более сложной структуры. Наличие метки в позиции-триггере используется с целью представления потоков управления. Метки-данные в позиции-регистре представляют элементы потока данных. Наличие данных в регистре p_i отмечается в маркировании M значением $m(p_i) = 1$.

На рис. 31 приведена схема вычислительной структуры, реализующей алгоритм, приведенный на рис. 30.

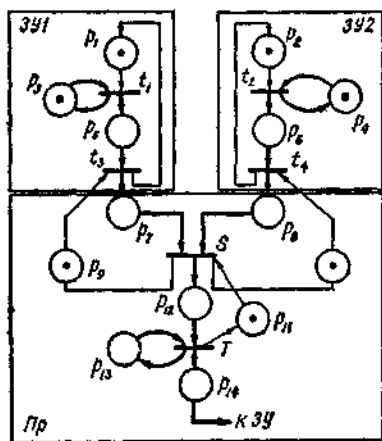


Рис. 31. Структурная реализация процессора вычисления суммы парных произведений:

3У — запоминающие устройства; p_1, p_{2s} — триггеры, управляющие операцией чтения данных в 3У; p_3, p_4 — хранилища информации 3У, p_5, p_6 — выходные регистры 3У; p_7, p_8 — входные регистры Пр; $p_9 - p_{11}$ — управляющие триггеры (двоичные семафоры); p_{12}, p_{13} — регистры произведений S_i и сумм T_i ; p_{14} — выходной регистр Пр; t_1, t_2 — операции чтения в 3У; t_3, t_4 — прием данных из p_5 и p_6 в p_7 и p_8 ; S и T — схемы, вычисляющие S_i и T_i

Элементы S и T модели процессора на рис. 31 соответствуют операторам S и T алгоритма рис. 30.

На этапе анализа структуры устройства необходимо определять безопасность, ограниченность, живость и другие свойства сетевой структуры, проверяя корректность разработки.

Применяя операции над сетями, можно формальными методами решить вопрос о совмещении нескольких функций в одном устройстве, после связывания с одним общим ресурсом организовать управление совместным его использованием. Если в результате моделирования и анализа установлено, например, что в сети существуют позиции p_i и p_j , через которые потоки данных не проходят одновременно, то они совмещаются, обеспечивая уменьшение числа регистров или других средств хранения информации в схемных реализациях. Если выполнение алгоритма организовать на структуре, не сбалансированной с моделью алгоритма, то необходимо установить связь между моделью алгоритма и моделью вычислительной структуры, связав их взаимодействие с помощью соответствующих средств операционного управления (семафоров, мониторов и т. п.). Единая модель «алгоритм — средства управления взаимодействием — ресурсы» представляется в такой же форме, как и каждый из элементов отдельно, — нагруженной сетью. Применяя указанные правила определения информационной зависимости, можно найти фрагменты распределенного алгоритма, которые могут выполняться независимо на различных процессах системы. Выделив множества сильно связанных фрагментов алгоритма, имеющих как прямые, так и обратные ссылки по данным, их выполняют на одном процессоре, уменьшая число обменов. Объединив сильно связанные фрагменты в макрофрагменты и исследовав сетевую модель алгоритма из макрофрагментов, можно оценить эффективность его конвейерной реализации.

Богатым источником параллелизма являются циклы. Известны различные способы распараллеливания циклических алгоритмов. Наиболее практичные в массовом применении методы расщепления циклов, использующих несвязанные данные, и фронта волны вычислений, выявляющие возможные последовательности параллельной обработки данных в рекуррентных циклах над массивами.

На рис. 32 приведена схема действий при конвертировании циклов в операторы для параллельных вычислений.

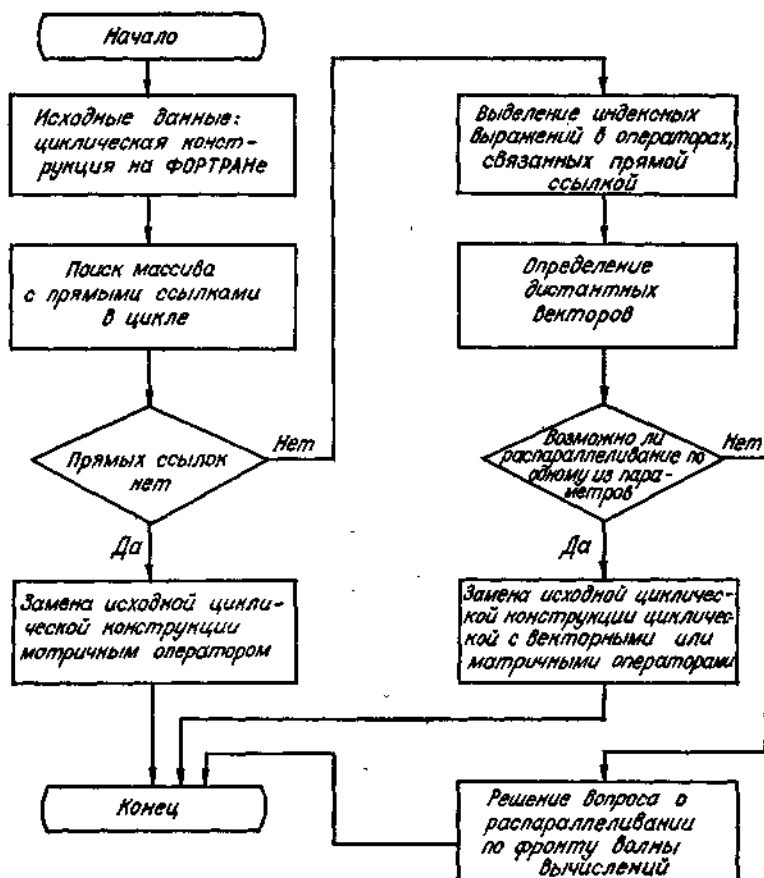


Рис. 32. К объяснению конвертирования циклов в программы для параллельных вычислений

При выполнении действий по трансформации распределенных программ среди операторов или их последовательностей могут выделяться те, которые эффективно реализуются на проблемно-ориентированных процессорах.

Программное обеспечение для построения и анализа распределенных алгоритмов предполагает объединение в **единый комплекс средств:** автоматизированной подготовки графических образов распределенных алгоритмов по их описанию на традиционном языке высокого уровня; анализа информационной связи в блочно-

параллельных и циклических программных структурах, конвейерных и макроконвейерных алгоритмах; имитации исполнения алгоритмов и оценки их эффективности; диагностики алгоритмов, обеспечивающих исследование корректности распределенной обработки. Для отладочной имитации и оценки эффективности распределенных алгоритмов можно применять стандартные средства имитационной отладки, представленные операционной системой моделирования и управления. **Результат распараллеливания алгоритма представляется:** графическим образом алгоритма; формальной спецификацией графического образа соответствующими структурами данных; результатами оценки корректности и качества распределенных действий алгоритма. Подобным результатом может также завершаться спецификация процессов распределенной технологии производственной системы.

9.2.6. Математическое обеспечение для расчета сетей с очередями как моделей ОУВС и ГАПС

Представив общие вопросы анализа сетей с очередями, опишем наиболее эффективные и качественно ясные способы их расчета, называемые **методами анализа средних**. (Следуя современной терминологии, наравне можно использовать термины: сеть с очередями, сеть массового обслуживания и стохастическая сеть.) Сеть состоит из узлов или систем массового обслуживания, пронумерованных числами $i = 0, 1, \dots$. **Объект, который обслуживается в узлах, называют заявкой (запросом, сообщением, процессом).** Заявка может представлять как **информационные, так и материальные объекты**. Маршрут заявок описывает последовательность проходимых узлов и может характеризовать как детерминированные, так и случайные процессы (см. раз. 9.2.1). Рассмотрим сети с активными ресурсами (узлами), трудоемкость работы в которых характеризуется временем v_{ir} , где $r = 1, 2, \dots, R$ — тип заявки и ее цепь. Если r -заявки (заявки типа r) поступают в сеть из внешнего источника и после обслуживания покидают ее, сеть называется открытой (разомкнутой) по отношению к цепи r . Сеть, не имеющая внешних источников, называется замкнутой. В смешанных сетях существуют как открытые, так и замкнутые цепи заявок. В зависимости от области приложений сети с несколькими типами заявок называют либо многоцепными либо многопродуктовыми. В замкнутых цепях назначают узел, возможно даже фиктивный, который принимают

за начало и конец маршрута. Некоторые из узлов могут повторяться в маршрутной цепи несколько раз. Число α_{ir} , характеризующее число визитов в узел i заявок маршрута r , называют **коэффициентом посещения (передачи)**. Коэффициент посещения в стационарном режиме обслуживания можно определить из отношения

$$\alpha_{ir} = \lambda_{ir} / \lambda_{0r}, \quad (2.23)$$

где λ_{0r} — интенсивность потока r -заявок из начального узла маршрутной цепи заявок; λ_{ir} — интенсивность потока заявок в узел i . Для разомкнутой цепи величину λ_{0r} задают. Для замкнутой цепи величина λ_{0r} определяется через множество параметров сети и характеризует ее производительность (пропускную способность). В стохастической сети движение r -заявок описывают маршрутной матрицей вероятностей переходов $P_r = \|p_{ij}^r\|$, где p_{ij}^r — вероятность того, что r -заявка после обслуживания в узле i переходит к узлу j . В стационарном режиме обслуживания для каждого из узлов записывают следующее условие баланса потоков:

$$\lambda_{ir} = \sum_{j=0}^N \lambda_{ji}^r. \quad (2.24)$$

Поскольку p_{ij}^r характеризует долю r -заявок, переходящих из узла i в узел j , то

$$\lambda_{ji}^r = \lambda_{ij}^r p_{ji}^r. \quad (2.25)$$

Подставив выражение (2.25) в уравнение (2.24), получим систему уравнений

$$\lambda_{ir} = \sum_{j=0}^N p_{ji}^r \lambda_{jr}; \quad i = \overline{0, N}. \quad (2.26)$$

Поскольку для разомкнутой цепи задают поток из внешнего источника λ_{0r} , то из уравнений (2.23) и (2.26) можно найти λ_{ir} и α_{ir} . Для замкнутой цепи уравнения баланса потоков (2.26) представляются однородной системой с бесконечным множеством решений. Поэтому для расчета процессов в замкнутых цепях в качестве исходных данных берут величины α_{ir} . Поскольку за полный цикл заявка посещает начальный узел трассы один раз, коэффициент посещения нулевого узла равен единице. Учитывая, что $\alpha_{0r} = 1$ и подставляя $\lambda_{ir} = \alpha_{ir} \lambda_{0r}$ в систему уравнений (2.26), получаем уравнения для расчета коэффициентов посещения замкнутой цепи.

$$\alpha_{ir} = \sum_{j=0}^N p_{ji}^r \alpha_{jr}.$$

Итак, для спецификации маршрута процесса в сетевой модели необходимо задать либо вектор коэффициентов посещения, либо

матрицу вероятностей перехода, которая определяется по свойствам алгоритмов, закладываемых для реализации в управляющую вычислительную систему. Та часть маршрута, которая относится к объекту управления автоматизированного производства, специфицируется технологическим маршрутом изделий. Если маршрут заявки детерминирован, он сразу описывается коэффициентами посещения, поскольку число визитов в каждый из узлов сети определено. Стохастический же маршрут представляется матрицей P . Внутри каждой цепи переходы заявок могут задаваться не на множестве устройств $i = 1, 2, \dots, N$, а на множестве пар (i, l) , где l — параметр, характеризующий локальный класс заявок, $l = 1, 2, \dots, L$. Если $L \neq 0$, то сеть называют много- или мультиклассовой. Каждый из маршрутов многоцепной мультиклассовой сети характеризуется следующей матрицей вероятностей передач: $P_r = |p_{il,js}|$. Компоненты матрицы P_r задают вероятности переходов заявок к обслуживанию классом s в узле сети j , после обслуживания их в узле i классом l . Состояние заявки внутри каждой цепи характеризуется парой (i, l) , что позволяет с помощью $p_{il,js}$ отражать сложные траектории движения заявок и строить такие модели реальных систем, которые обладают большей достоверностью по сравнению с одноклассовыми.

Интенсивность потока заявок в класс s системы i из других систем сети обозначим λ_{is} . Уравнения баланса потоков стационарного режима сети имеют вид:

$$\lambda_{is} = \sum_{j=0}^N \sum_{l=1}^L \lambda_{jl, is}. \quad (2.27)$$

Для замкнутой сети из выражения (2.27) можно определить лишь коэффициенты передачи по отношению к некоторой начальной точке маршрута, которая представляется парой $(i, l) = (0, 1)$.

Коэффициент посещения $\alpha_{is} = \lambda_{is} / \lambda_{01}$ характеризует число запросов на обслуживание s -го типа в узле i за полный цикл заявки. Поскольку $(0, 1)$ — начальная точка маршрута, то $\alpha_{01} = 1$. С учетом этого факта, а также принимая во внимание, что $\lambda_{is} = \alpha_{is} \lambda_{01}$, из выражения (2.27) получаем систему уравнений для вычисления коэффициентов посещения

$$\alpha_{is} = \sum_j \sum_l \alpha_{jl} p_{jl, is}. \quad (2.28)$$

Разработаны эффективные способы расчета сетей, реализующих в узлах следующие способы обслуживания: обслуживание в порядке поступления (FIFO); разделение времени (PS), предполагающее, что если в узле находится n запросов, то в единицу времени каждому из

них будет представлен квант обслуживания длиной $1/n$; прерывание на основе абсолютных приоритетов с дообслуживанием в обратном порядке; обслуживание без ожидания (D). Первые три способа представляют узлы, обслуживающие с ожиданием (узлы типа 1). Узлы второго типа, обслуживающие без ожидания, представляют индивидуальные ресурсы, закрепленные за процессом. Обозначим: среднее число заявок класса l в узле i — n_{il} ; среднее число заявок, в узле i — $n_i = \sum_l n_{il}$; среднее число заявок цепи r — $K_r = \sum_i n_{ir}$;

число заявок в сети —

$$K = \sum_r K_r.$$

Состояние сети описывается вектором $\bar{n} = (\bar{n}_1 \bar{n}_2 \dots \bar{n}_i \dots \bar{n}_N)$, где $\bar{n}_i$ — состояние узла i . В случае многопоточковой одноклассовой сети состояние узла представляется набором $\bar{n}_i = (n_{i1}, \dots, n_{ir}, \dots, n_{iR})$. Если $p(\bar{n})$ — вероятность состояния сети, то их сумма по всем состояниям сети

$$\sum_{\bar{n}} p(\bar{n}) = 1.$$

Основной подход к расчету сетей состоит в том, что делается группа предположений, используемых в теории стохастических процессов: система работает в стационарном режиме, процессы движения заявок стохастически независимы, задания переходят от узла к узлу по марковской цепи в соответствии с матрицей вероятностей передач $P = |p_{ij}|$, распределение времени обслуживания в узлах экспоненциально и т. д. Характеристики сетей находят через вероятности ее состояний.

Базовые сети очередей — это подмножество общего класса сетевых моделей, для которых распределение вероятностей состояний представляется в особо простой аналитической форме. Анализ таких моделей можно выполнить за короткое время, поэтому их называют быстрыми. Можно выделить также модели, которые позволяют получить точные результаты, ко время их нахождения велико. Такие модели называют медленными. И, наконец, для некоторых моделей неизвестны методы анализа, гарантирующие получение точных результатов. В этом случае применяют приближенные методы, основывающиеся на использовании базовых моделей.

К медленным моделям можно отнести многие марковские модели сложных систем. Практическое решение систем с несколькими тысячами состояний упирается в проблему времени. В других случаях

может потребоваться большой объем памяти. К медленным могут относиться имитационные модели. Хотя они обладают высоким правдоподобием в отображении процессов, эксплуатация имитационных моделей может быть весьма дорогостоящей. В этих случаях применяют простые модели для получения границ изменения характеристик более сложных моделей.

При выборе между применением аппроксимации или имитационного моделирования для анализа особенно медленных и неразрешимых моделей следует учитывать такой фактор как достоверность (правдоподобие) модели. Программная имитация часто может рассматриваться как приближенный метод, так как не дает точных результатов. Например, оценка среднего времени ответа при имитации бывает близкой к точному значению, но обычно не равна ему. Использование же численного решения сетевой модели дает точное значение среднего времени ответа.

Для расчета сетей, которые здесь для удобства названы *базовыми*, используется возможность представления вероятностей их состояний в форме произведения

$$P(s_1, s_2, \dots, s_N) = (p_1(s_1) p_2(s_2) \dots p_N(s_N)) / G, \quad (2.29)$$

где $p_i(s_i)$ — вероятность того, что i -я система сети находится в состоянии $s_i = 0, 1, \dots, K$; K — число заявок, циркулирующих в сети; G — нормализующая константа, выбираемая такой, чтобы сумма вероятностей всех состояний сети была единичной.

Выражение вероятности состояния сложной системы в виде произведения вероятностей состояний более простых элементов сети связано с поиском «независимости» между различными узлами сети, или, что то же самое, с использованием декомпозиции (структурирования). Основой классических алгоритмов вычисления $G(K)$ является операция свертки нескольких векторов, которая представима в виде рекуррентных вычислений по многомерной схеме Горнера. Однако, хотя конечные результаты вполне представимы в диапазоне чисел с плавающей запятой современных ЭВМ, промежуточные результаты при расчетах могут легко превысить уровень плавающей точки, т. е. привести к переполнению. Проблема переполнения требует масштабирования и для сложных сетей существенно усложняет задачу моделирования, делая ее в ряде случаев вообще не решаемой в приемлемые сроки.

При расчете замкнутых сетей используют также рекуррентные процедуры над такими характеристиками как средняя длина очереди, среднее время ожидания. Этот подход называют методом анализа средних (МАС). Алгоритмы свертки сложны в том смысле, что часто за математическими соотношениями и процессами вычислений не ясен

их содержательный (прикладной) смысл. Метод анализа средних основан на ясных содержательных трактовках и разработан для решения численных проблем, возникающих в алгоритмах свертки. Поскольку априорно невозможно указать эффективность того или иного алгоритма в условиях конкретной задачи и конкретной ЭВМ системы проектирования, в составе средств САПР реализуются различные алгоритмы расчета.

Разомкнутые сети. Представим математическое обеспечение для расчета однородных экспоненциальных сетей с несколькими потоками заявок. Математические модели названного класса описывают следующими исходными данными: интенсивностью внешних источников пуассоновских потоков — λ_{0r} ; экспоненциально распределенной трудоемкостью обслуживания в узле i — $\nu_i = 1/\mu_i$; коэффициентами посещения узлов i — α_{ir} . Доказано, что в этих условиях сеть математически декомпозируется на множество несвязанных узлов. Характеристики сети рассчитывают следующим образом. Загрузка узла i со стороны потока заявок типа r

$$\rho_{ir} = \lambda_{ir} \vartheta_i = \alpha_{ir} \lambda_{0r} \vartheta_i,$$

тогда суммарная загрузка узла

$$\rho_i = \sum_r \rho_{ir}.$$

Среднее время ожидания до начала обслуживания в узле i вычисляют по формуле $W_{ir} = W_i = \rho_i \vartheta_i / (1 - \rho_i)$. Время пребывания $V_{ir} = V_i = W_i + \vartheta_i = \vartheta_i / (1 - \rho_i)$. Воспользовавшись формулой Литла, найдем число заявок каждого типа в узле i :

$$n_{ir} = \lambda_{ir} V_{ir} = \rho_{ir} + \lambda_{ir} W_{ir}.$$

Полное время пребывания заявки r в сети

$$V_r = \sum_i \alpha_{ir} V_{ir}.$$

Замкнутые сети. Алгоритм расчета сети через вероятности состояний. Для замкнутых марковских сетей вероятности состояний определяются из решений, представимых в форме произведения (2.29). Если сеть состоит из FIFO-узлов, то вероятности состояний

$$P(n_1 n_2 \dots n_N) = \frac{1}{G(K)} \prod_{i=1}^N \rho_i^{n_i}, \quad (2.30)$$

где $\rho_i = \lambda_i / \mu_i$ — загрузка i -й системы, равная отношению интенсивности потока к интенсивности обслуживания в i -й системе;

$$G(K) = \sum_A \prod_{i=1}^N \rho_i^{n_i}. \quad (2.31)$$

Здесь A — множество векторов состояний системы $(n_1 n_2 \dots n_N)$, для которых $K \geq n_i \geq 0$, при условии

$$\sum_i n_i = K. \quad (2.32)$$

Загрузка i -й системы с учетом коэффициентов посещения $\alpha_i = \lambda_i / \lambda_0$

$$\rho_i = x_i / \lambda_0, \quad (2.33)$$

где $x_i = \alpha_i v_i$.

Из соотношений (2.30) — (2.33) можно найти, что

$$P(n_1 n_2 \dots n_N) = \prod_{i=1}^N x_i^{n_i} / \sum_A \prod_{i=1}^N x_i^{n_i}. \quad (2.34)$$

Таким образом, первый шаг алгоритма предполагает вычисление по формулам (2.34) вероятностей всех возможных состояний сети. На втором шаге эти значения используются для расчета характеристик сети. Например,

$$\rho_i = 1 - \sum_{n_i=0} P(n_1 \dots n_N),$$

где суммирование ведется по всем состояниям множества A , для которых число заявок в системе равно нулю,

$$n_i = \sum_j j p(n_i = j); \quad j = 1, \dots, K,$$

где $p(n_i = j)$ — суммарная вероятность всех состояний из множества A , для которых $n_i = j$.

Из выражения (2.33) следует, что пропускная способность сети $\lambda_0 = \rho_i / \alpha_i v_i$, а время пребывания, определяемое по формуле Литла, $V_i = n_i / \lambda_i = n_i / \alpha_i \lambda_0$. Недостаток алгоритма заключается в необходимости хранения возможных состояний сети. Однако отвергать этот алгоритм не следует, так как он позволяет рассчитать любые тонкие свойства сети.

Алгоритм свертки предусматривает последовательное нахождение нормализующих констант $G_1, G_2, \dots$ и на их основе — необходимых характеристик сети. Для двухпоточковой сети

$$G_i(m, n) = G_{i-1}(m, n) + x_i G_i(m-1, n) + y_i G_i(m, n-1),$$

где

$$x_i = \alpha_{i1} v_{i1}; \quad y_i = \alpha_{i2} v_{i2}; \quad G_i(0, n) = G_i(m, 0) = 0; \quad G_0(m, n) = 0; \\ m = 1, 2, \dots, K1; \quad n = 1, 2, \dots, K2; \quad i = 1, 2, \dots, N.$$

Рекуррентные вычисления дают в конце каждого двумерного цикла нормализующие константы $G(K1, K2) = G_N(K1, K2)$. Значение $G(K1, K2)$ используют для вычисления характеристик узлов сети.

Загрузка узла i со стороны каждого из потоков и их сумма

$$\begin{cases} \rho_{i1}(K1, K2) = x_i G(K_1 - 1, K2) / G(K1, K2); \\ \rho_{i2}(K1, K2) = y_i G(K_1, K_2 - 1) / G(K1, K2); \\ \rho_i = \rho_{i1} + \rho_{i2}. \end{cases}$$

Пропускная способность цепей

$$\lambda_{01} = G(K1 - 1, K2) / G(K1, K2);$$

$$\lambda_{02} = G(K1, K2 - 1) / G(K1, K2).$$

Число заявок каждого из двух типов в узле i

$$n_{i1}(K1, K2) = \rho_{i1}(K1, K2) [1 + n_i(K1 - 1, K2)];$$

$$n_{i2}(K1, K2) = \rho_{i2}(K1, K2) [1 + n_i(K1, K2 - 1)].$$

Премия пребывания в узлах i заявок каждой из цепей

$$V_{i1}(K1, K2) = \Phi_{i1} [1 + n_i(K1 - 1, K2)];$$

$$V_{i1}(K1, K2) = \Phi_{i2} [1 + n_i(K1, K2 - 1)]. \quad (2.35)$$

Приведенные зависимости верны, если в FIFO-узлах время обслуживания не зависит от цепи, т. е. $v_{i1} = v_{i2}$. В PS-узлах оно может различаться. При необходимости через G можно найти вероятности состояний и другие характеристики. Основной недостаток алгоритма — большой диапазон изменения величины G , приводящий к переполнению, потере значности, погрешностям округления.

Метод анализа средних может быть легко получен из алгоритма свертки и формул Литла для цепи и узлов сети. Так, в случае одноцепной сети при $K1 = K$ из выражения (2.35) имеем

$$V_i(K) = \Phi_i [1 + n_i(K - 1)]. \quad (2.36)$$

Исходя из формулы Литла для цепи и учитывая, что время полного цикла заявки в сети

$$C(K) = \sum_i \alpha_i V_i(K), \quad (2.37)$$

получаем

$$\lambda_0(K) = K / C, \quad (2.38)$$

и из формулы Литла для узла находим

$$n_i(K) = \alpha_i \lambda_0(K) V_i(K). \quad (2.39)$$

Заметим что $n_i(0) = 0$. Рекуррентные вычисления по формулам (2.35) — (2.38) сразу дают искомые характеристики процессов и узлов сети. Загрузку находят из соотношения $\rho_i(K) = \alpha_i \lambda_0(K) v_i$.

Рассмотрим уравнение (2.36). Время пребывания заявки и узле i складывается из времени обслуживания v_i , и времени ожидания, равного

$$W_i(K) = \Phi_i n_i(K - 1). \quad (2.40)$$

Из выражения (2.40) следует, что если число заявок сети равно K , то в момент прихода в узел i очередной заявки их среднее число в узле равно $n_i(K-1)$, а поскольку каждая из заявок требует времени обслуживания v_i , необходимо подождать $v_i n_i(K-1)$ единиц времени до начала обслуживания пришедшей в узел заявки.

Содержательная трактовка формулы Литла $n_i(K) = \lambda_i(K) \times V_i(K)$ может быть следующей. В единицу времени в узел i приходит λ_i заявок. За время от момента прихода заявки в узел до момента ухода из узла, равное V_i в узле накапливается $\lambda_i V_i$ заявок. В стационарном режиме обслуживания их число в узле сохраняется. Просуммировав n_i по всем узлам сети

$$K = \sum_i n_i(K) = \lambda_0 \sum_i \alpha_i V_i(K),$$

получим формулу (2.38)

Рассмотрим более общий случай обслуживания в узлах. Введем величину $b_i(j)$, характеризующую емкость узла i , когда в нем находится j заявок. Для одноканальных обслуживающих приборов с постоянной скоростью обслуживания $b_i(j) = 1$. Для D-узлов $b_i(j) = 1$. Для узлов, скорость обслуживания в которых зависит от нагрузки, имеем $0 < b_i(j) < \infty$. Обычно $b_i(j)$ — монотонная неубывающая функция j , т. е. $b_i(j) > b_i(j-1)$ и

$$b_i(j+1) - b_i(j) \leq b_i(j) - b_i(j-1).$$

Например, если в узле находится двухканальный обслуживающий прибор, то $b_i(1) = 1$, $b_i(j) = 2$ для всех $j \geq 2$. Емкость узла (нагрузочная способность) определяется отношением

$$b_i(j) = \mu_i(j)/\mu_i(1), \quad (2.41)$$

где $\mu_i(j)$ — интенсивность обслуживания в узле i , если в нем находится j заявок, а $\mu_i(1)$ — если одна заявка.

Если скорость обслуживания в узле зависит от нагрузки, как это описано формулой (2.41), то время пребывания можно вычислить по формуле

$$V_i(K) = \phi_i \left[1 + n_i(K-1) + \sum_{j=1}^{b_m} j \left(\frac{1}{b_i(j)} - 1 \right) P_i(j-1, K-1) \right], \quad (2.42)$$

где b_m — максимальная нагрузочная способность узла $b_m \leq K$;

$$P_i(j, K) = \frac{1}{b_i(j)} \phi_i \lambda_i(K) P_i(j-1, K-1) \quad (2.43)$$

— вероятность того, что в i -м узле находится j из всех K заявок, циркулирующих в сети;

$$P_i(0, K) = 1 - \sum_{j=1}^K P_i(j, K) \quad (2.44)$$

— вероятность того, что узел j пуст.

Для одноканального FIFO-узла из формулы (2.42) получается выражение (2.36), а для D-узла

$$V_i(K) = \Phi_i. \quad (2.45)$$

Алгоритм расчета характеристик методом анализа средних

1. Находят начальные условия: $n_i(0) = 0$.
2. Вычисляют $V_i(K)$ по формулам (2.36), (2.42) — (2.44) или (2.45), если узел, соответственно, является одноканальным FIFO или PS прибором, прибором с переменной скоростью обслуживания, или D-прибором, обслуживающим без ожидания.
3. Находят $\lambda_0(K)$ по формулам (2.37) и (2.38).
4. Вычисляют распределение заявок по узлам по выражению (2.39).
5. Если не достигнуто заданное K , то $K = K + 1$, перейти к шагу 2, иначе — конец.

Приведем один из алгоритмов расчета многоцепных сетей с одноканальными FIFO-, PS- и D-узлами. Обозначим: $K = (K_1 K_2 \dots K_R)$ — вектор числа заявок сети; E_r — единичный r -вектор; $K - E_r$ — вектор, содержащий в цепи r на одну заявку меньше, чем вектор K :

$$K - E_r = (K_1 K_2 \dots K_{r-1} \dots K_R).$$

1. Вводят начальные условия: $n_{ir}(0) = n_i(0) = 0$.
2. $V_{ir}(K) = v_{ir}[1 + \delta_i n_{ir}(K - E_r)]$,

где

$$\delta_i = \begin{cases} 1 & \text{для FIFO-, PS-узлов;} \\ 0 & \text{для D-узлов.} \end{cases}$$

$$\lambda_{0r}(K) = K_r / \sum_i \alpha_{ir} V_{ir}(K).$$

3.

$$n_{ir}(K) = \lambda_{0r}(K) \alpha_{ir} V_{ir}(K); \quad n_i(K) = \sum_r n_{ir}(K).$$

4.

5. Возврат к шагу 2, если не достигнута точка с заданным значением вектора K , иначе — конец

Порядок обхода узлов для двухцепного случая показан на рис. 33, $K = (K_1, K_2) = (2, 2)$, т. е. в каждой из цепей движется по две заявки.

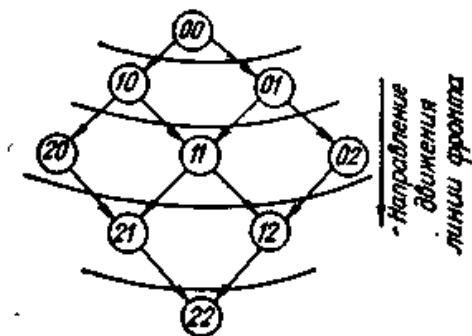


Рис. 33. Порядок обхода узлов при расчете двухцепной сети с популяцией $K = (2, 2)$

Приведенный фронт волны вычислений позволяет соответствующим образом организовать цикл прохода от точки (00) к точке (K_1 , K_2). Изображенный на рис. 33 граф — это не диаграмма состояний сети, а диаграмма возможных объемов заявок в сети, содержащая несравнимо меньшее число вершин, чем диаграмма состояний. Число вершин в графе не зависит от числа узлов в сети. Вычислительная сложность алгоритмов свертки, метода анализа средних и алгоритмов локального баланса имеет один и тот же порядок. Однако в зависимости от особенностей исходных данных тот или иной из алгоритмов может давать меньшие погрешности, которых невозможно избежать в силу рекуррентного характера счета. По существу здесь реализуются различные схемы вычислительных процессов, но предпочесть какую-либо по соображениям численной устойчивости трудно. При инженерных исследованиях и расчетах предпочтительнее содержательно ясные алгоритмы метода анализа средних. Для численного контроля результатов можно использовать одновременные расчеты по нескольким различным алгоритмам.

Смешанные сети. В смешанных сетях можно выделить потоки с постоянным числом процессов (замкнутые цепи) и потоки, число процессов в которых зависит от интенсивности внешних источников и характера обслуживания. На рис. 34 приведен пример, из которого видно, что цепь 2 — разомкнутая, цепь 3 — замкнутая.

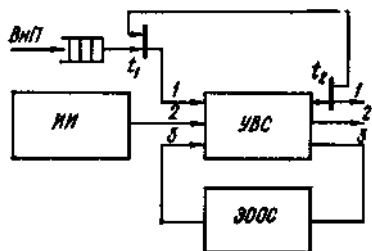


Рис. 34. Структурная схема системы управления, представленная смешанной сетью:

ВнП — внешний пакет запросов; *МО* — инициативные источники объекта управления; *ЭООС* — элементы объекта, охваченные замкнутым контуром управления через *ОУВС*; t_1 — разрешение ввода; t_2 — конец обработки запроса из *ВнП*

Цепь 1 хотя и имеет внешний источник, но в определенных условиях может считаться замкнутой. Если число процессов в цепи 1 ограничено некоторой величиной, которая поддерживается постоянной за счет внешнего пакета запросов, цепь по отношению к *ОУВС* может считаться замкнутой. Разомкнутые цепи могут порождаться и внутренними источниками, находящимися в самой *ОУВС*. Так, если первичные задачи замкнутой цепи порождают для параллельного выполнения вторичные, то они могут относиться к разомкнутой цепи. Интенсивность потока запросов разомкнутой цепи здесь не задана и зависит от параметров обеих цепей.

Каждая из цепей описывается своими параметрами. Для исходных данных разомкнутой цепи должно выполняться следующее условие устойчивости: $\rho_{ip} < 1$, где p — признак разомкнутой цепи (аналогично, z — замкнутой). При расчете смешанной сети выполняют приведение разомкнутой цепи к замкнутой.

Пусть $\bar{\rho}_{ip} = 1 - \rho_{ip}$ — часть мощности i -го узла, не используемая процессами разомкнутой цепи.

Алгоритм расчета смешанной сети

1. Рассчитывают $\rho_{ip} = \lambda_{ip} \vartheta_{ip}$; $\bar{\rho}_{ip} = 1 - \rho_{ip}$; $\rho_{ip} < 1$.
2. Корректируют значения времен обслуживания замкнутой цепи: $\bar{\vartheta}_{iz} = \vartheta_{iz} / \bar{\rho}_{ip}$.

Рассчитывают характеристики замкнутой цепи смешанной сети, где в качестве исходных данных используют скорректированные значения

времен обслуживания $\bar{\vartheta}_i$ замкнутой цепи. Используют один из рассмотренных выше методов.

4. Рассчитывают характеристики разомкнутой цепи смешанной сети $n_{ip} = \rho_{ip} (1 + n_{iz}) / (1 - \rho_{ip})$; $V_{ip} = \vartheta_{ip} + n_{ip} \times \vartheta_{iz} + W_{iz}$. При расчете на шаге 3 необходима проверка условия корректности $\rho_{iz} < \bar{\rho}_i$, которое обеспечивает условие устойчивости разомкнутой цепи.

Описанные алгоритмы составляют основу математического обеспечения САПР для расчета сетевых моделей с очередями.

9.3. Методы анализа и оптимизации процессов в облачных управляющих вычислительных системах

9.3.1. Взаимодействие процессов в мультипрограммных ОУВС и средства-советчики САПР

Типичным режимом эксплуатации ОУВС является режим мультипрограммирования. Он реализуется для обеспечения группового управления как одного из основных принципов управления автоматизированным производством. Кроме того, мультипрограммная эксплуатация позволяет использовать общую базу данных и создает условия для эффективного использования вычислительных мощностей аппаратных средств системы. При мультипрограммной эксплуатации ОУВС возникают задачи, связанные с выбором оптимального числа обслуживаемых элементов объекта из условий режима реального времени, обеспечивающих высокую загрузку управляющей аппаратуры. На рис. 35 приведена структура управляющей системы, содержащей центральный процессор ЦП, внешнюю память ВП и процессор группового управления ПргУ.

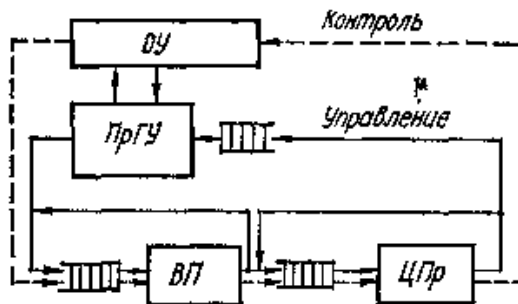


Рис. 35. Модель ОУВС, выполняющей задачи контроля и управления

ОУВС выполняет задачи управления и контроля состояния объекта. На схеме отражены объект управления *ОУ*, аппаратные ресурсы ОУВС, очереди мультипрограммной операционной системы и трассы процессов контроля и управления. *ЦПр* выполняет обслуживание в режиме разделения времени с квантованием. *В/П* обслуживает запросы в порядке их поступления. Трудоемкости задач контроля и управления различны. Отличаются также ограничения, накладываемые на временной режим прохождения каждой из задач. Так, для задач управления задают ограничение на время цикла, определяющее интервал времени между двумя последовательными взаимодействиями процессора группового управления и объекта. Задачи контроля целесообразно выполнять как можно чаще и в большем числе точек. Чем больше точек контроля, тем больше коэффициент мультипрограммной обработки по этой группе задач ОУВС. Расчет режима мультипрограммного взаимодействия в ОУВС на основе усреднения характеристик потоков здесь недопустим. Имитационное моделирование многопоточкового режима требует значительных затрат времени. Поэтому возникает необходимость в построении аналитических моделей многопоточковых ОУВС, на которых можно анализировать круг вопросов, связанных с взаимодействием потоков в системе.

Расчет приведенной системы выполнен на основе многопоточковой сетевой модели и соответствующего программного обеспечения САПР.

Задачи управления требуют выполнения операций ввода исходных данных из внешней в основную память и *ПрГУ*, а также обработки в

ЦПр. После этого следует запуск *ПрГУ*, осуществляющего управляющие операции. Задачи контроля проходят по иной трассе. Временем считывания контрольной информации на объекте по сравнению с другими временами пренебрегаем. После считывания выполняются запоминание данных во внешней памяти, их анализ в процессоре и переход на новый опрос объекта. *ЦПр*, *ПрГУ* и *ВП* можно пронумеровать цифрами $i = 1, 2, 3$. Маршрут первой задачи охватывает эти узлы. Маршрут задачи контроля охватывает два узла — *ЦПр* и *ВП*. Коэффициенты посещения a_{lr} для первого и второго потока задач $a_{11} = a_{31} = 10$; $a_{21} = 1$; $a_{12} = a_{32} = 1$; $a_{22} = 0$. Время обслуживания запроса в *ВП* не зависит от его типа и равно $v_{32} = 0,09$ с. Другие времена: $v_{11} = 0,01$, $v_{12} = 1$, $v_{22} = 0$.

Рассмотрим случай, когда проектировщик может менять производительность *ПрГУ*, полагая, что с увеличением числа объектов управления время обслуживания процессором группового управления изменяется по следующему закону: $v_{12} = 30/K_1$. Моделирование выполняется для оценки влияния числа управляемых объектов в группе на коэффициент использования аппаратуры и временной режим управления. Необходимо изучить влияние производительности *ЦПр* на характеристики процессов управления и контроля. На рис. 36, а показан график изменения загрузки p_i устройств системы в зависимости от числа задач управления K_y , и фиксированном числе задач контроля $K_k = 10$.

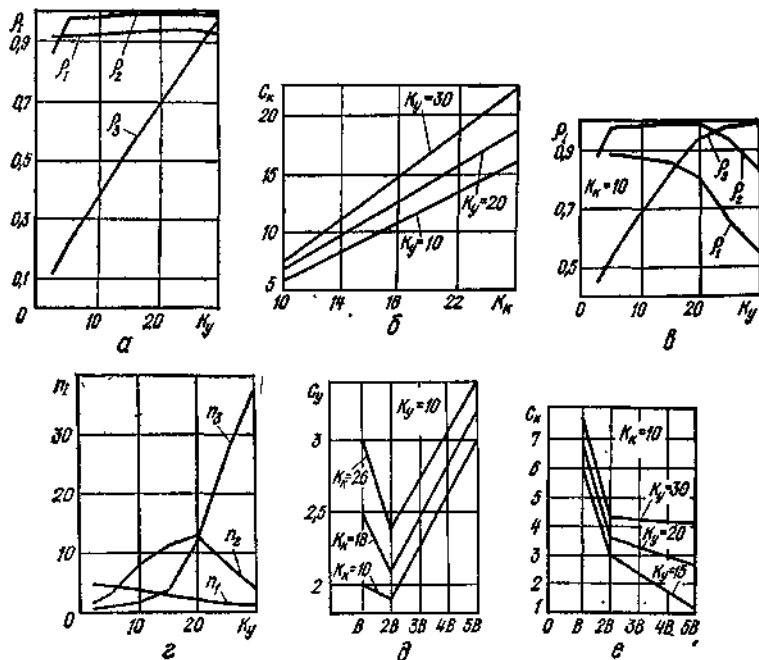


Рис. 36. Характеристики УВС, выполняющей задачи контроля и управления

Его анализ свидетельствует о том, что в этих условиях уже при малом числе задач управления высоко загруженными являются ЦПр и ПрГУ. Загрузка внешнего запоминающего устройства приближается к единице лишь при $K_y = 30$. Хотя коэффициент использования ЦПр велик при всех значениях K_y , взятых в диапазоне от 2 до 30, анализ показал, что только 10 % времени он тратит на обслуживание задач управления, а остальное время занимается, главным образом, решением задач контроля. Зависимость времени цикла задач контроля C_k от числа задач контроля и управления изображена на рис. 36, б. С помощью аналитического моделирования изучено влияние производительности процессора на характеристики системы. Из сравнения рис. 36, а и 36, в видно, что в результате замены ЦПр, имевшего производительность B , процессором производительностью 5 B его загрузка падает, а загрузка ВП возрастает. При числе задач управления K_y , из-

меняющемся от 1 до 20, узким местом ОУВС является *ПрГУ*, коэффициент использования которого близок к 1. Однако при дальнейшем увеличении K_y , узкое место смещается, и наиболее загруженным устройством становится *ВП*, в которой для обслуживания скапливается очередь значительной длины (см. график зависимости n_3 от K_y на рис. 36, г). На графиках рис. 36, д и 36, е показано влияние производительности процессора на время цикла задач управления C_y и контроля C_k . Из рис 36, е видно, что повышение производительности процессора ведет к уменьшению C_k (это естественно). На рис. 36, д увеличение производительности *ЦПр* до величины $2B$ уменьшает время цикла задач управления, однако пятикратное увеличение производительности процессора, вопреки ожиданию, приводит не к уменьшению, а к значительному увеличению времени C_y . Этот эффект обусловлен смещением узких мест в системе, задерживанием задачами одного потока задач другого. В данном случае уменьшение цикла задач контроля (что видно из рис. 36, е) вызвало их преимущественное накопление в очереди к внешней памяти, создавая условия для задержки здесь же задач управления.

Приведенный пример иллюстрирует ситуацию, которая *трудно, или вообще не предсказуема без расчетов*. Располагая подобными расчетами, можно установить оптимальное соотношение между числом задач разных типов, обеспечивая соответствующий временной режим, эффективную эксплуатацию системы, сбалансированность характеристик аппаратуры и процессов в ОУВС. Результаты таких расчетов могут служить важным ориентиром и обоснованием выбора средств системы и организации их использования.

Приведем решение еще одного примера. На рис. 37, а изображена модель ОУВС, решающей инициативные задачи, вызываемые по запросу от объекта, и периодические, трасса которых проходит через *ЦПр* и один из четырех *ПрГУ* с вероятностью $p_{1j} = 0,25$; $j = 2,3,4,5$.

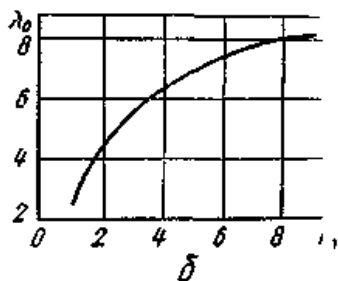
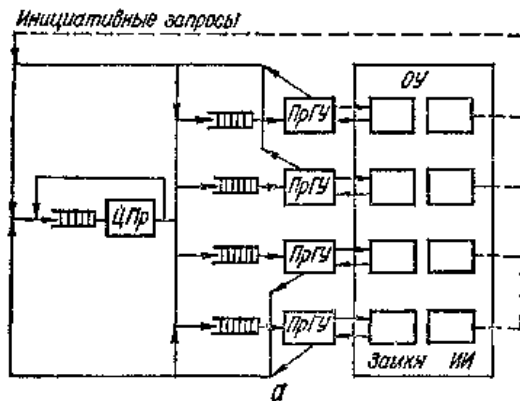


Рис. 37. Система управления, представленная смешанной сетью (а) и график изменения пропускной способности для задач замкнутой цепи управления в зависимости от их числа (б)

Исходные данные для расчета: коэффициенты посещения для разомкнутой цепи $\alpha_1 = 1$, $\alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = 0,25$; интенсивность потока запросов от инициативных датчиков — $\lambda_0 = 0,05$; коэффициенты посещения для процессов замкнутой цепи $\alpha_1 = 10$;

$\alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = 2,25$; время обслуживания запросов не зависит от цепи — $v = 0,002$; $v_2 = v_3 = v_4 = v_5 = 0,04$. Максимальное число замкнутых контуров управления объектом, представленных замкнутой цепью запросов, $K_3 = 9$. Расчет данной системы выполняется на основе смешанной сетевой модели. Рассмотрим некоторые из результатов Пропускная способность ОУВС для периодических задач показана на рис. 37, б, из которого видно, что пропускная способность ОУВС ограничена некоторым значением, которое практически достигается при реализации десяти процессов управления. Узкое место данной ОУВС — процессоры группового управления, загрузка которых при $K_3 = 9$ достигает величины $\rho = 0,75$. Время реакции на инициативный запрос при этом достигает величины $C_p = 0,72$ с и время цикла периодических задач — $C_3 = 1,086$ с.

Таким образом, применение аналитического моделирования позволяет получать сажные *результаты*. Расчеты позволяют: обнаруживать узкие места в аппаратных и программных ресурсах ОУВС; подбирать наилучшие смеси работ для данной сети ресурсов ОУВС; регулировать вычислительную нагрузку и степень мультипрограммирования; анализировать взаимное влияние различных работ ОУВС; исследовать влияние на критерии качества системы таких параметров как производительность устройств, число и трудоемкость работ по управлению объектом. Некоторые из получаемых при расчетах результатов носят *труднопредсказуемый характер*, что видно из примеров. Итоги анализа должны использоваться *для выбора и модификации аппаратных средств, а также для настройки управляющих средств, операционной системы* на зффективный режим организации облачных управляющих и вычислительных процессов. Для ускорения проектных работ в САПР кроме средств расчета должны использоваться **средства-советчики**, которые до выполнения проектного анализа или оптимизации представляют проектанту возможные направления поиска и вероятные последствия от применения того или иного шага в проектных решениях. Средства-советчики обеспечиваются **диалоговой поддержкой**, а **выдаваемые советы оформляются в виде графической и текстовой информации**. Разумеется, что в силу сложности ОУВС как объекта системного проектирования такие советы на все случаи подготовить сложно. Однако понимание даже простых случаев анализа позволяет обучить и повысить производительность труда проектанта, использующего САПР. Основой для разработки советчика САПР являются предварительные аналитические и численные исследования определенных классов систем.

Опишем, например, результаты математического анализа уравнений метода анализа средних, которые позволяют сформировать советующие средства САПР, используемые при решении перечислявшихся выше проектных задач. Заметим, что уравнения метода анализа средних, в отличие от алгоритмов свертки и локального баланса, дают явные зависимости характеристик системы от степени мультипрограммирования K , трудоемкости обслуживания в узле j Θ_j , производительности аппаратуры j -го ресурса ОУВС B_j и т. д. При математическом анализе использован аппарат теории рекуррентных уравнений. Анализируемые рекуррентные уравнения метода анализа средних нелинейны. Параметрический анализ выполняется как по дискретным аргументам (K), так и по непрерывным (Θ_j , B_j и т. д.). Математический анализ мультипрограммного режима ОУВС с K однородными пользователями, модель которого представлена уравнениями метода анализа средних, показывает следующее:

1. Пропускная способность системы $\lambda_o(K)$ — ограниченная возрастающая функция K , причем

$$\lim_{K \rightarrow \infty} \lambda_o(K) < 1/\max_i (\alpha_i \vartheta_i).$$

2. Загрузка i -го прибора не превышает величины

$$\rho_i^* = \alpha_i \vartheta_i / \max_i (\alpha_i \vartheta_i),$$

а прибор s , имеющий загрузку $\rho_s^* = 1$, есть узкое место системы.

3. Доля заявок $a_i = n_i/K$, сосредоточенных в i -м узле сети, пропорциональна трудоемкости работ этого узла $\alpha_i V_i(K)$, а соотношение между долями зависит от K . В сбалансированной системе $a_i(K) = \text{const}$. Оценку предельных характеристик системы для режима большой загрузки (при достаточно большом числе процессов K) выполняют путем решения системы алгебраических уравнений

$$a_i = \alpha_i \vartheta_i / \sum_i \alpha_i \vartheta_i,$$

откуда находят предельные величины $n_i = a_i/K$, определяющие распределение достаточно большого числа процессов K по обслуживающим узлам системы.

4. Влияние производительности j -го устройства B_j и трудоемкости работ на этом устройстве Θ_j на характеристики системы представлено зависимостями на рис. 38, а и б.

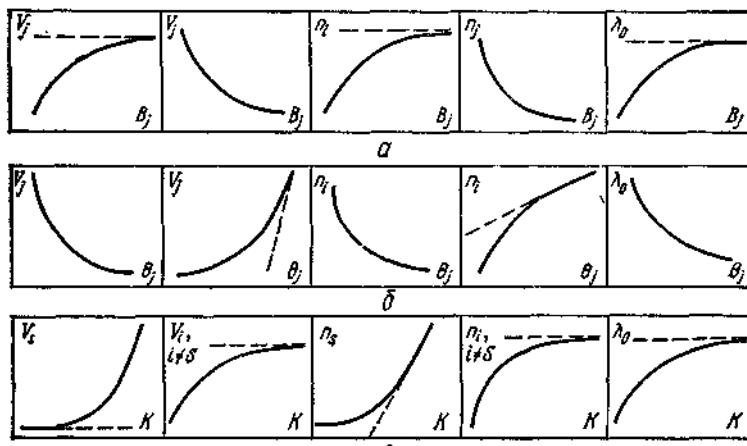


Рис. 38. Графическая информация советчика САПР

Графики соответствующим образом интерпретируются и комментируются текстовой информацией советчика системы автоматизированного проектирования.

5. Характер зависимостей $V_i(K)$, $n_i(K)$ и $\lambda_0(K)$, представляемых графической информацией советчика САПР, дан на рис. 38, в.

Анализ зависимостей $V_s(K)$ и $n_s(K)$, например, указывает, что при возрастании числа процессов K в системе заявки скапливаются в ее узком месте s . Могут быть подробно прокомментированы и другие особенности представленных графиками зависимостей. Итак, предаваемая графическая информация советчика САПР сопровождается текстовыми комментариями и рекомендациями для проектанта.

Математический анализ уравнений метода анализа средних для мультипрограммного режима ОУВС с двумя однородными потоками задач, число которых равно соответственно K_1 и K_2 , также позволяет сформировать исходную информацию для разработки советчика САПР. Пусть $r = 1$ — номер цепи, в которой происходит варьирование числа процессов K_1 , а число заявок второго потока $K_2 = \text{const}$. Тогда:

1. Пропускная способность $\lambda_1(K_1)$ цепи, в которой происходит увеличение числа заявок K_1 , возрастающая функция, ограниченная некоторой величиной λ_1 , а пропускная способность $\lambda_2(K_2)$ для заявок второй цепи — убывающая к нулю функция.

2. Пусть $n_{i1}(K_1)$ — число заявок типа 1 в узле i при заданном K_1 .

Увеличение числа K_1 , ведет к тому, что для некоторых обслуживающих узлов $n_{i1}(K_1)$ — возрастающая выпуклая

(неограниченная) функция, в других узлах функция $n_{i1}(K_1)$ — возрастающая и ограниченная, и, наконец, могут существовать узлы, где $n_{i1}(K_1)$ имеет один экстремум (максимум).

3. Пусть $n_{i2}(K_1)$ — функции, описывающие изменение числа заявок цепи 2 в узлах i при $K_2 = \text{const}$ и $K_1 = \text{var}$. Существуют обслуживающие узлы, где $n_{i2}(K_1)$ — ограниченная возрастающая функция, и существуют узлы i , где она монотонно убывает к нулю.

4. Характер зависимостей для времен пребывания $V_n(K_x)$ и $V_{l2}(K_x)$ совпадает с характером зависимостей для числа заявок в узлах.

5. Поскольку $\lambda_1(K_1)$ и $\lambda_2(K_1)$, соответственно, возрастающая и убывающая функции, максимум пропускной способности системы, определяемой как сумма $\Lambda = \lambda_1 + \lambda_2$, может быть в любой точке области изменения K_1 . Заметим, что в ОУВС с одним потоком задач $\lambda_2 = 0$, и максимум пропускной способности лежит на бесконечности по K_1 .

6. Можно доказать, что в отличие от однопотокового случая, максимум коэффициента использования ресурсов системы

$$R = \sum_i p_i$$

может располагаться как на границе, так и внутри области изменения показателя мультипрограммирования $K = (K_1, K_2)$. Выполненные исследования позволяют утверждать, что увеличение показателя мультипрограммирования K может снижать как эффективность использования, так и производительность системы, но предсказать это явление априорно не представляется возможным.

7. Изменение производительности одного из устройств системы качественно так же влияет на изменение характеристик мультипрограммной системы, как и в случае одного потока задач (рис. 38, а).

Представленные выше результаты обобщаемы на многозадачные (многопотоковые) режимы мультипрограммирования.

9.3.2. Агрегирование структур и слияние потоков — основа для ускорения проектных расчетов

Декомпозиция предполагает выделение в системе частей (блоков, агрегатов), которые используются при ее расчете как некоторые первичные элементы. Например, обслуживающий прибор с очередью может быть эквивалентом сети с очередями. В этом случае функция, определяющая время пребывания заявки в приборе, должна совпадать с функцией, определяющей время ответа эквивалентной ей подсети.

Декомпозиция сетевых моделей ВС дает вычислительные преимущества, обеспечивая исследование систем большой сложности. Цель метода эквивалентирования — представление подсети очередей единственным узлом, характеристики которого эквивалентны макрохарактеристикам этой подсети. В подсети может обслуживаться несколько заявок одновременно. В эквивалентной очереди — только одна заявка. Поэтому время обслуживания в эквивалентной очереди должно зависеть от количества заявок в ней, т. е. от нагрузки. Для некоторых подсистем эквиваленты можно получить путем имитационного моделирования подсети, применением регрессионного анализа. Для стохастических сетей с решением в виде произведения можно применять рассматриваемый здесь метод эквивалентных замен. В более сложных случаях эквивалентирование может выполняться путем моделирования процессов в подсети как марковских процессов с дискретными состояниями и т. п. Марковская модель подсети может решаться численными или аналитическими (рекурсивными) методами, давая значения вероятностей состояний и точные эквивалентные характеристики очереди, заменяющей подсеть.

Подход к эквивалентированию сетей с очередями аналогичен методам, применяемым в теории электрических цепей. Если цепь состоит из источников и резисторов и необходимо исследовать режим одного из элементов электрической цепи при варьировании его параметров, то используют *метод эквивалентного генератора*. Заменяв все элементы цепи, за исключением данного, набором из двух компонентов — источника и резистора, — строят простую цепь и обеспечивают ускорение расчетов.

Рассмотрим замкнутую сеть с N очередями, в которой движется K процессов (рис. 39, а).

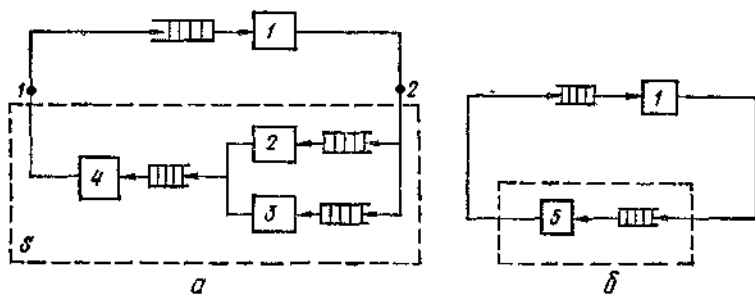


Рис. 39. К объяснению эквивалентных преобразований сетей: а — исходная, б — эквивалентная ей упрощенная сеть

Необходимо построить эквивалентную сеть, в которой подсеть S заменена одним обслуживающим узлом (рис. 39, б). Метод эквивалентных замен состоит в том, что производится замыкание входа и выхода эквивалентизируемого участка сети и определяется его пропускная способность $\lambda(K)$. Пропускная способность — величина, обратная времени цикла заявки относительно точки замыкания. Она зависит от числа заявок в подсети.

Заменяв подсеть S единственным узлом-агрегатом, время обслуживания которого $\nu(K) = 1/\lambda(K)$, получим простую сеть из двух очередей. Такое преобразование порождает сеть, все свойства которой по отношению к выделенному элементу сохраняются такими же, каковы они в исходной сети. Однако анализ ее благодаря малой сложности значительно облегчается. Процедура определения пропускной способности короткозамкнутой подсети S при автоматизированных расчетах в САПР реализуется путем задания нулевого времени обслуживания всем узлам сети, не включенным в S . Подобные эквивалентные преобразования возможны также для разомкнутых и смешанных сетей. В разомкнутых сетях агрегируемую подсистему заменяют новым источником заявок с интенсивностью λ . Величина λ равна пропускной способности исключаемой подсистемы S . При декомпозиции многоцепных сетей с очередями эквиваленты подсетей определяют аналогично одноцепному случаю, соединением выхода каждой цепи с ее входом в точке замыкания подсети. Рассмотрим, например, замкнутую сеть, изображенную на рис. 39, и выполним показанное на нем эквивалентное преобразование. Здесь узлы 2, 3 и 4 заменяются одним узлом 5. Скорость обслуживания в узле 5 должна быть равна скорости, с которой заявки проходят между точками 1 и 2, когда они замкнуты. Пусть число заявок в сети равно трем; времена обслуживания $\vartheta_1 = 1; \vartheta_2 = 2; \vartheta_3 = 2; \vartheta_4 = 1,5$; вероятности перехода заявок на разветвлении $p_{12} = p_{13} = 0,5$. Для определения коэффициентов посещения α_i ($i = 1, 4$) составим уравнения баланса потоков: $\lambda_1 = \lambda_4$;

$$\lambda_2 = p_{12}\lambda_1; \lambda_3 = p_{13}\lambda_1; \lambda_4 = \lambda_2 + \lambda_3.$$

Учитывая, что $\alpha_i = \lambda_i/\lambda_1$ получаем $\alpha_1 = \alpha_4 = 1; \alpha_2 = \alpha_3 = 0,5$. Рассчитаем пропускную способность закороченной между точками 1 и 2 подсети S методом анализа средних, в результате $\lambda(1) = 0,46; \lambda(2) = 0,69; \lambda(3) = 0,82$. Время обслуживания в узле 5, эквивалентном агрегату S , — величина, обратная пропускной способности закороченной подсети. Узел-эквивалент 5 имеет переменную скорость обслуживания, зависящую от числа заявок, находящихся в нем: $\nu_5(1) = 0,86$;

$v_6(2) = 1,21$; $v_5(3) = 1,37$. Таким образом, для исследования свойств сети при изменении параметров узла 1 можно использовать не исходную сложную сеть, а агрегированную простую.

При параметрическом анализе, основанном на агрегировании, сложную систему можно разбить на агрегаты, каждый из которых представляется единственным узлом с переменной скоростью обслуживания. Для определения скорости замыкают обратной связью выход и вход агрегата и рассчитывают пропускную способность этой части сложной системы по отношению к цепи обратной связи для различного числа заявок в агрегате.

Для упрощения сетей применяют также метод *объединения цепей и классов процессов*. Вначале рассмотрим замещение процессов, представленных многоклассовыми цепями, одноклассовыми, а затем — построение приближенного представления многоцепной сети в виде одноцепной. Многоклассовым сетям свойственна высокая изобразительная способность. Изменение классов используется для простого описания точной последовательности прохождения проблемных программ и их взаимодействия с системными. При этом рассматриваются классы системных и проблемных (пользовательских) действий. Если предположить, что за каждым пользовательским запросом следует системный запрос, и присоединить их время к соответствующим элементам рабочей нагрузки пользователей, то можно сделать первый шаг в упрощении структуры процессов. Однако если длина пользовательских действий разных классов различна, то необходимо сделать еще один шаг упрощений, заменив L классов процессов одним.

Преобразование множества из L классов обслуживания в один выполняется на основе непосредственного использования исходных данных о процессах, и представления информации о трассе программы в виде марковской цепи. Каждая вершина цепи помечается парой (i, l) , обозначающей, что работа класса l выполняется в узле i . Цель преобразований состоит в построении эквивалентной марковской цепи, где все состояния с одинаковым значением i заменяются одним состоянием.

Пусть v_{il} — время обслуживания заявок класса l в узле i , α_{il} — соответствующие коэффициенты посещения; p_{il} — вероятность того, что пришедшая в узел заявка требует l -го типа обслуживания. Введем среднее время

$$\phi_i = \sum_l p_{il} \phi_{il} = \frac{1}{\alpha_i} \sum_l \alpha_{il} \phi_{il} \quad (3.1)$$

и коэффициент посещения

$$\alpha_i = \sum_l \alpha_{il}.$$

Вероятности p_{il} , используемые в выражении (3.1), получаются из следующих условий:

$$p_{il} = \frac{\lambda_{il}(K)}{\lambda_i(K)} = \frac{\lambda_{il}(K)}{\sum_l \lambda_{il}(K)} = \frac{\alpha_{il} \lambda_0(K)}{\sum_l \alpha_{il} \lambda_0(K)} = \frac{\alpha_{il}}{\alpha_i}.$$

Отсюда видно, что вероятность p_{il} не зависит от K .

Используя формулу (3.1), можно получить следующие зависимости для расчета неоднородных (многоклассовых) сетей на основе метода анализа средних:

$$n_{il}(0) = 0;$$

$$V_{il}(K) = \vartheta_{il} + \delta_i \sum_l \vartheta_{il} n_{il}(K-1);$$

$$\lambda_0(K) = K / \left(\sum_i \sum_l \alpha_{il} p_{il} V_{il}(K) \right);$$

$$n_{il}(K) = \alpha_i \lambda_0(K) V_{il}(K);$$

$$p_{il}(K) = p_{il} \alpha_i \lambda_0(K) \vartheta_{il}.$$

Аналогично усредняются времена обслуживания в многоклассовых разомкнутых сетях.

Рассмотрим подход к объединению цепей. Необходимо заменить R

$$K = \sum_r K_r,$$

цепей задач одной, с указав для нее трассовые вероятности p_{ij} или коэффициенты посещения и время обслуживания в узлах v_i . Совмещение цепей выполняется на основе следующих вычислений:

$$p_{il} = \sum_r p_{il,r} q_{il,r}; \quad \vartheta_i = \sum_r \vartheta_{il,r} q_{il,r},$$

где $q_{ij,r}$ — вероятность того, что случайно выбранный запрос на пути от узла i к узлу j сети принадлежит типу r ; $q_{i,r}$ — вероятность того, что случайно выбранный запрос, обслуженный в узле i , принадлежит цепи запросов r . Точные значения этих величин

$$q_{ij,r} = \lambda_{ij,r} / \sum_r \lambda_{ij,r}; \quad q_{i,r} = \sum_j q_{ij,r}.$$

Приближенные априорные оценки $q_{ij,r}$ можно выполнить на основе расчета:

$$q_{ij,r} = \lambda_r^* / \sum_r \lambda_r^* = \frac{K_r / C_r^*}{\sum_r (K_r / C_r^*)},$$

где C^*_r — оценка времени цикла процессов цепи r на основе полной трудоемкости задач в одном цикле их выполнения; λ^*_r — оценка пропускной способности цепи r .

Поскольку пропускная способность замкнутой цепи зависит от числа заявок в ней, при таком усреднении невозможно добиться совпадения характеристик исходной и эквивалентной сетей при всех K . Если не учитывать сказанное, то слияние нескольких потоков в один может исказить характеристики и привести к принятию неверного проектного решения. Иллюстрация объединения цепей в одну дана при изложении вопросов создания сбалансированных систем в гл. 9.3.3. Описанные методы агрегирования структур и слияния процессом являются основой для ускорения проектного анализа, обеспечивая быстрое выполнение расчетов в условиях жестких временных ограничений на интервал проектных работ.

9.3.3. Балансирование рабочей нагрузки и параметров ОУВС

Изменение рабочей нагрузки в ОУВС связано с перестройкой ГАПС. Перестройка производства меняет порядок функционирования управляющей системы и перераспределяет нагрузку на процессоры, средства обмена и ввода-вывода. Узкое место — это такой ресурс системы, который сдерживает производительность ОУВС и ГАПС в целом при заданном порядке ее функционирования.

Процедура проектирования позволяет улучшить характеристики системы за счет увеличения производительности наиболее загруженного, насыщенного устройства. В результате такого решения при дальнейшем увеличении нагрузки на систему (глубины мультипрограммной обработки ОУВС K), узким местом станет другой узел, насыщающийся при большем K . Наступает такой момент в проектировании, когда очередное узкое место нельзя устранить по техническим причинам, поскольку в ОУВС отсутствует более мощный ресурс соответствующего типа. Последовательность устранения узких мест при повышении нагрузки на систему прерывается на критическом ресурсе (устройстве или программе).

Метод последовательного устранения узких мест — это стандартный метод последовательной оптимизации. Для расчетов в реальном масштабе времени необходимо располагать другим, более быстрым методом оптимизации пропускной способности и экономичности системы, но до некоторой степени менее точным.

Разумеется, что ускорению расчетов способствует не только улучшение алгоритмов оптимизации, но и упрощение модели системы,

являющейся основой для выполнения анализа. Упрощение модели целесообразно и по другим причинам. Так, усложнение модели может вести к уменьшению точности результата из-за погрешностей численного анализа и низкой точности исходных данных. Использование упрощенных моделей, несмотря на их некоторую неточность, может давать такие отклонения рекомендуемых параметров системы, которые не сказываются на подборе варианта комплектации ОУВС. В комплекте ОУВС могут использоваться устройства ограниченной номенклатуры, параметры которых представлены дискретным рядом. Поэтому смещение оптимума в определенных пределах может и не повлиять на выбор комплектности средств из дискретного ряда устройства.

Рассмотрим подход к обеспечению *сбалансированного режима работы однопроцессорной ОУВС и объекта управления*. ОУВС используют для группового управления объектом, состоящим из K управляемых элементов (рис. 40, а).

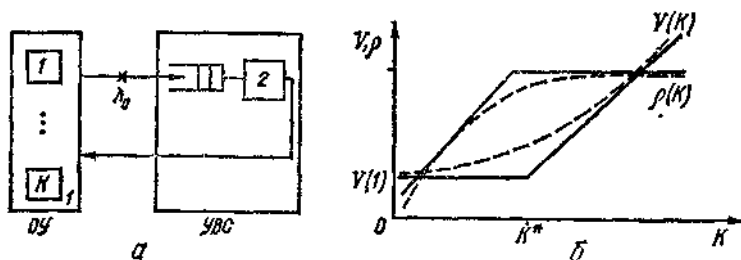


Рис. 40. Модель ОУВС группового управления и ее приближенные характеристики

Параметры ОУВС и объекта управления ОУ: время отработки управляющего воздействия элементом объекта, равное интервалу от выдачи управляющего воздействия из ОУВС до момента формирования следующего запроса — $v_1 = 1/\lambda_1$; трудоемкость выработки управления в ОУВС, выраженная числом процессорных операций — Θ ; производительность УВС — B ; время цикла управления — C ; время обслуживания в ОУВС — $v_2 = \Theta/B$ и интенсивность обслуживания $\mu = 1/v_2$.

Время реакции ОУВС при $K = 1$

$$V(1) = \vartheta_2 = 1/\mu = \Theta/B. \quad (3.2)$$

Время цикла заявки в мультипрограммном режиме не превышает величины

$$C = \vartheta_1 + V(K) < \vartheta_1 + K\vartheta_2. \quad (3.3)$$

Здесь время реакции ОУВС $V(K)$ получено из условия, что v_1 мало по сравнению с v_2 , и тогда почти все заявки цепи сосредоточены в ОУВС. Предположим, что ОУВС работает в режиме большой нагрузки. Тогда $\lambda_0 \approx \mu$, где λ_0 — суммарный поток от объекта. По формуле Литла $\lambda_0 = K/C$. Из выражения (3.3), с учетом сделанного предположения, получим

$$V(K) \approx \frac{K}{\mu} - \frac{1}{\lambda}. \quad (3.4)$$

На рис. 40, б точное значение $V(K)$ показано штриховой линией. Зависимость $V(K)$, описанная формулами (3.2) и (3.4), является приближенной, причем каждая из этих формул представляет собой асимптоту, соответствующую малой и большой нагрузкам на ОУВС. Приведем теперь решающий аргумент, определяющий подход к балансированию нагрузки и выбору производительности ОУВС. Пусть K^* — точка пересечения асимптот (3.2) и (3.4). При числе элементов объекта управления $K < K^*$ время реакции в системе мало. Однако и нагрузка $\rho(K)$ процессора далека от предельной. Поэтому экономические соображения диктуют чтобы $K > K^*$. Следовательно, $K = K^*$ — разумный компромисс для выбора K в данной конфликтной ситуации.

Укажем требования к постановке проектных работ, выполняемых на основе реализации условия сбалансированности:

1. Необходимо выбрать число элементов объекта группового управления, обслуживание которых целесообразно поручить ОУВС с фиксированным быстродействием. Дано: λ , μ и B , находится оптимальное K —
2. Необходимо выбрать процессор, обеспечивающий управление объектом в режиме сбалансированной эксплуатации. Дано: λ , Θ , K , определяется B .
3. Выбрать процессор и средства управления объектом, если необходимый цикл управления S регламентируется режимом реального времени. Дано: S — период выдачи управления; Θ_1 и Θ_2 — трудоемкость операций отработки и выработки управления соответственно; K — число объектов управления, найти оптимальную производительность отрабатывающей управление аппаратуры объекта B_1 и оптимальную производительность B_2 ОУВС, вырабатывающей управление.

Очень часто в практике встречается более общая постановка задачи проектных работ, являющаяся обобщением трех предыдущих для случая управления распределенным объектом со стороны распределенной ОУВС; причем выбору подлежит здесь лишь часть управляющих элементов объекта и средств ОУВС, поскольку

применение некоторых аппаратных ресурсов регламентируется условиями производства.

Во всех поставленных задачах система объект — ОУВС должна быть сбалансированной, чтобы режим эксплуатации обеспечивался в точке пересечения асимптот. Такой выбор гарантирует компромисс между временными параметрами и экономическими соображениями эксплуатации автоматизированного производства. Для сбалансированности объекта, создающего рабочую нагрузку, и ОУВС их параметры должны быть такими, чтобы режим эксплуатации соответствовал точке излома асимптот $V(1)$ и $V(K)$. В точке излома $V(1) = V(K^*)$, поэтому из выражений (3.2) и (3.4) получим

$$K^* = (\mu/\lambda) + 1. \quad (3.5)$$

Выражение (3.5) позволяет найти оптимальное число управляемых элементов объекта при заданных временных характеристиках обслуживания.

Если необходимо решить задачу подбора ОУВС при заданном числе элементов объекта K (постановка 2), то оптимальная производительность ОУВС, обеспечивающая условие сбалансированности, находится из выражения

$$B = (K - 1)\lambda\theta, \quad (3.6)$$

которое следует из уравнения (3.5), если $K = K^*$.

Рассмотрим подход к выбору числа управляемых объектов и производительности средств ОУВС, если ОУВС представлена в виде *распределенной системы*, состоящей из нескольких устройств. Пусть ОУ, помеченный числом $i = 1$ на рис. 40, *а*, содержит K управляемых элементов, а ОУВС включает устройства $i = 2, 3, \dots, N$, участвующие в выработке управления. Обозначим: $t_i = \alpha_i v_i$ — трудоемкость работ в цикле управления на устройстве i ; s — номер устройства ОУВС, являющегося ее узким местом.

Если $K = 1$, то аналогично выражению (3.2) получим

$$V(1) = \sum_{i=1}^N t_i, \quad i = 1, 2, \dots, N. \quad \text{Подобно уравнению (3.4)}$$

$V(K) = K t_s - 1/\lambda$; $t_s = \alpha_s v_s$. В сбалансированной системе все $p_i = r$.

Учитывая, что $p_i = \alpha_i v_i \lambda_0$, в сбалансированной системе на всех устройствах $t_i = t$. Поэтому $V(1) = t(N - 1)$. С учетом этого, приравняв $V(K) = V(1)$, получим

$$K^* = [(N - 1)t + \theta_1]/t, \quad (3.7)$$

определяющее оптимальное число управляемых объектов. Для решения проектной задачи выбора производительностей устройств ОУВС B_i воспользуемся выражением (3.7), взяв $K^* = K$. Отсюда

находят $t = v_i/(K - N + 1) = t_i$ и оптимальные B_i , следовательно, берутся равными

$$B_i = \alpha_i \Theta_i / t, \quad (3.8)$$

где Θ_i — трудоемкость работ в устройстве i , выраженная числом операций в секунду.

Выравнивание времен обслуживания так, чтобы $t_i = t = \text{const}$ для всех узлов, приводит к равномерному распределению процессов по обслуживающим узлам. В противном случае перегружаются одни узлы и недогружаются другие, имея избыток производительности.

Если устройства системы состоят из многоканальных приборов, число которых в узле равно m , то рассмотренные формулы модифицируются. Так, аналогично выражениям (3.2) — (3.6) получим

$$V(1) = 1/(m\mu); \quad V(K) = \frac{K}{m\mu} - 1;$$

$$K^* = \frac{m\mu}{\lambda} + 1; \quad B = (K - 1)\lambda\Theta/m,$$

где полагается, что интенсивность обслуживания с помощью многоканального устройства возрастает в m раз по сравнению с одноканальным.

Необходимо заметить, что точка равенства ρ и точка максимума пропускной способности могут не совпадать, но все же они близки. Выводы о совпадении этих точек сделаны по приближенным оценкам, когда сложные зависимости аппроксимировались двумя прямыми. Формулы (3.5) — (3.9) содержат лишь несколько арифметических операций, но позволяют быстро и с достаточной инженерной точностью выбрать комплект ОУВС для заданного режима эксплуатации объекта. Расчет K^* и B выполняется, соответственно, при первых двух постановках задачи балансирования, перечислявшихся ранее.

Рассмотрим более сложный случай, когда управление объектом выполняет распределенная ОУВС, имеющая несколько процессорных и других средств, производительность которых B_i следует определить. Дано: число различных типов задач системы — R ; коэффициенты посещения α_{ir} устройств i задачами типа r ; средние трудоемкости работ на устройствах Θ_{ir} ; требуемое среднее время цикла прохождения задач управления (средняя периодичность управления) S . Известно, что локальная система управления объектом имеет производительность B , и при однократном обращении должна выполнить Θ операций группового управления. Найти производительности устройств управляющей системы, которые обеспечивают необходимую среднюю периодичность управления S и эффективное использование мощности ресурсов.

Методика расчета объединяет следующие действия: 1) переход от сложной многопоточковой модели, представленной исходными данными, к однопоточковой (объединение потоков выполняется на основе методов, изложенных в гл. 9.3.2); 2) интерпретацию однопоточковой модели простыми арифметическими выражениями, обеспечивающими расчет искомых производительностей B_i ;

3) применение однопоточковой модели процессов для оценки среднего времени цикла управления C ; 4) проверочный расчет на основе многопоточковой модели, позволяющий дифференцировать источники нагрузки и время цикла для управляющих процессов каждого из r типов (выполняется при необходимости); 5) подбор конфигурации и компонентов ОУВС (на основе рассчитанных B_i). Если результат расчетов C неудовлетворителен, то необходимо выполнить одно или несколько следующих действий: а) уменьшить число объектов группового управления, обслуживаемых ОУВС; б) изменить конфигурацию системы, создавая условия для повышения ее производительности; в) изменить алгоритм управления объектом; г) принять меры к углублению распределенной обработки на основе распараллеливания вычислительных процессов.

Замещение r -поточковой системы однопоточковой с $K=R$ процессами (первый шаг алгоритма) выполняют путем усреднения коэффициентов посещения и трудоемкостей работ Θ_{ir} в каждом из узлов i . Так, коэффициент посещения устройства i средней задачей находим таким образом:

$$\alpha_i = \frac{1}{R} \sum_r (\alpha_{ir} / \beta_r),$$

где $\beta_r = T_r / T$; $T_r = \sum_i \alpha_{ir} \Theta_{ir}$; $T = \sum_r T_r / R$; T_r — трудоемкость средней задачи. Аналогично усредняют Θ_{ir} и находится Θ_i .

Интерпретацию полученной однопоточковой модели и расчет оптимальных B_i (второй шаг алгоритма) выполняют по формуле

$$B_i = \alpha_i \Theta_i / (\alpha_i \Phi_i), \quad (3.10)$$

где j — узел, представляющий управляемый элемент ОУВС; v_j — время, затрачиваемое на отработку управляющего воздействия.

Выражение (3.10) получено на основании следующих соображений. Необходимо, чтобы загрузка всех устройств была максимальной, т. е. $\rho_i = \alpha_i \Phi_i \lambda_0(K) = \alpha_i \Theta_i \lambda_0(K) / B_i = 1$.

В этом выражении представлена также загрузка управляющих устройств j : $\rho_j = \Phi_j \lambda_0(K) = 1$. Из отношения ρ_i к ρ_j можно получить выражение (3.10). Итак, второй шаг алгоритма расчета, так же как и первый, выполняется по элементарной формуле. Третий и

четвертый шаг алгоритма выполняются с использованием программного обеспечения САПР и являются проверочными, за которыми следует подбор комплекта ОУВС, наиболее приближенного к оптимальному (пятый шаг алгоритма).

Приведем пример. Пусть число объектов управления $R = 4$, трассы соответствующих им процессов определены коэффициентами посещения устройств $i=1, 2, 3$ и даны в табл. 1.

Таблица 1

$i \backslash j$	1	2	3	4
1	8	11	11	16
2	2	7	6	10
3	5	3	4	5

Трудоемкости работ на устройствах ОУВС: $\Theta_1 = 18720$; $\Theta_2 = 1,56$; $\Theta_3 = 1,3$. Они одинаковы для каждой из цепей. Производительность устройства группового управления локальной системы управления $B_2 = 15 \text{ с}^{-1}$. Выполнив расчеты, получим характеристики средней задачи: $\alpha_1 = 11,5$; $\alpha_2 = 5,91$; $\alpha_3 = 4,52$. Оптимальные производительности устройств $B_1 = 350\,000$; $B_3 = 9,56 \text{ с}^{-1}$. Таков результат решения задачи.

Предполагаемая загрузка устройств $\rho_i(4) = 1$. Средние длительности обслуживания, полученные, исходя из выбранных B , $v_1 = 0,0535$; $v_2 = 0,104$; $v_3 = 0,136$. Сделаем проверку, применив формулы методов теории сетей с очередями к расчету характеристик сети со «средней» задачей. Соответствующие результаты приведены в табл. 2.

Таблица 2

r	V_{1r}	V_{2r}	V_{3r}	λ_{0r}	ρ_{1r}	ρ_{2r}	ρ_{3r}	π_{1r}	π_{2r}	π_{3r}
Характеристики, рассчитанные на многоцепной модели										
1	0,13	0,29	0,29	0,33	0,17	0,08	0,27	0,34	0,19	0,47
2	0,13	0,27	0,33	0,24	0,17	0,21	0,21	0,34	0,44	0,23
3	0,15	0,26	0,32	0,23	0,17	0,17	0,15	0,36	0,2	0,29
4	0,15	0,22	0,33	0,16	0,17	0,2	0,13	0,39	0,35	0,26
Усредненные и просуммированные по множеству потоков $K = \overline{1, 4}$ характеристики										
	0,14	0,26	0,32	0,96	0,67	0,66	0,66	1,43	1,18	1,15
Результаты расчетов на модели со «средней» задачей										
	0,13	0,25	0,33	0,89	0,67	0,67	0,67	1,33	1,34	1,34

Здесь же даны результаты расчетов на исходной многопотоковой сетевой модели при производительностях устройств V_i , вычисленных выше.

Из проверочного расчета видно, что суммарная загрузка устройств ОУВС одинакова и равна $\rho_i \approx 0,67$, причем эти цифры получены как для вычислительно сложной многопотоковой модели, так и для более простой однопотоковой модели, полученной путем слияния потоков. Приближенная формула (3.10) выведена из условий, что будет обеспечена загрузка $\rho_i = 1$. Получаемый при любых вариантах запас производительности устройств должен быть гарантией для обеспечения работоспособности ОУВС. Анализируя результаты, сведенные в табл. 2, можно видеть, что при вычислениях среднего времени цикла, сделанных на различных по сложности моделях, погрешности не превышают 8%. Заканчивая анализ результатов расчета, отметим, что найденные значения V_i обеспечивают наиболее эффективную сбалансированную загрузку средств ОУВС при среднем времени цикла задач управления, равном $C = K/\lambda_0 \approx 4,4$ с.

В приведенном примере рассмотрены одновременно два аспекта проектных расчетов — слияние потоков, излагавшееся в гл. 9.3.2, и методика оптимизации ОУВС на основе балансирования рабочей нагрузки с аппаратными возможностями ОУВС.

На основе вычисленных V_i подбираются или комплектуются устройства соответствующего типа. Таким образом, выполнение задачи проектирования на основе двух этапов упрощения модели —

слияния потоков и последующего применения простых приближенных формул позволяет с достаточной инженерной точностью решить задачу выбора параметров, обеспечивающих эффективную эксплуатацию и необходимый временной режим ОУВС. Время таких расчетов, поддерживаемое соответствующими средствами САПР, приемлемо для проведения оперативных проектных разработок.

9.3.4. Анализ и оптимизация ОУВС с общими шинами, динамической памятью и другими пассивными или одновременно захватываемыми ресурсами

В практических расчетах необходимо учитывать такие существенные свойства процессов и систем, как многообразие дисциплин обслуживания процессов; присутствие в системах пассивных ресурсов, блокировки по условиям, семафорные механизмы; параллельная обработка ветвей одного процесса; разнообразие свойств законов обслуживания в активных элементах ОУВС. Перечисленные характеристики — лишь часть из тех, которые затрудняют всеобъемлющий учет свойств процессов. В связи с возникшими трудностями предложены различные итерационные и приближенные подходы к исследованию ВС.

Для выполнения определенных действий процессу может потребоваться одновременно несколько ресурсов. Например, при загрузке детали на станок с помощью робота деталь «захватывает» на определенное время действие двух ресурсов — станка и робота. После загрузки деталь «отпускает» робота, но по-прежнему захватывает обрабатывающее устройство — станок. Одновременный захват ресурсов процессом часто встречается в облачных управляющих системах. Так, при передаче данных по каналу связи между двумя устройства осуществляется совместная работа трех ресурсов — передатчика, приемника и шины. Здесь ресурсы, часть из которых считается активными, а часть — пассивными, объединяются для выполнения совместных действий. Тот из двух ресурсов $R1$ и $R2$, который выделяется по запросу раньше другого, называют первичным. Рассмотрим основные подходы к расчету *одновременного захвата ресурсов*.

1. В модели отражается только первичный $R1$ или только вторичный ресурс $R2$, если временем их совместной работы или задержкой в одном из них можно пренебречь.
2. При расчете системы с одновременным использованием ресурсов предполагается, что первичный ресурс является пассивным. Индивидуальную производительность вторичного ресурса $R2$ рассчитывают для каждого числа заявок, после чего учитывают

ограничение, накладываемое ресурсом $R1$ на производительность $R2$, и таким образом их пара представляется единым обслуживающим узлом-агрегатом. Этот подход не является универсальным и применяется, если первичный ресурс многоприборный, т. е. состоит из ряда ресурсных единиц, причем существует соответствие между числом выделенных единиц $R1$ и числом заявок в $R2$. Данный подход можно применить, например, к расчету схемы на рис. 8, в которой $R1$ — множество разделов основной памяти; $R2$ — процессор.

3. Система представляется в виде комплекса взаимодействующих моделей $M1$ и $M2$, причем в модели $M1$ точно отражается индивидуальная работа первого ресурса, а в модели $M2$ — второго (узлы $R1_0$ и $R2_0$ на рис. 41).

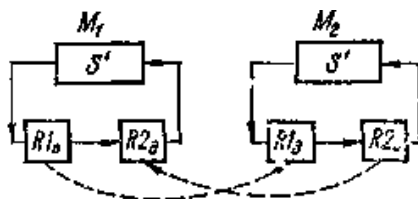


Рис. 41. Схема взаимодействия подмоделей в модели одновременного захвата ресурсов

Для учета совместной работы и взаимодействия ресурсов в модели $M1$ и $M2$ вводятся дополнительные задержки $R1_d$ и $R2_d$, которые совместно с основными узлами $R1_0$ и $R2_0$, соответственно, должны представлять ресурсы $R1$ и $R2$. Априорно известны только параметры индивидуальной работы ресурсов, т. е. узлов $R1_0$ и $R2_0$. Начальное приближение параметров $R1_d$ и $R2_d$ определяется из исходных данных, либо полагается нулевым. Итерационный охват расчетов моделей $M1$ и $M2$ позволяет уточнять параметры дополнительных задержек в узлах $R1_d$ и $R2_d$ путем обмена данными по схеме, указанной на рис. 41. Характеристики $R1_d$ уточняются из расчета $M1$, а $R2_d$ — из расчета $M2$. Если итерационный процесс сходится, то он завершается тем, что обе модели дают одинаковые характеристики, т. е. модели $M1$ и $M2$ становятся эквивалентными. Если число одновременно используемых ресурсов больше двух, то соответственно увеличивается число моделей, охватываемых итерационными уточнениями.

Перечисленные три подхода к расчету одновременного захвата ресурсов могут использоваться комбинированно. Рассмотрим реализацию способов расчета для конкретных случаев.

Анализ и оптимизация структур распределенных ОУВС, состоящих из ЭВМ с общей шиной. ОУВС (рис. 42) состоит из центральной подсистемы ЦП, периферийных подсистем ПП_{*i*}, $i = 1, \dots, n$, и адаптеров дистанционной связи АДС.

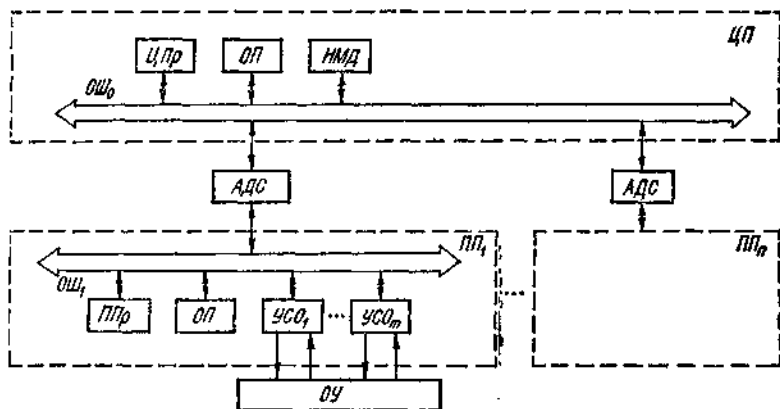


Рис. 42. Структура распределенной ОУВС, комплексируемой из ЭВМ с общей шиной

Число управляемых элементов объекта равно K . Таким образом, каждая из ПП выполняет управление группой из $m = K/n$ элементов объекта управления ОУ. Назначение ПП — обеспечить взаимодействие с ОУ (выдачу информации или накопление данных в основной памяти ОП) и первичную обработку информации средствами устройств связи с объектом УСО и периферийного процессора ППр. Назначение ЦП — обработка информации с ОУ, выработка управляющих воздействий для запуска ПП. АДС служит для связи ЦП и ПП.

Использование нескольких ПП позволяет распараллелить обработку информации, совмещая эти операции в центральном и периферийных процессорах ЦПр и ППр. Маршрут задачи в управляющей системе представлен следующей трассой: обработка в ЦП, запуск группы УСО_{*i*}, автономная работа УСО (управление объектом и ввод данных), первичная обработка информации в ППр, передача информации из ПП_{*j*} в ЦП. Описанный маршрут повторяется циклически.

Операцией спуска УСО_{*ij*} можно пренебречь, включая время на запуск и общее время автономной работы устройств связи с объектом.

При передаче данных из ПП в ЦП происходит захват общих шин ОШ_{*i*} и ОШ₀. Производительность ЦПр зависит от загрузки $\rho_{ш0}$ шины ЦП.

ЦПр выполняет проблемные, системные задачи, контроль.

Как только магистраль $OШ_0$ освобождается, если при этом нет запросов от других устройств, $ЦПр$ немедленно занимает $OШ_0$ для выполнения цикла обмена с $ОП$. Завершив цикл обмена по $OШ_0$, процессор самоблокируется на время выполнения операций обработки данных. Работа процессора сопровождается интенсивным обменом с $ОП$ и накопителем. Так как приоритет процессора на $OШ_0$ самый низкий, то ему приходится ожидать освобождения $ОП$, если она занята обменом с другими устройствами. В связи с этим необходимо учитывать потерю производительности процессора. Зависимость производительности $ЦПр$ от загрузки шины можно выразить следующей формулой:

$$B = B^*(1 - \rho_{ш0}), \quad (3.11)$$

где B^* — максимальная производительность $ЦПр$; $\rho_{ш0}$ — загрузка общей шины. Загрузка шины априорно неизвестна, поэтому до выполнения расчетов невозможно указать фактическую производительность процессорного элемента.

Потерю производительности $ЦПр$ можно учесть, выполняя итерационную процедуру, охватывающую решение задачи при фиксируемом $\rho_{ш0}$. Аналогично можно учесть снижение производительности $ППр$, учитывая загрузку шин $ПП$.

Цель проектирования — выбрать число периферийных микропроцессорных подсистем $ПП$, создающих условия для распределенного группового управления K объектами в режиме реального времени.

Исходные данные для проектного анализа. Число точек управлений и глубина мультипрограммирования ОУВС $K = 10$. Паспортная производительность процессора $B^* = 7 \cdot 10^8$ операций/с. Каждая $ПП$ состоит из группы из пяти $УСО$ ($m = 5$). Любая из задач УВС в очередном цикле может обратиться к одному из $УСО$ с вероятностью $p = 1/m$. $УСО$ взаимодействует с $ОУ$ в течение времени $v_1 = 40$. В этом интервале $УСО$ и $ППр$ участвуют в действиях по первичной обработке информации, совмещаемых с работой $УСО$. После этого следует запрос на передачу данных в $ОП$ центральной подсистемы. Обмен выполняется по тракту: $УСО — OШ_i — OШ_0 — ОП$, т. е. сообщение захватывает одновременно несколько ресурсов. Время обмена $v_2 = 15$. Полученные данные обрабатываются $ЦПр$. Трудоемкость обрабатываемой программы $\Theta = 8,2 \cdot 10^9$ операций. Если ОУВС состоит из нескольких одинаковых $ПП$, то часть действий $ЦПр$ передается на $ППр$ за счет их разгрузки, связанной с увеличением числа $УСО$ и уменьшением частоты обращения к $ПП$. Трудоемкость обрабатываемой программы $ЦПр$ при добавлении нового $ППр$ уменьшается на $1,7 \cdot 10^9$ операций.

Необходимо выбрать наименьшее число *ППр* распределенной ОУВС, обеспечивающих пропускную способность задач управления $\Lambda_0 > \Lambda = 0,035 \text{ с}^{-1}$, т. е. должен реализоваться соответствующий цикл управления объектом. Для улучшения иллюстративности расчетов в тракте одновременного захвата ресурсов не будем учитывать соперничество между *УСО* за *ОШ ППр*.

При выполнении расчетов воспользуемся агрегированием узлов, относящихся к *ПП*. Модель этого агрегата приведена на рис. 43, а.

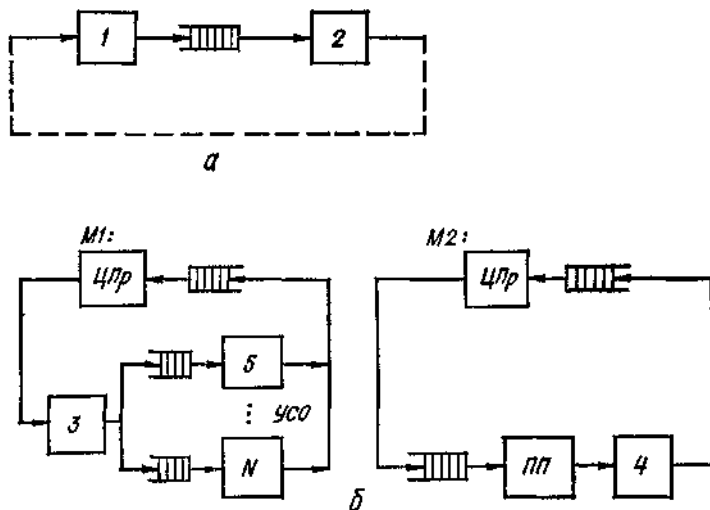


Рис 43. Расчетные модели распределенной ОУВС:

а — модель периферийной подсистемы; б — модель конкуренции за общую шину

В ней представлены следующие действия: узел 1 отражает взаимодействие *УСО* и *ОУ* в течение времени ν_1 , а узел 2 — передачу данных. Очереди к *УСО* территориально сосредоточены в центральном процессоре, поэтому они не входят в агрегат и не отражены на рисунке. Закоротив выход и вход модели, определим ее пропускную способность. В табл. 3 приведены значения $\lambda(K)$ и $b(K) = \lambda(K)/\lambda(1)$, определяющие эквивалентный агрегат в виде узла с переменной скоростью обслуживания $\lambda(K)$ и соответствующей емкостью $b(K)$.

Таблица 3

K	1	2	3	4	5	6	7	8	9	10
$\lambda_0 (K)$	0,018	0,034	0,046	0,055	0,061	0,064	0,065	0,066	0,067	0,068
$b (K)$	1	1,88	2,578	3,078	3,389	3,572	3,66	3,689	3,6	3,72

На рис. 43, б приведены модели $M1$ и $M2$, описывающие одновременный захват шин. В модели $M1$ точно отражены действия $УСО$, представленные группой параллельных очередей и узлов с временем обслуживания $v = v_1 + v_2$. Узел 3 представляет априорно неизвестное дополнительное время, связанное с конфликтом за общие шины. В модели $M2$ $ПП$ представлена узлом с переменной скоростью обслуживания, вычисленной выше, и очередью к ней, формируемой в $ЦПр$. В модели $ПП$ точно учтены действия, связанные с обменом через $ОШ$, однако не отражено время, связанное с ожиданием в очередях 2 — N к $УСО$. Узел 4 модели $M2$ предназначен для представления этого времени, которое точно отражено в модели $M1$.

Последовательный расчет моделей должен учитывать, что время задержки в узле 3 модели $M1$ берется из $M2$, причем оно равно времени ожидания в узле $ПП$. Аналогично, найденное в $M1$ время ожидания $УСО$ берется в качестве задержки в узле 4.

После завершения итераций в моделях $M1$ и $M2$ производится сравнение фактической загрузки $ОШ_0$ с предполагавшимся ее значением, и по выражению (3.11) корректируется производительность $ЦПр$. Вычисления цикла корректировки $\rho_{ш0}$ охватывают цикл расчета моделей $M1$ и $M2$. Расчет $M1$ и $M2$ выполняется описанными ранее методами теории сетей с очередями.

В табл. 4 показана итерационная последовательность расчета моделей $M1$ и $M2$ на последнем цикле уточнения производительности $ЦПр$ в системе с двумя $ПП$ ($n = 2$).

Таблица 4

Модель М1			Модель М2		
Время ожидания шин	Загрузка ЦПр	Время пребывания в УСО	Время ожидания УСО	Загрузка ЦПр	Время пребывания в шинах
0	0,954	82,67	27,67	0,78	112,31
57,31	0,831	76,23	21,23	0,656	177,21
122,21	0,66	70,46	15,46	0,635	191,76
136,76	0,63	69,14	14,14	0,62	192,49
137,49	0,63	69,14	14,14	0,62	192,49

Процесс решения сошелся через 5 циклов.

В табл. 5 сопоставлены некоторые из характеристик для ОУВС с одним ($n = 1$) и двумя ($n = 2$) ППр.

Таблица 5

n	λ_0	$\rho_{пр}$	$\rho_{УСО}$	$\rho_{Ш0}$	$\rho_{Шf}$	$B_{пр}$	c
1	0,028	0,78	0,308	0,224	0,224	$546 \cdot 10^3$	357
2	0,039	0,62	0,215	0,312	0,156	$434 \cdot 10^3$	256

Из данной таблицы видно, что условие проектной задачи оптимизации выполняется при $n = 2$, обеспечивая требуемое значение пропускной способности λ_0 при минимальной сложности ОУВС. Число ППр, равное двум, обеспечивает режим реального времени ОУВС.

Сравнивая результаты расчетов табл. 5, можно видеть, что нагрузка шин ППр при переходе от одной к двум ППр падает, уменьшается нагрузка УСО, нагрузка ОШ₀ увеличивается. Это объясняется тем, что с увеличением числа подсистем уменьшаются задержки в УСО, увеличивается число обращений к ОШ центральной подсистемы, что в свою очередь приводит к потере производительности ЦПр. В связи с перераспределением работ время обслуживания задачи на процессоре остается неизменным. Целесообразно отметить, что в распределенной системе с двумя ППр потенциальная производительность ЦПр не может быть использована в силу конфликта за общую шину ОШ₀ между различными источниками обмена. Захват шины снижает производительность ЦПр от 700 до 434 тыс. операций/с.

Приведенный пример иллюстрирует широкие возможности итерационных методов над комплексами взаимодействующих моделей при расчете распределенных ОУВС с общими шинами.

Проектные расчеты многошинной ОУВС. Приведенная на рис. 44 система включает объекты управления OY_i , $i = 1, 2, \dots, K$, процессоры с

функциональной специализацией действий $Pr_j, j = 1, 2, \dots, m$ и общие шины $OШ_b, b = 1, 2, \dots, B$.

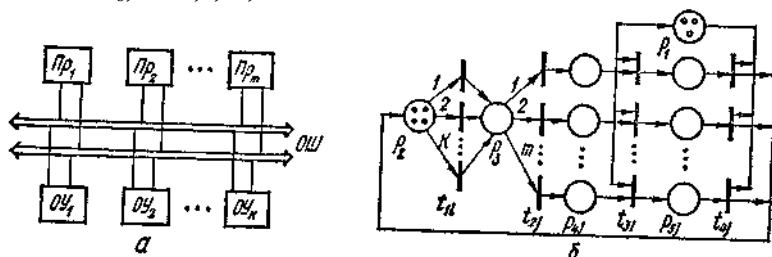


Рис. 44. Многошинная мультипроцессорная ОУВС (а) и модель ее работы (б)

Модель описывает прохождение задач управления через систему. Содержательный смысл элементов модели: t_{1i} — множество из K переходов, отражающих индивидуальную работу объектов $OУ_1 \dots OУ_K$ по обработке управляющих воздействий; p_3 — список процессов, завершивших управление объектами; t_{2j} — множество из m переходов, отражающих запросы процессов к одному из m процессоров; p_{4j} — очереди запросов на выделение шины; t_{3j} — операции выделения одной из шин, представленных метками в позиции p_1 ; p_{5j} — очереди к процессорам; t_{4j} — совместная работа одного из объектов управления, шины и процессора, а также следующее в конце работы освобождение шины. Начальное состояние системы отражено следующим маркированием сетевой модели: число меток в позиции p_2 равно числу управляемых объектов K , причем каждой метке сопоставлен атрибут-номер процесса; число меток в позиции p_2 равно числу шин (метки, обозначенные кружком, представляют шины); метки и позиции, представляющие процессоры, охватывают соответствующие переходы группы t_4 и не изображаются на схеме, поскольку исключение их не исказит структуру процессов.

Модель отражает следующий порядок прохождения i -го процесса через систему: автономная работа $OУ_i$ — (t_{1i}); обращение к Pr_j — (t_{2j}); ожидание шины — (p_{4j}); выделение шины — (p_{3j}); захват вторичного ресурса (процессора) и совместные действия $OУ_i, Pr_j$ и $OШ_b$ по обмену, обработке данных и управлению объектом — t_{4j} . Управляющая система содержит $B < \min(K, m)$ шин, любая из которых может представляться по запросу для связи между $OУ_i$ и вызванным Pr_j .

Автономная работа объекта выполняется в течение времени t_{1j} , а совместные действия t_{4j} — в течение t_{4j} . Задана также вероятность

обращения процесса i к Pr_j . Вероятности p_{ij} сопоставляются дугам, исходящим из p_3 в t_{2j} .

Цель проектирования — выбрать число общих шин ОУВС, изучив их влияние на производительность системы управления и эффективность использования оборудования.

Расчетные модели $M1$ и $M2$ многошинной ОУВС приведены на рис. 45.

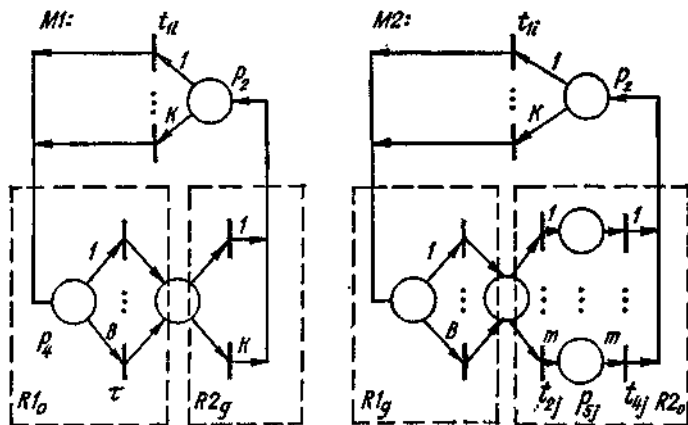


Рис. 45. Расчетные модели многошинной мультипроцессорной ОУВС

Они получены интерпретацией модели с совместным использованием рис. 44, б. Здесь первичный ресурс $R1$ — шины, а вторичный $R2$ — процессоры. $R1_o$ в $M1$ отражает В-канальный прибор (шины), время использования которых равно τ ; p_4 — очередь к шинам. $R2_o$ в $M2$ соответствуют очередям к процессорам p_{5j} и использованию процессоров t_{4j} . Дополнительные элементы $R2_d$ указывают среднее время ожидания процессоров и связаны со средним временем пребывания в очередях p_{5j} модели $M2$. Аналогично, $R1_d$ в $M2$ отражает среднее время ожидания в очереди к шинам, которое равно среднему времени пребывания в очереди-позиции p_4 модели $M1$.

В табл. 6 приведены характеристики ОУВС при различном числе шин B для следующих исходных данных: $K = 5$; $m = 4$; $v_1 = v_2 = 10$; вероятности обращения объектов i к шинам j одинаковы и равны $p_{ij} = 0,25$, т. е. коэффициенты посещения процессоров $\alpha_2 = 0,25$, а объекта $\alpha_1 = 1$.

Таблица. 6

B	λ_0	R_0	R_{Π}	$R_{\text{ш}}$
1	0,099	1,95	1,84	0,99
2	0,118	2,64	2,56	1,778
3	0,132	3,28	3,05	2,16
4	0,14	3,36	3,12	2,24

В таблице указаны производительность ОВС λ_0 и среднее число используемых приборов объекта R_0 , процессоров R_{Π} и шин $R_{\text{ш}}$. Из табл. 6 видно, что при числе шин, равном трем и четырем, характеристики системы отличаются мало. Следовательно, можно сделать вывод: использование в системе более трех шин при заданном числе объектов и процессоров нецелесообразно.

Выбор оптимального объема основной и числа контроллеров внешней памяти. Система, модель которой приведена на рис. 8, имеет два вида пассивных ресурсов — основную память $ОП$, выделяемую динамически, и контроллеры накопителей.

Оба ресурса выделяются и освобождаются по запросам, придаваясь активным ресурсам — процессору $Пр$ и накопителям архивной памяти $АП$. ОУВС имеет максимальную производительность K , если число разделов $ОП$ равно числу задач системы, а число периферийных контроллеров равно числу дисков. Ограничение объема $ОП$ и уменьшение числа накопителей может снижать пропускную способность системы.

Оптимальным назовем такое число пассивных ресурсов системы, которое не снижает производительность системы больше заданного уровня. Пусть $\lambda(p)$ — пропускная способность ВС на заданном наборе пассивных ресурсов p .

Цель проектирования — найти такой допустимый набор пассивных ресурсов p , который обеспечил бы минимальную стоимость ОУВС, не снижая ее производительности более чем на ε процентов, т. е. необходимо найти

$$\min_{p \in \Omega} \|p\|$$

при соблюдении условия $\lambda - \lambda(p) \leq \varepsilon$, где $\|p\|$ — стоимость ресурсов p ; p_i — число ресурсов типа i ; $i = 1, 2, \dots, N$; Ω — множество допустимых значений p .

Для того, чтобы учесть в расчетах пассивные ресурсы, выполним замыкание точек $B2$ и $O2$ модели (см. рис. 8) и представим дисковую

подсеть одним агрегатом-узлом с переменной скоростью обслуживания. Затем делается следующий шаг агрегирования сетевой модели — определяется пропускная способность закороченной подсети, ограниченной точками $B2$ и $O2$ на схеме рис. 8. Вся эта подсеть заменяется одним узлом с переменной скоростью обслуживания. После выполнения этих двух этапов получена сеть, содержащая всего два узла — узел, представляющий объект управления, и узел, являющийся эквивалентом ОУВС. Первый из них имеет тип D (отработка управления объектом), а второй — узел типа FIFO с переменной скоростью обслуживания.

Проведем проектный анализ ОУВС, управляющий пятью объектами ($K=5$). Пронумеруем элементы системы (см. рис. 8): OU — 1, OP — 2, Pr — 3, AP — 4. Временные характеристики процессов: $\tau_1 = 30$; $\tau_3 = 0,5$, $\tau_4 = 1$. Маршрут задач представляется коэффициентами посещения $\alpha_1 = 1$; $\alpha_3 = 30$, $\alpha_4 = 6$. Расчеты выполним при условии, что число динамически запрашиваемых разделов памяти и число контроллеров накопителей изменяется от 1 до 5. Число накопителей равно 5.

В табл. 7 приведены результаты расчетов производительности ОУВС.

Таблица 7

Вариант комплектации	1	2	3	4	5
Число разделов ОП	5	5	4	3	1
Число накопителей	5	4	3	2	1
λ_0	0,024	0,019	0,0188	0,0182	0,0177

Отметим, что целесообразно рассматривать только такие варианты комплектации ОУВС, при которых число разделов основной памяти больше или равно числу накопителей, поскольку в противном случае они будут простаивать без использования. Анализируя таблицу, можно видеть, что переход от первого варианта комплектации ко второму снижает производительность ОУВС на $\varepsilon = 21$ %, а пятый вариант — на $\varepsilon = 28$ %, при значительно меньшей сложности системы. Используя результаты расчетов, можно выбрать комплект ОУВС.

9.3.5. Оптимизация размещения файлов и вычислительных работ в локальных управляющих сетях

Алгоритмы распределения вычислений по процессорам и файлов по носителям формулируются в единообразной постановке, сводящей

выполнение двух перечисленных задач назначения к единой задаче размещения работ в локальной сети микро-ЭВМ. Математическое обеспечение задачи размещения вычислений и файлов базируется на использовании естественной сетевой интерпретации структур и процессов. Формирование плана распределенной обработки выполняется комбинированием эвристических комбинаторных методов и методов локальной оптимизации (типа метода вектора спада), реализующих направленный поиск. К существенным особенностям, которые принимаются во внимание в методах сетевой оптимизации, относятся учет очередей к устройствам хранения и обработки; дисциплин обслуживания запросов; временных задержек на обработку, доступ к данным и обмены. Задача размещения предполагает использование сетевых моделей как системы, так и алгоритма распределенной обработки (см. раз. 9.2.6), отражающих взаимосвязь вычислительных операторов между собой и с общими данными.

Процесс размещения вычислительных операторов и файлов по ресурсам ОУВС реализуется путем установления соответствия между элементами сетевых моделей ОУВС и алгоритмов с их данными.

Модель функционирования ОУВС представляет собой, таким образом, единую структуру, отражающую взаимодействие процессов и ресурсов. Планирование распределенной обработки, изменяющее место выполнения программных модулей и размещение данных, состоит в выборе маршрутов процессов на ресурсной среде. Изменение направлений движения потоков задач влияет на эффективность использования оборудования, время реакции ОУВС и т. п.

Файлы желательно локализовать в тех запоминающих устройствах, которые приданы процессорам, использующим данные. Это предотвращает задержки на пересылку информации, снижает загрузку линий связи, увеличивает использование локальных запоминающих устройств и может повышать производительность системы. Однако если много файлов разместить на одном устройстве памяти, характеристики ОУВС ухудшаются из-за увеличения задержки в очереди к этому ресурсу.

Задача распределения файлов сложна и нетривиальна, однако потенциальные выгоды от ее реализации могут быть значительными. Задача планирования вычислений в мультипроцессорных вычислительных системах и сетях ЭВМ заключается в следующем. Оптимизация вычислений делается для минимизации времени выполнения набора задач, с целью реализации задач к заданным срокам, для максимизации загрузки аппаратуры. В качестве моделей

алгоритмов и их наборов используются графы. Однако критерии оптимизации не могут быть непосредственно соотнесены с измеряемыми в ВС характеристиками, такими как время реакции ОУВС, поскольку идя от граф-схемы алгоритма, трудно учитывать задержки в очередях к различным ресурсам. Чаше всего предлагается применять либо методы целочисленного программирования, либо эвристические приближенные методы. Выбор пригодных эвристик — не простая задача. Установлено, например, что при максимизации системы производительности система эвристик, базирующаяся только на анализе загрузки, может быть недостаточной.

Улучшение качества ОУВС требует понимания не только инструментов оптимизации, но и самой системы. Знание механизмов работы распределенной системы позволяет построить ее модель в виде сети процессоров, средств обмена и хранения информации, связанных очередями для запоминания запросов. Конкретное назначение вычислительных работ и файлов на процессоры и запоминающие устройства позволяет задать исходные данные для аналитического или имитационного моделирования и оптимизации процессов.

Рассмотренные аналитические сетевые модели и описанные в раз. 9.4.2 программные имитационные средства обеспечивают решение задачи назначения вычислительных работ и файлов как в традиционной постановке, предполагающей анализ алгоритмов, так и в современной, предполагающей учет взаимодействия процессов исполнения программ на конкретных ресурсных средах. Поскольку вопросы моделирования сетей с очередями рассмотрены достаточно подробно, приведем несколько примеров, иллюстрирующих возможности моделирующих средств по решению названных задач, приведем некоторые из алгоритмов оптимизации и состав базовых средств для автоматизированной поисковой оптимизации распределения вычислительных работ и файлов в локальных управляющих сетях. На рис. 46 изображена модель двухпроцессорной системы, интерпретирующая один из режимов работы ОУВС, физическая структура которой дана на рис. 42.

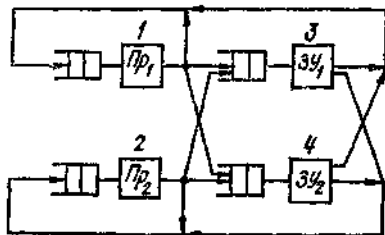


Рис 46. Модель двухпроцессорной системы

Исходные данные включают в себя: R — число различных типов задач распределенной системы, заданных маршрутами по устройствам $i = 1, \dots, N$; $r = 1, \dots, R$ — номер маршрута или потока задач; v_{ir} — среднюю длительность обслуживания задач потока r на устройстве i ; N_{jr} — среднее число запросов из задачи r к файлу j ; δ_i — функцию, характеризующую дисциплину обслуживания устройства r ; α_{ir} — коэффициенты посещения, определенные для всех процессорных элементов и не заданные для запоминающих устройств $ЗУ_i$; m_i — максимальное число файлов, допустимое для записи на $ЗУ_i$. Необходимо найти такое распределение файлов по $ЗУ$, которое обеспечивает максимальную производительность системы

$$\Lambda = \sum_{r=1}^R \lambda_{\Phi}^r. \quad (3.12)$$

Если Φ_1 — множество файлов, размещаемых на $ЗУ_1$, а Φ_2 — на $ЗУ_2$, то коэффициенты посещения этих устройств определяются по формуле

$$\alpha_{ir} = \sum_{j \in \Phi_i} N_{jr}; \quad i = 3, 4; \quad j = \overline{1, J}. \quad (3.13)$$

Так находится число обращений к устройству i от задач типа r .

Пусть $R = 4$; $K_r = 1$, $i = 4$, обращения к $ЗУ$ чередуются с процессорной обработкой, первые две задачи закреплены за процессором $Пр_1$, а остальные — за $Пр_2$. На $ЗУ$ размещен общий банк данных управляющей системы.

Дано: $\Phi_{1r} = \Phi_{2r} = 0,06$; $\Phi_3 = 0,1$; $\Phi_4 = 2,5$; $m_3 = 3$; $m_4 = 12$;
 $j = 1, 2, 3, 4, 5, 6$

$$N_{jr} = \begin{pmatrix} 0 & 2 & 2 & 4 & 1 & 0 \\ 5 & 0 & 2 & 0 & 0 & 3 \\ 0 & 2 & 4 & 0 & 3 & 1 \\ 6 & 0 & 4 & 3 & 0 & 2 \end{pmatrix}.$$

Для расчета производительности системы необходимо сформировать множество файлов Φ_1 и Φ_2 найти α_{ir} по формуле (3.13), после чего выполнить расчет характеристик системы по формулам метода анализа средних для многопоточковых сетей с очередями и, наконец, по выражению (3.12) определить искомую величину, характеризующую суммарную пропускную способность системы. В данном случае число возможных вариантов назначений файлов на $ЗУ$ $C_6^3 = 20$.

Расчеты показывают, что максимальная пропускная способность BC $\Lambda_{\max} = 0,131$ обеспечивается при $\Phi_1 = \{1, 2, 4, 5\}$; $\Phi_2 = \{3, 6\}$, а $\Lambda_{\min} = 0,056$ — при $\Phi_1 = \{1, 2, 5, 6\}$; $\Phi_2 = \{3, 4\}$. Из анализа результатов легко видеть, что перестановка файлов 4 и 6 между $ЗУ$ уменьшает более чем в два раза пропускную способность системы, что

подтверждает эффективность от внедрения задачи размещения файлов. Даже при таком небольшом числе $3U$ и файлов оптимальный вариант размещения без расчетов получить весьма затруднительно из-за наличия очередей и взаимодействия процессов на общих ресурсах. В рассмотренной задаче размещение файлов происходит при закреплении вычислений за процессорами. В практике может возникнуть и такой случай, при котором размещение файлов фиксируется, а варьируется размещение вычислительных работ по процессорам. И, наконец, оптимизация ОВС может предполагать поиск наилучшего варианта размещения как файлов, так и вычислений. Во всех трех случаях сохраняется одинаковый подход к расчету характеристик на модели и поиску оптимального назначения. При росте числа устройств и распределяемых файлов или вычислительных модулей число возможных назначений очень быстро растет. В связи с тем, что полный перебор возможных назначений обычно затруднителен, можно рекомендовать следующие простые и вместе с тем достаточно эффективные методы оптимизации: **метод последовательного сдвига файлов или вычислительных модулей и метод поиска оптимума на основе перестановок.**

Метод последовательного сдвига предполагает перемещение файла (или вычислительного модуля) по множеству устройств, на которые допустимо назначение. Такой сдвиг аналогичен одномерному поиску лучшего положения данного файла. Далее перемещается следующий из файлов и т. д. Цикл перемещений (итерация) может повторяться. Метод последовательного сдвига аналогичен замене многомерного поиска последовательностью одномерных. Одна итерация предполагает просмотр лишь части из всего множества назначений и позволяет найти локальный экстремум оптимизируемой функции цели.

Поиск оптимума на основе перестановок двух файлов предполагает возможность учета в процессе оптимизации нескольких характеристик системы или процессов. Так, при выборе пары файлов, которые надлежит переставить между $3U$, могут учитываться такие характеристики, как загрузка соответствующего $3U$ и вклад в эту величину того или иного файла. Взяв один из файлов для перестановки, его обменивающийся напарник подбирают таким образом, чтобы выравнивалась загрузка устройств одинаковой производительности и максимизировалась загрузка устройств высокой производительности, т. е. кроме основного критерия оптимизации могут учитываться и косвенные, определяющие подбор пары для перестановки.

Методы последовательного сдвига и перестановок работ дают локальные экстремумы, поэтому могут комбинироваться между собой.

Поскольку сетевые модели позволяют оценивать время реакции системы, пропускную способность, загрузку, стоимость обмена и хранения, на их основе могут вычисляться различные функции цели в задачах оптимизации. Предполагается применение ограниченного набора процедур дискретного поиска для решения проектных оптимизационных задач.

При человеко-машинных процедурах оптимизации необходимо использовать возможности по назначению хорошего начального приближения. Так, начальное размещение файлов необходимо выполнять по принципу: **наиболее часто используемые файлы помещаются на наиболее быстрые устройства.** Может допускаться копирование файлов. Однако в этом случае увеличивается поток корректирующих действий, чтобы копия и оригинал файла совпадали. Следует учитывать узкие места, позволяющие выбрать файлы или программные модули, подлежащие перестановке.

В приведенном выше примере исходные данные содержали число обращений к файлу j t задач r . Если заданы вероятности обращений к файлу j t задачи r то назначение файлов может выполняться в два этапа: методами непрерывного поиска на сетевой модели находят трассовые вероятности, обеспечивающие оптимум целевой функции; назначение файлов выполняют таким образом, чтобы суммы вероятностей обращения к файлам на соответствующих устройствах наилучшим образом приближались к оптимальным трассовым вероятностям. Поиск наилучших трассовых вероятностей для системы выполняется на основе сетевой модели системы единожды (для заданной ее конфигурации). Назначение же файлов на устройства — вычислительно — менее трудоемкий этап, требующий минимизации разностей между оптимальными и фактическими вероятностями обращения к устройствам. Этот этап выполняется при перестройках работы ОУВС ГАПС.

Для обеспечения режима реального времени может потребоваться назначение *взаимосвязанных* вычислительных работ на несколько параллельно работающих взаимодействующих процессоров. Чтобы унифицировать программное обеспечение САПР, для решения задач данного класса также целесообразно применять методы перестановок (вектора спада) широко применяемые для решения задач дискретного математического программирования. Задачу минимизации времени выполнения взаимосвязанных вычислений на многопроцессорной системе с учетом операций обмена можно решить следующим методом. Если выполнена предварительная работа по подготовке

параллельного алгоритма (см. раз. 9.2.5), то для каждой пары операторов (i, j) алгоритма известно отношение связности по данным. Ясно, что сильно связанные операторы (i, j) , требующие общих или взаимопорождаемых данных, необходимо назначать на один процессор, не затрачивая дополнительное время на пересылку информации и ее ожидание. Слабо связанные задачи (операторы) целесообразно назначать на различные процессоры, реализуя распределенную многопроцессорную обработку. Введем следующий критерий качества назначений:

$$F = F_1 + F_2 = \sum_{i=1}^K \sum_{\substack{j=1 \\ g_i \neq g_j}}^K q_{ij} + \sum_{i=1}^K \sum_{\substack{j=1 \\ g_i = g_j}}^K p_{ij},$$

где $i = 1, 2, \dots, K$ — номера задач (операторов); g_i — номер процессора, на который назначена задача i ; q_{ij}, p_{ij} — значения штрафов (например, для слабо связанных задач $q_{ij} = 0; p_{ij} = 1$, а для сильно связанных — $q_{ij} = 1, p_{ij} = 0$); F_1 — штраф за назначение слабо связанных задач на один и тот же процессор (сумма F_1 включает те составляющие штрафа q_{ij} , которые соответствуют слабо связанным задачам (i, j) , назначенным на различные процессоры $g_i \neq g_j$); F_2 — штраф за назначения сильно связанных задач на разные процессоры.

В значениях штрафов p_{ij} и q_{ij} может учитываться не только факт наличия связности, но также и объем данных, передаваемых при взаимодействии операторов, скорость обмена. Назначение задач на процессоры — это вектор $g = (g_1, g_2, \dots, g_K)$, указывающий процессоры, закрепляемые за операторами $1, 2, \dots, K$. Для оптимального назначения задач функция штрафа F имеет минимальное значение.

Метод поиска оптимального назначения основан на вычислении спада штрафа ΔF при перераспределении задач. Если изменение назначения задач приводит к спаду ΔF , то соответствующая перестановка в назначении целесообразна. Перестановки здесь могут реализоваться такими же методами, какие описывались выше. Таким образом создаются условия для определенной унификации в подходах к решению задач и уменьшению **неоправданного разнообразия программного обеспечения САПР**. Результаты решения задачи о назначениях по критерию F могут **использоваться в качестве начального приближения в более точном, сетевом подходе, отражающем задержки в очередях, каналах связи, ожидание в точках синхронизации процессов и учитывающем влияние мультипрограммного режима использования системы**. Назначение вычислительных работ и файлов, получаемое в результате решения оптимизационных задач, используется для загрузки данных и

программных модулей в выбранные ЗУ. Должны также определиться логические коммуникационные пути и скорректироваться условия доступа к данным. Эти действия выполняются средствами операционной системы.

9.4. Реализация программного обеспечения ОУВС

9.4.1. Безбумажная Р-технология моделирования и подготовки распределенного управления в диалоговой среде

Базовым средством автоматизированного проектирования облачных управляющих ВС является программное моделирование. В процессе автоматизированного проектирования проверяются различные проектные предложения и выполняются оптимизационные расчеты. Результаты должны документироваться в форме, обеспечивающей настройку ОУВС на эффективный режим эксплуатации и управления гибким производством. Наиболее отработанные и развитые САПР функционируют в сфере автоматизации программирования. Среди средств поддержки процесса изготовления программ на промышленной основе выделяется Р-технология, направленная на организацию безбумажной работы коллектива программистов, использование графической формы записи алгоритмов на всем технологическом этапе изготовления и эксплуатации программ. Она позволяет автоматизировать управление деятельностью разработчиков-программистов.

Этапу разработки программного продукта предшествует этап проектирования структуры и системной работы ОУВС ГАПС. Поскольку этап системного проектирования ОУВС и этап разработки (настройки) программного обеспечения взаимосвязаны, возможности Р-технологии программирования целесообразно распространить на все этапы технологической подготовки облачного управления гибким производством, обеспечивая преемственность и единство в общей технологической цепочке автоматизированного проектирования.

Согласование концепций Р-технологии моделирования и Р-технологии программирования позволяет организовать единую технологическую линию различных этапов автоматизированного проектирования сложных систем, какими являются ГАПС.

Технологическая подготовка ГАПС включает в себя два взаимосвязанных направления работ. Первое направление

реализуется технологами производства, а второе — технологами по управляющей системе. **Технологическая подготовка ОУВС предполагает: выполнение разнообразных проектных расчетов, призванных создать условия для качественного управления объектом и эффективного использования ОУВС; технологическую подготовку к реализации распределенного программного комплекса; создание управляющих моделей и локальных управляющих программ.** Вслед за технологической подготовкой решению по реализации программно-технического комплекса выполняется работа по изготовлению программного продукта ОУВС и настройке ее структуры. Сформулируем отличительные положения Р-технологии разработки и использования управляющих моделей ГАПС, согласуя их с базовыми принципами Р-технологии программирования.

1. В Р-технологии моделирования и подготовки распределенного управления модели систем, алгоритмов и процессов предлагается рисовать на языке нагруженных сетей (ЯНС). Графический ЯПС обеспечивает высокую наглядность, реализацию на дисплеях, алфавитно-цифровых устройствах, он обладает широким спектром изобразительных возможностей, согласован с достижениями современной математической теории. Графические сетевые языки могут быть оболочкой, в которую погружены языки представления параллельных алгоритмов и языки моделирования. Перевод графической и символьной информации из языка нагруженной сети в определенный язык моделирования может осуществляться путем трансляции. Вместе с тем на основе предложенного графического языка строятся различные интерпретирующие средства, обеспечивающие непосредственное использование графического образа и соответствующих структур данных для решения задач анализа, синтеза, моделирования и оптимизации. Графическое представление алгоритмов, моделей и структур ОУВС позволяет ввести в моделирование и подготовку распределенного управления понятия чертежа и использовать в процессе проектирования стандарты ЕСКД и ЕСПД. Для технологического единства с Р-языком программирования чертеж управляющей модели оформляется в виде следующих полей: официального (рамка и штамп, содержащий учетную технологическую информацию); неформального описания объекта проектирования; формального описания объекта проектирования на ЯНС, содержащего графический образ и специфицирующую информацию. В формировании чертежа участвует ЭВМ.

2. Графический язык представляется иерархией совместимых снизу вверх входных языков моделирования и средств их интерпретации.

Такой подход позволяет объединить достижения теории сетей Петри, сетей с очередями, теории и практики имитационного, численного и аналитического моделирования. Применение раскрашенных сетей обеспечивает перспективы применения цветных дисплеев.

3. Представляемые в памяти и на экране данные должны непосредственно интерпретироваться обрабатывающими программами САПР. Приближение внутреннего и внешнего представления данных создает условия для эффективной диалоговой работы с использованием стандартных и специальных средств программного обеспечения автоматизированного рабочего места, отладку моделей и распределенных алгоритмов.

4. Проектирование осуществляется по безбумажной технологии, вся документация и все чертежи хранятся в памяти машины и используются коллективом проектантов.

5. Основным режимом использования автоматизированного рабочего места (АРМ) является диалоговое взаимодействие между ЭВМ и разработчиком. Причем предусматривается ведение диалога как с инициативой от проектировщика, так и в соответствии с заранее разработанными сценариями, ведомыми от ЭВМ. При совместной работе нескольких проектировщиков обеспечивается управление групповым взаимодействием в соответствии с **групповым сценарием. Сценарий группового общения разработчиков описывается на графическом сетевом языке, поскольку групповое взаимодействие проектировщиков в вычислительной сети по принципам взаимодействия аналогично взаимодействию отдельных ЭВМ распределенной ОУВС.** Диалоговое взаимодействие осуществляется как с использованием директив (команд) базовой операционной системы АРМ, так и с использованием ограниченного естественного языка проектирования, интерпретируемого монитором операционной системы САПР.

6. Проектные решения должны обеспечивать гибкость и перестройку программотехнических компонентов ОУВС ГАПС.

Отметим некоторые особенности реализации языка управляющих моделей ГАПС. Входная информация об объекте проектного анализа может задаваться двумя способами: путем непосредственного обращения к процедурам библиотеки САПР на основе языка данных, либо с использованием графической спецификации объекта проектирования. При втором способе этапу расчета предшествует трансляция символьного экранного файла в данные для обращения к библиотечным процедурам САПР. Сохранение двух способов обращения к процедурам проектирования является естественным, поскольку в практике проектных расчетов применяется как

графический метод спецификации объектов, так и метод текстового общения с ЭВМ и ввода в нее порций запрашиваемых у пользователя данных. Если информация об объекте проектирования представлена компактными структурами данных, то перед их визуализацией может выполняться преобразование информации в форму чертежа.

Для широкого внедрения Р-технологии в процессы моделирования и разработки распределенных систем в САПР предусматривается использование как графических, так и алфавитно-цифровых дисплеев. При использовании графических дисплеев и функциональных клавиш графическая информация на экране представляется точно в такой же форме, какая используется для изображения сетевых моделей на бумаге. Функциональные клавиши используются здесь для изображения таких элементов сетевых моделей, как позиция, переход, стрелка. Кроме стандартных сетевых символов, для компоновки много-листовых чертежей используются символы-ссылки типа межстраничного соединителя. Числовая специфицирующая информация ЯНС представляется отдельными массивами либо непосредственно у элементов биграфа (над или под позициями, переходами, дугами, в разрыве дуг). Если позиции или переходы не нумеруются на вводимой схеме, эту операцию выполняет ЭВМ. Текстовая информация чертежа на дисплее обеспечивает комментарии к рисунку. Комментарии допустимы как в соответствующем поле чертежа, так и в поле спецификации модели, при условии, что введены ограничения на размещение на чертеже нагружающих атрибутов сети. Принятое ограничение предполагает, что чертеж сети не должен загромождаться специфицирующей информацией для позиций, переходов и дуг. Основная часть нагружающих характеристик сети перечисляется в массивах времен срабатывания, управляющих и операционных функций, маркировок сети и т. д. Некоторые из характеристик, например условия движения процессов по дугам (т. е. управляющие функции), могут изображаться над соответствующими элементами сети, или в разрывах дуг.

При использовании алфавитно-цифровых дисплеев, подобно графике Р-технологии программирования, дуги сети изображаются горизонтальными линиями с использованием знаков «минус», и вертикальными линиями с использованием знаков «двоеточие». Стандартное направление дуг — слева направо, однако оно может задаваться знаками «больше» и «меньше». Вершины биграфа изображаются буквами «О» и «I», представляя позицию и переход. Символ «I», стоящий над или под элементами «О» и «I» является символом-расширителем соответствующего элемента и используется при отображении нескольких входящих или исходящих дуг. Позиции и

переходы нумеруются путем записи чисел над изображающими их элементами. Если дуга прерывается записью вида « Kp_i » или « Kt_j », то обеспечивается указание на связь с определенной позицией p_i или переходом t_j (i, j — целые числа).

Трансляция символьного дисплейного файла в компактную структуру данных для использования прикладными программами САПР выполняется на базе преобразователей с магазинной памятью (МП). В процессе работы МП-преобразователя осуществляется поиск символа *, указывающего на фиктивное начало рисунка, и просмотр символьной информации вдоль биграфа. Одновременно с разбором графической программы осуществляется контроль ее правильности. Чертеж на ЯНС изображается в поле, соответствующем размеру двойного экрана дисплея с рамкой или без нее. Для реализации структурного подхода к моделированию и проектированию допустимо создание чертежей меньшего размера, число строк которого указывается в описателе.

На рис. 47 приведен пример модели стохастической сети, представленной чертежом на ЯНС на экране дисплея.

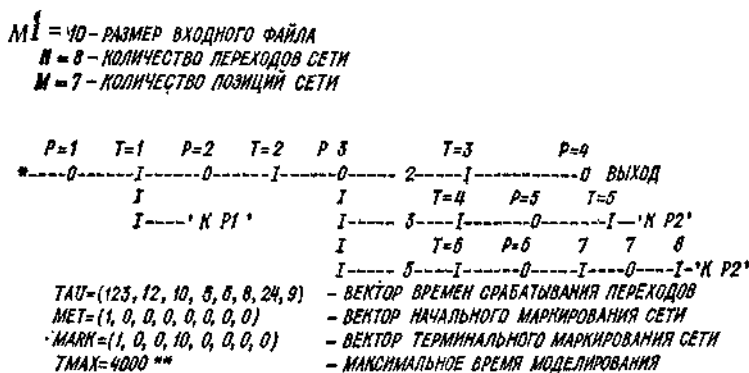


Рис. 47. Модель системы, представленная чертежом на экране дисплея (копия экранного файла)

Чертеж содержит указание на его размер (в числе строк), графическую, числовую информацию и комментарии. Трансляция дисплейного файла порождает следующие наборы данных: матрицы I и O , отображающие связи между позициями p_i и переходами t_j сетевой модели, матрицу вероятностей переходов, компоненты которой на экранном рисунке пишутся в разрывах дуг. Векторы начального маркирования, временные характеристики объекта и указания к организации модельных расчетов непосредственно специфицируются соответствующими массивами и числами на экранном чертеже. При

создании экранной информации проектировщик использует стандартные средства текстового редактирования АРМ и проблемно-ориентированные средства диалогового взаимодействия. Остановимся на реализации средств интерактивного взаимодействия. Их главная особенность состоит в полном отделении средств управления диалогом от прикладных программ САПР и генерируемости (настраиваемости) диалоговых мониторов на потребности конкретных программ и пакетов.

Для повышения производительности труда пользователей САПР и уменьшения интеллектуальной нагрузки на разработчика диалоговые средства реализуют следующие принципы взаимодействия: 1) общение с ЭВМ, происходящее на ограниченном естественном языке; 2) контролируемость интерактивного взаимодействия, в том числе возможность прервать диалог, начать сначала и т. д.; 3) выразительность, легкость изучения, широкое использование принципа умалчивания; 4) синтаксический и содержательный контроль поступающей от пользователя информации; 5) применение подсказок пользователю и полноту системы диагностических сообщений: желательно, чтобы диагностическое сообщение не только фиксировало факт нарушения, но и указывало путь его устранения; 6) индикацию протокола диалогового взаимодействия, осуществляемую без нарушения правил составления чертежей и информации о проектных работах конструктора, работающего за АРМ; 7) удобство выходных форм результатов и управляемость уровнями обобщения или детализации выходной информации; 8) возможность доступа в любой точке диалога к общесистемным средствам; 9) генерируемость диалоговых сценариев, создающую условия для расширяемости и удобства в поддержании и обслуживании программного обеспечения САПР.

В состав диалогового обеспечения включаются средства управления взаимодействием человека с ЭВМ и средства генерации диалогов.

Управление взаимодействием — это прикладная программа, включающая в себя: анализатор синтаксиса, выполняющий синтаксический разбор входных строк, поступающих в ЭВМ со стороны пользователя; интерпретатор команд пользователя, связывающий входные строки диалога с внутренним представлением команд человека в ЭВМ; монитор диалога, управляющий действиями системы в соответствии со входными запросами пользователя; подсказчик пакета, обеспечивающий работу пользователя путем выдачи вспомогательной информации о его возможных действиях; информатор, сообщающий о составе и способе работы с конкретным пакетом программ.

На рис. 48 приведена последовательность работы монитора интерактивной системы проектирования, увязанная с действиями человека за экраным пультом.

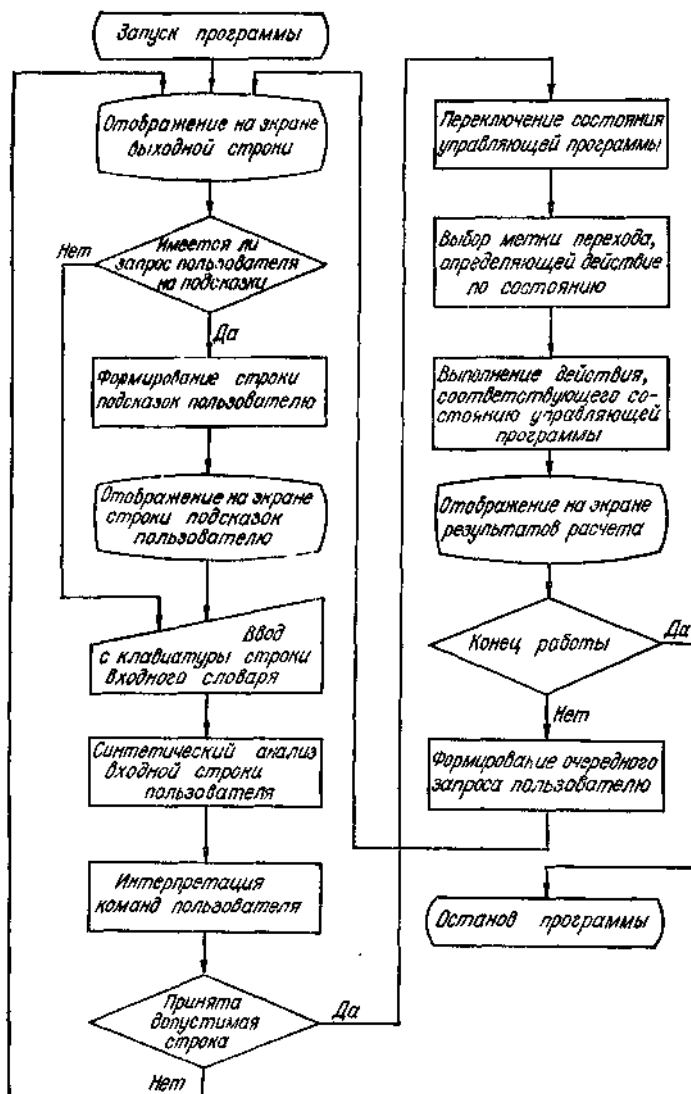


Рис. 48. Алгоритм действий монитора диалога

Монитор управления взаимодействием исполняется в среде базовой операционной системы АРМ как прикладная программа в режиме пользователя, поэтому на его фоне исполнимы любые стандартные системные команды, отрабатываемые базовой операционной системой. Такой подход к управлению взаимодействием обеспечивает активную роль в диалоге как для пользователя, так и для ЭВМ. При управлении от диалогового монитора активная роль принадлежит ЭВМ, а пользователь формирует ответы, при инициативе же пользователя управление обеспечивается монитором базовой операционной системы. Монитор диалога реализуется на основе автоматного подхода.

На рис. 49 приведена структура и отражено взаимодействие средств управления диалоговыми пакетами САПР.

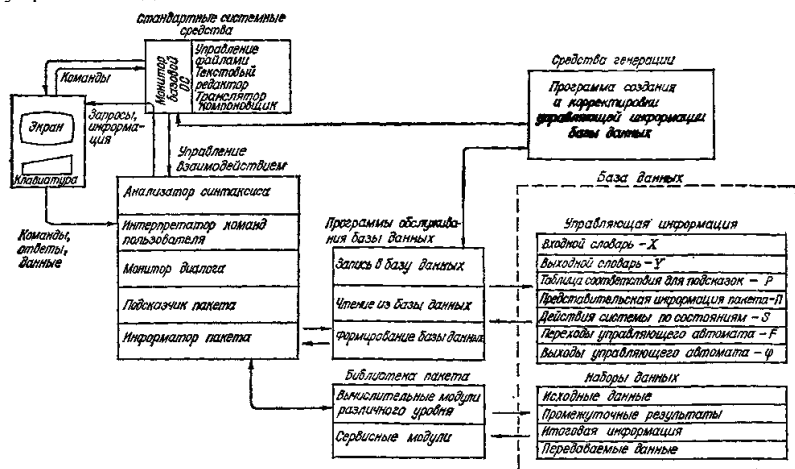


Рис. 49. Структурная схема средств интерактивного пакета

Управление диалоговым пакетом осуществляется по стандартным системным командам через мониторы базовой операционной системы или с помощью средств управления взаимодействием, выдающих запросы и информацию через стандартные системные средства АРМ на экран. Ответы пользователя САПР на сформированные к нему запросы интерпретируются и используются монитором диалога. Управляющая информация монитора диалога представлена в базе данных (рис. 49). Распознав входную строку $x \in X$ и учтя текущее состояние диалога $s \in S$, монитор изменяет состояние диалога, реализуя переход $s(p+1) = F(s(p), x(p))$, где p — момент асинхронного времени. Таблица действий по состояниям

определяет реакцию системы, состоящую в выполнении конкретных расчетов или обслуживающих программных действий. Реакция $y \in Y$ системы может определяться только результатом выполнения программы, либо зависеть также от принятой ранее входной последовательности x , т. е. $y(p+1) = \varphi(s(p), x(p))$.

Сценарии диалога предназначены для регламентирования последовательности действий пользователей САПР. Поскольку структура диалога определяется управляющей информацией базы данных, подготовка сценария диалоговой работы с конкретной программой или пакетом осуществляется путем подготовки автоматной модели сценария. Автоматизация настройки диалога предполагает заполнение управляющей информации базы данных конкретного пакета с помощью средств генерации. Средства генерации представлены управляющей программой, которая опрашивает разработчика конкретного пакета САПР, получает от него состав фраз пользователя и ЭВМ, запрашивает формализованное описание сценария и формирует управляющую информацию в базе данных монитора диалога.

Выбранное для обеспечения САПР направление поддержки проектных работ создает условия для использования облачной технологии в процессе системотехнического проектирования ОУВС ГАПС, при отладочном моделировании процессов управления и подготовке управляющих программ для распределенных ОУВС.

9.4.2. Особенности реализации программного обеспечения систем имитации и управления процессами ГАПС

Отличительная особенность рассматриваемого в работе подхода состоит в организации программного управления процессами автоматизированного производства и ОУВС путем интерпретации сетевых облачных управляющих моделей. На этапе проектирования процессов управления или их гибкой перестройки интерпретация облачных управляющих моделей выполняется с целью анализа алгоритмов управления и решения различных оптимизационных задач, вытекающих из общей цели системотехнического проектирования. Будучи спроектированной и отлаженной с помощью систем имитации, облачная управляющая модель может использоваться для управления процессами автоматизированного производства. На этом этапе интерпретация облачной управляющей модели выполняется с

помощью тех же программных средств, которые использовались на этапе проектного анализа.

Представляя собой единый комплекс, системы имитации и управления обеспечивают повышение качества и сокращение сроков проектирования и перестройки системотехнической работы ОУВС и ГАПС в целом. Основные методологические принципы разработки моделей имитации и управления сформулированы в раз. 9.1.4, а состав средств их анализа и интерпретации охарактеризован на рис. 11.

Реализация широкого спектра моделирующих систем осуществляется на условиях использования единой базы данных, расширяемости, переносимости, модульности, совместимости снизу вверх, функциональной избыточности и других стандартных требований к принципам организации программного пакета.

Программное обеспечение пакета систем имитационного моделирования включает в себя модули управления моделированием; программы создания и обслуживания базы данных моделирования; сервисные программы. Моделирующие системы используют структуру данных, работа с которыми унифицирована в пределах всего комплекса моделирующих средств. Для представления графических сетевых объектов широко используются сетевые структуры данных. Сетевая структура использует два типа объектов — элементарные и составные. Для представления простых моделей, типа сетей Петри, достаточно использования элементарных сетевых структур. По мере усложнения сетевых моделей усложняются и структуры данных.

Основной скелет данных «нагружается» дополнительными структурами — таблицами, списками, очередями и т. д. Для работы с каждым из типов данных созданы обслуживающие программы. В их числе программы создания структур данных, записи, поиска, сортировки, упорядочения и чтения данных. Управляющие программы моделирующих систем интерпретируют сетевые структуры данных, модифицируют их в соответствии с правилами управления моделированием образуют сбор и обработку результатов имитации. Сервисные программы создают условия для ввода и вывода информации и удобном для пользователя формате. В состав САПР входят пакеты для имитационных, численных и аналитических проектных расчетов. Но несмотря на различие в методах, в системе проектирования сохраняется общая схема диалогового обеспечения, отраженная на рис. 49.

Основой для программной реализации систем моделирования и управления являются формальные описания соответствующих классов сетевых моделей, которые рассмотрены в раз. 9.2. Имитация динамики или статики объектов выполняется путем численного решения

уравнений состояний. По сложности и изобразительной возможности сетевые модели и соответствующие им *программные реализации* можно упорядочить в следующую последовательность:

Сети Петри (простые или базовые). Их динамика воспроизводится рекуррентным алгебраическим уравнением (2.2) и логическим уравнением (2.3). Последовательное накопление маркирований и связей между ними порождает граф достижимых состояний сети. *Временные сети Петри.* Здесь события упорядочиваются во времени, поэтому кроме воспроизведения уравнений (2.2) и (2.3) имитация требует создания упорядоченных списков переходов, определяющих порядок наступления событий. Управляющая программа системы моделирования динамики временных сетей организует выполнение следующих основных действий: расчет условий возбуждения, размещение моментов срабатывания и номеров переходов в упорядоченный по времени список событий; определение момента времени и номеров переходов, соответствующих ближайшей по времени группе действий из списка событий, и выработка управляющего вектора U ; расчет нового состояния (маркирования) системы; анализ (диагностику) качественных характеристик системы и ее процессов (живость, безопасность, ограниченность, достижимость терминального состояния, неопределенность в перемещении метки или конфликт за нее, тупик и т. д.). Расчет количественных характеристик, включающих в себя общее число срабатываний переходов t_j , среднее и максимальное числа меток и среднее время пребывания меток в позициях p_j , загрузка переходов; представление пользователю динамики системы в старт-стопном режиме, в виде развертки диаграммы состояний, отображение текущих и итоговых качественных и количественных характеристик моделируемой системы на экране дисплея, документирование результатов.

Стохастические сети. Стохастическая сеть — это такая временная сеть, в которой дугам $(p_i t_j)$ сопоставлены вероятности p_{ij} . Конфликт за метку, расположенную в позиции p_i , разрешается путем разыгрывания исходов событий с помощью генератора случайных чисел.

Вероятности p_{ij} позволяют отображать случайные блуждания меток (процессов) в сетевой модели. Если метка участвует в возбуждении нескольких, например k , переходов одновременно, то $\sum_{i=1}^k p_{ij} = 1$.

Управляющая программа для моделирования стохастических процессов в сетях дополняется генераторами случайных чисел, которые используются при разыгрывании исходов событий и имитации законов распределения временных свойств систем.

Марковские сети. Управляющая программа для расчета характеристик марковских сетей учитывает их особенность, в том числе и невозможность наступления двух событий одновременно. Для описания структуры систем используется их представление в виде сети Петри. При порождении пространства состояний задается допустимое начальное маркирование, и для всех формируемых состояний организуется продолжение вычислений по формуле (2.2). Для каждого достижимого маркирования находится множество единичных управляющих векторов, причем каждый из этих векторов должен удовлетворять отношению (2.3). После программного порождения пространства состояний выполняются действия по составлению линейных алгебраических уравнений Колмогорова — Чепмена, рассчитываются вероятности состояний (маркирований) и затем вычисляются все необходимые характеристики марковских процессов. Использование сети Петри для описания структуры и функций систем, а также для формирования пространства состояний обеспечивает полную автоматизацию расчета марковских сетей.

Нагруженные сети. Перечисленные выше сетевые модели адекватно представляют системы с однородными процессами и ориентированы на быстрый проектный анализ специальных свойств реальных систем. Наиболее широкий уровень имитационного воспроизведения достигается с помощью облачных моделирующих средств, использующих для описания процессов и систем нагруженные сети. На схеме на рис. 11 перечислены некоторые из классов моделирующих систем, в которых отражаются нагружающие атрибуты и отношения. В моделирующих системах с различаемыми метками каждой из них сопоставляется описатель, который указывает номер позиции, где размещается метка, номер перехода, работу которого возбудила данная метка, номер метки, набор параметров, описывающих свойства (атрибуты) меток, адреса связи в различных списках, в том числе в списке меток, упорядоченном в соответствии с отношениями, заданными управляющими функциями. Управляющие программы для моделирования облачных процессов с помощью нагруженных сетей содержат такие компоненты, которые работают быстро в специальных ситуациях упорядочения, поиска и т. д. Поскольку управляющие и операционные функции определяют тип сетевой модели, управляющие программы реализуются с учетом вида этих функций, обеспечивая повышение скорости и сервиса в моделировании. Кроме реализации средств для имитационного обеспечения процессе проектирования, перечисленных на рис. 11, в пакете предусмотрена возможность по настройке управляющих программ имитации на конкретные типы управляющих и операционных функций, например

на реализацию приоритетных правил управления процессами. После ввода данных о структуре сетевой модели проектировщик может уточнить правила управления возбуждением переходов на конкретных входящих дугах-предшественниках, если эти правила отличаются от общих для данного типа сетевой модели.

Логические сети. Для описания функций систем логического управления и действий программируемых логических контроллеров используются логические сети — графический язык высокого уровня, являющийся одной из интерпретаций нагруженных сетей. Разметка дуг сети величинами $c \in \{0,1\}$ позволяет с помощью действий на переходах отображать любую из логических операций. Например, для инвертирования аргумента достаточно соответствующую входную к t_j дугу нагрузить значением $c = 0$. Таким образом снимаются ограничения в представлении логических функций, свойственные базовым сетям Петри. А поскольку динамика логических сетей представима уравнением состояний (2.2), к анализу логических систем и обнаружению дефектов в системах логического управления применимы рассмотренные в раз. 9.2.3 аналитические методы.

Условие возбуждения перехода t_j представляется функцией

$$u(t_j) = \forall p_i \in PRE(t_j) ((c_{it_j} = 0) \rightarrow m(p_i) = 0) \& ((c_{it_j} = 1) \rightarrow m(p_i) \geq 1).$$

Переход t_j возбуждается, если во всех позициях $p_i \in PRE(t_j)$, соответствующих дугам $(p_i t_j)$ с $c_{ij} = 0$, нет меток, но есть хотя бы одна метка во всех позициях p_i , дуги $(p_i t_j)$ для которых нагружены значениями $c_{ij} = 1$. При срабатывании перехода t_j во всех его выходных позициях, следующих за дугами $c_{ij} = 1$, добавляется метка (единица), а из всех входных позиций для которых $c_{ij} = 1$ метка забирается.

Матрица $A = |a_{ji}|$, входящая в уравнение (2.2), для логической сети строится так: $a_{ji} = c_{ji} - c_{ij}$.

Логическую сеть с m -кратными дугами представляют сетью с однократными дугами, нагружаемыми числами $c \in \{0, 1, 2, \dots, m\}$. В этом случае логической сетью реализуется не только двоичная, но и многозначная логика, и для позиций сети должно выполняться, например, условие m -ограниченности. Логические сети с кратными дугами удобны при описании определенных действий со счетчиками, таймерами и регистрами при задании алгоритмов логического управления. Уравнение возбуждения в логических сетях с кратными дугами имеет вид

$$u(t_j) = \forall p_i \in PRE(t_j) ((c_{it_j} = 0) \rightarrow m(p_i) = 0) \& ((c_{it_j} = m) \rightarrow m(p_i) \geq m).$$

Логическая сеть представляет параллелизм в процессах управления и используется после отладки одним из следующих способов:

1) структура данных, представляющая сеть, интерпретируется операционной системой моделирования и управления, как это описано на рис. 7; эталонная управляющая модель описывает логико-временные операции как в управляющей системе, так и на объекте управления; 2) сетью представляют только алгоритм логико-временного управления, который интерпретируется операционной системой управления; 3) алгоритм логико-временного управления, описанный на сетевом языке, транслируется в коды программируемого логического контроллера (компадоаппарата) и исполняется им.

Структурированные сети. Для повышения эффективности и ускорения разработки управляющих моделей ГАПС в САПР предусмотрено использование методологии многоуровневого проектирования «сверху вниз». Она основывается на представлении процесса, модели или структуры системы в виде множества взаимосвязанных элементов. После описания и проверки моделей на них может производиться ссылка как на макродействия, описываемые одним переходом. При модульном проектировании отдельные системные единицы описываются независимо и размещаются в виде наборов данных в архивной базе данных. Извлекаясь из нее и связываясь с помощью определенных правил, эти модули порождают структурированную модель. Связи между модулями могут создаваться на основе семафорных (буферных) позиций, вводимых между заданными переходами двух модулей. Возможны и другие формы комплексирования моделей: слияние двух позиций заданных модулей, слияние переходов, создающее элемент синхронизации и т. д. Описание облачной управляющей модели сопровождается присвоением номера каждому модулю. При создании структурированной сети один и тот же базовый модуль может копироваться из базы данных неоднократно. Каждой копии сопоставляется новый номер. В описании структурированной сети представляются прямо описанные подсети, вызываются описания макропереходов из архива, задаются связи и маркирования. На основе этих данных порождается стандартная информация о параметрах сети, что позволяет выполнять интерпретацию структурированной модели с помощью тех же управляющих программ, которые созданы для анализа и интерпретации неструктурированных сетевых моделей.

Составные сети. В структуре составных сетей могут присутствовать модули, которые принадлежат к сетям различного типа. Например, часть управляющей модели может специфицироваться как простая, а часть — как нагруженная сеть. Это создает условия для

преимущества разработок, естественности отображения объектов и может повысить эффективность управления интерпретацией, что крайне необходимо для организации работы ОУВС в режиме реального времени. Управление интерпретацией составных сетей организуется по иерархическому принципу, когда на верхнем, супервизорном уровне используется монитор — интерпретатор нагруженных сетей, а на нижнем уровне — мониторы — интерпретаторы сетей более простого типа.

При реализации распределенного управления в ГАПС каждый «агрегат» составной сети может управляться собственным монитором операционной системы управления и моделирования. Тогда операционная система становится распределенной. Точки синхронизации (взаимодействия) мониторов во времени могут прогнозироваться на основе информации о структуре и параметрах управляющей модели производства.

Структурируемость составной сети качественно соответствует распределенности процессов и структур ОУВС и ГАПС в целом. А составная управляющая программа-интерпретатор является *распределенной операционной системой программного управления* оборудованием ГАПС.

Поскольку скорость имитационного прогона зависит от способа планирования и поиска активных процессов и действий по их выполнению, то в рамках одного и того же типа сетевых моделей в зависимости от числа процессов и элементов сети эффективнее может работать та или иная управляющая программа. Так, если число процессов (меток) значительно меньше числа переходов t_j сетевой модели, то механизм имитационного планирования эффективнее работает при организации слежения за метками. Если же число меток значительно больше числа переходов и процессы обеспечивают высокую загрузку ресурсов, то быстрее работает механизм имитационного планирования, исходящий из слежения за активностью переходов. Учитывая эту особенность, в пакет имитации и управления систем ввели несколько управляющих программ, реализующих эффективное управление моделированием в зависимости от сложности модели, состава процессов, наличия узких мест. Тип планирования имитации выбирается пользователем или подстраивается адаптивно на основе анализа исходных или текущих данных.

Процедура проектной имитации системы предполагает выполнение пользователем следующих действий: выбора типа моделирующей системы, наиболее приемлемого для выполнения конкретного вида проектных работ; ввода данных, специфицирующих модель

проектируемой системы (в том числе графовых данных); задания режима проектных экспериментов с моделью.

На рис. 50 приведена управляющая модель двухстаночной системы, обслуживаемой роботом.

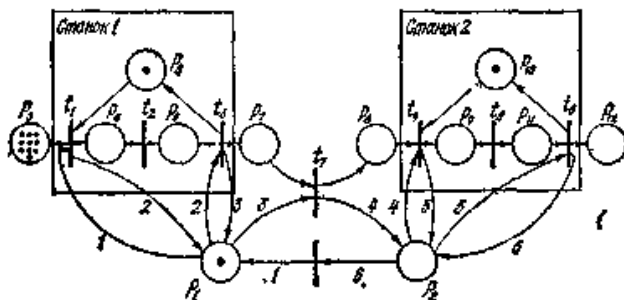


Рис. 50. Управляющая сеть, описывающая обслуживание двухстаночной системы роботом-манипулятором

Робот представлен динамическим элементом модели — меткой. Его начальная позиция p_1 соответствует обслуживанию станка 1, а позиция p_2 — обслуживанию станка 2. Партия обрабатываемых деталей помещается перед началом работы в позицию-бункер p_3 . Обработанные на первом станке детали выгружаются в позицию-накопитель p_7 . Переходы t_1 , t_2 и t_3 , соответственно, представляют операции загрузки, обработки и разгрузки станка. Позиция p_5 и ее управляющая метка создают условия для выполнения этих операций только с одной деталью. Переход детали в позицию p_4 соответствует окончанию загрузки, а в p_6 — завершению обработки. Модели обоих станков имеют аналогичную структуру, представляя соответствующие фазы обслуживания. Переход t_7 имитирует транспортировку детали роботом из позиции p_7 в позицию p_8 , а t_8 имитирует холостой ход робота от станка 2 к станку 1. Для отражения алгоритма действий робота дуги на его пути раскрашиваются признаками 1, 2, ..., 6. Такие же значения атрибута могут присваиваться метке-роботу. Начальное значение атрибута метки, размещенной в позиции p_1 , равно 1. Хотя из p_1 имеется три исходящих дуги, робот реализует то действие, которое соответствует текущему значению атрибута в условии возбуждения переходов.

Для представления динамики робота используем систему моделирования, в которой управляющие функции определены отношением равенства $x=y$, где x — значение текущего атрибута метки, а y — пометка у дуги, исходящей из соответствующей позиции.

Операционные функции имеют вид оператора присваивания $x = y$. Учитывая значения нагружающих атрибутов для меток и дуг, маршрут робота для сетевой модели (рис. 50) представим следующей последовательностью: $W = t_1 t_3 t_7 t_4 t_6 t_8 t_1 \dots$, отражающей один из ВОЗМОЖНЫХ путей робота в двухстаночной системе.

Для моделирования процессов в приведенной системе используем многоклассовую сетевую модель, настроив управляющую программу на реализацию заданных типов операционной и управляющей функций. Если дуги не помечены аргументами для расчета управляющих и операционных функций, соответствующие действия на них управляющей программой не организуются. Исходную информацию для расчета зададим путем ввода данных, представленных маркированным нагруженным биграфом (рис. 51). Введем также времена срабатывания переходов $\tau = (1, 2, 1, 1, 5, 1, 3, 3)$ и время прогона модели $T = 100$, задающее требуемое предельное время обработки партии из 10 деталей, размещенных во входном накопителе p_3 . Времена загрузки и разгрузки станков роботом здесь одинаковы и равны единице, а время транспортировки детали роботом $\tau_7 = \tau_8 = 3$. Модельный расчет показывает, что за заданное время T будет обработано лишь 6 из 10 деталей партии. В связи с тем, что увеличить скорость обработки на станках невозможно, остаются следующие возможности по повышению производительности оборудования: увеличить скорость загрузки-разгрузки и транспортировки деталей роботом или изменить маршрут его движения. Оптимизацию маршрута можно выполнить методами, рассмотренными в раз. 9.3. При заданных здесь временных параметрах обслуживания изменение маршрута робота не обеспечивает обработку партии деталей за время T . Варьирование скоростей загрузки-разгрузки и транспортировки позволяет установить, что при значениях времени загрузки, разгрузки и транспортировки $\tau = 0,5$ обеспечивается обработка партии деталей в режиме реального времени, определенном величиной T . На рис. 51 приведен участок циклограммы, представляющей выполнение операций на участке ГАПС гальванопокрытий.

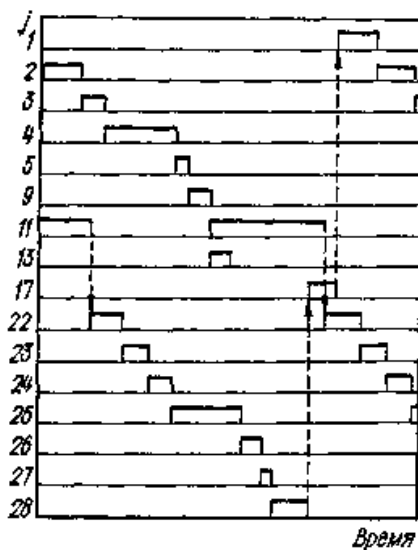


Рис. 51. График выполнения операций t_j , полученный интерпретацией управляющей модели на рис. 5 (участок циклограммы)

График получен программной интерпретацией управляющей модели (см. рис. 5). Аналогично строятся графики движения изделий и роботов, загрузки оборудования рабочих мест. Анализ этих показателей позволяет обосновать и усовершенствовать нормативы, повысить эффективность использования оборудования и трудовых ресурсов, применить к этому формализованному объекту методы математической оптимизации и учесть эвристические соображения оперативного персонала ГАПС.

Все перечисленные в примерах, а также другие возможные проектные расчеты обеспечиваются интерактивной операционной системой, которая спроектирована как диалоговая система моделирования. По мере освоения САПР пользователь может отказаться от высокой степени детализации диалога и использовать системы имитационного моделирования диалоговым файловым или подпрограммным способом.

Описание моделей для использования имитации выполняется не в виде программ, а структурами данных, формирование и входной контроль которых полностью отделены от моделирующих программ. Поскольку

разработка модели не требует программирования, а предписывает лишь подготовку данных, создаются условия для эффективного использования ЭВМ САПР и повышения производительности труда проектировщика.

Описав формальными методами требуемое поведение объекта управления и ОУВС, промоделировав и выполнив оптимизацию их работы и, наконец, убедившись в достаточном качестве алгоритмов, можно перейти к реализации управления. На этом этапе производят размещение входов и выходов объекта управления по адресному пространству устройств ввода-вывода ОУВС, а также настройку локальных программ обмена и управления. Таким образом устанавливается связь между действиями управляющей модели, управляемыми и информационными элементами объекта управления и ОУВС. Управление ГАПС организуется по схеме рис. 7 с использованием выбранной на этапе проектирования управляющей программы интегрированной системы проектирования и управления.

9.4.3. Средства САПР на базе микропроцессорной техники

Анализ основных задач САПР, а также единство цели, общность математических моделей систем проектирования и непосредственного управления, требования легкой перестраиваемости структуры, программной и информационной совместимости, высокого быстродействия и надежности в условиях автоматизированного человеко-машинного управления обуславливают необходимость разработки базового комплекта аппаратных и программных средств интерактивного взаимодействия, территориально приближенных к пользователю и обеспечивающих эффективное функционирование ГАПС в соответствии с заданным комплексным критерием качества. Эффективное решение задач управления технологическим оборудованием и вычислительными ресурсами системы управления предполагает использование как нескольких локальных процессоров небольшой мощности для распределенной (параллельной) обработки потока задач невысокой сложности, так и одного процессора большой мощности для централизованной обработки сложных задач. В общем случае возникает необходимость обеспечения многопользовательского режима решения задач выбора технологических структур при коллективном доступе к вычислительным и информационным ресурсам САПР. Тенденция к децентрализации системы управления выражается в рассредоточении вычислительной мощности и соответствующем распределении функций между несколькими процессорами. При этом

вычислительные средства, в виде микропроцессорных контроллеров, располагаются в непосредственной близости к объектам управления, т. е. на технологических установках, где собираются и обрабатываются все исходные данные, необходимые для цикла непосредственного управления. Территориально разнесенные вычислительные средства ГАПС взаимодействуют друг с другом по сети передачи данных и коллективно используют такие общие ресурсы системы, как программные средства, единые базы данных, средства внешней памяти на магнитных дисках, печатающие устройства, алфавитно-цифровые и графические дисплеи и другие периферийные устройства. Преимущества такой распределенной системы по сравнению с централизованной очевидны и заключаются в существенном уменьшении объемов данных, передаваемых от места их сбора к местам обработки, что снижает требования к скорости и пропускной способности каналов связи, следовательно, появляется возможность использования менее дорогой аппаратуры передачи данных, преимущественно последовательного типа. Также существенно уменьшаются количество и протяженность линий связи, что снижает затраты на кабельную продукцию и позволяет применение в таких системах волоконно-оптических линий связи. Поскольку в локальных подсистемах непосредственного управления технологическим оборудованием процессоры одновременно (параллельно) выполняют необходимые вычисления, то снижаются требования к их производительности, но при достаточно большом их количестве удается достигать результирующей производительности, эквивалентной применению дорогостоящей ЭВМ. Такой же эффект наблюдается и в отношении оперативной памяти системы, которая при соответствующей организации информационного обмена может быть доступна любому процессору. Обработка относительно небольшого объема информации в локальных подсистемах управления осуществляется, как правило, с использованием ограниченного набора алгоритмов, что существенно снижает требования к сложности применяемых операционных систем, структуре и организации прикладного программного обеспечения.

Расширение системы управления, изменение ее конфигурации и структуры путем включения в ее состав дополнительных недорогих подсистем экономически целесообразнее, чем переход на более мощную центральную машину. Упрощается также сама процедура наращивания технологического, программного и информационного обеспечения, создаются удобства для поэтапного ввода в эксплуатацию отдельных подсистем. Одним из наиболее важных преимуществ распределенной обработки и управления является повышенная

отказоустойчивость (живучесть), проявляющаяся в том, что отказ одного или даже нескольких автономных контроллеров, как правило, не приводит к полной потере работоспособности всей системы. Кроме того, поскольку коллективное использование ресурсов является естественным свойством распределенных систем, совпадающих с их сетевой математической моделью, то более простыми и эффективными становятся приемы резервирования и перераспределения функций отказавших устройств. В раз. 9.3.1 отмечалось, что объединение процессов проектирования и производства целесообразно поддержать интегрированием их технических и программных средств, баз данных и персонала.

Приложение

ПО для обучения

[Рекомендуемые бесплатные программы для обучения, решения задач, выполнения расчетов в Windows](#)

<u>WebMath.ru</u>	Сервис решения задач по математике, теории вероятности, геометрии, а также физике. Для решения задач не надо ничего устанавливать на свой компьютер, нужно только ввести данные, дальше программа все сделает сама! Webmath.ru создан для on-line помощи школьникам и студентам, нуждающимся в решении задач по алгебре, геометрии, теории вероятности, физике и другим предметам.
<u>Rebar</u>	Бесплатный калькулятор
<u>BlackBox Component Builde</u>	Бесплатная и открытая система программирования для

	Компонентного Паскаля, являющаяся вариантом системы Оберон, в общем аналогичная Турбо Паскалю, Дельфи и т.п., но обладающая удивительной комбинацией свойств.
<u>Pascal ABC</u>	Система Pascal ABC предназначена для обучения программированию на языке Паскаль и ориентирована на школьников и студентов младших курсов. По мнению авторов этой программы первоначальное обучение программированию должно проходить в достаточно простых и дружественных средах, в то же время эти среды должны быть близки к стандартным по возможностям языка программирования и иметь достаточно богатые и современные библиотеки

	стандартных подпрограмм.
Teach Book Lite	Система разработки электронных учебников высокого уровня. Совершенно не имея навыков программирования, пользователь сможет разрабатывать пособия с мощным интерфейсом и широкими мультимедийными возможностями.
Программа расчета на прочность стальных газопроводов	Программа расчета на прочность стальных газопроводов согласно СНиП 2.04.12-86 РАСЧЕТ НА ПРОЧНОСТЬ СТАЛЬНЫХ ТРУБОПРОВОДОВ.
Программы для расчетов по электронике	Здесь вы можете скачать бесплатные программы для расчетов по электронной тематике.
Транзисторный усилитель	Компьютерная модель для изучения принципов действия транзистора и его работы в усилительном каскаде. выполненном

<u>Mueller-dict</u>	Электронная версия популярного Англо-Русского <u>словаря</u> профессора Владимира Карловича Мюллера в формате DICT. Текущая редакция содержит более 50000 слов и представляет собой практически полную копию 7 издания вышеупомянутого словаря с некоторыми дополнениями и исправлениями, источником которых послужили более поздние типографские издания.
<u>Smart</u> <u>Virtual Teacher Background Panel</u>	Программа калькулятор предназначена для расчёта математических выражений (поддерживает сложные выражения такие как $(3+\sin(3-1)*4)*\exp(2)$). Virtual Teacher Background Panel это программа которая будет обучать вас языку всегда когда вы работаете за компьютером.

	Маленькая панель на вашем экране будет показывать вам иностранное слово и его перевод, вам останется только время от времени смотреть на нее и запоминать новые слова. Вы можете управлять цветом фона и текста, длительностью показа слов. И главное легко помечать уже выученные слова.
Explanatory NauLearning	С помощью этой программы можно быстро узнать значение слова используя имеющиеся в наличии словари. При вводе первых букв слова оно автоматически дополняется до ближайшего, найденного в словарях. Программа содержит словарь Ожегова, дополнительные словари могут быть скачаны с сайта программы. Система предназначена для разработки учебных

	курсов, проведения дистанционного обучения, управления учебным процессом и составления отчетности и может использоваться в качестве основной или вспомогательной среды обучения тренинг-центрами, консалтинговыми компаниями, корпоративными учебными центрами, HR-службами распределенных компаний, высшими и средними специальными учебными заведениями.
SunRay BookOffice	Пакет программ для создания и просмотра электронных книг и учебников. С помощью пакета можно создавать документацию в виде EXE файлов, CHM, HTML, PDF форматах, а так же в любых других (используя шаблоны).
TestTurn	Компактная и простая в применении бесплатная программа для

<u>PL Table</u>	проведения тестирования. Может быть использована для обучения и контроля знаний учебными заведениями или частными преподавателями, для аттестации сотрудников компаний, а так же в домашних условиях. Многофункциональная электронная реализация периодической системы элементов Менделеева.
<u>Adit Testdesk</u>	Пакет Adit Testdesk - это программный пакет для создания тестов, проведения тестирования и обработки полученных результатов. С помощью Adit Testdesk Вы можете проводить тестирования знаний учащихся в образовательных учреждениях, а также использовать программный комплекс для проведения психологического анализа и аттестации

<u>SunRay TestOfficePro</u>	или сертификации. Пакет SunRay TestOfficePro является комплексным решением для проведения тестирования в образовательных учреждениях и на предприятиях.
<u>ConverCalc</u>	Полезный технический инструмент. ConverCalc - приложение, созданное, чтобы помочь Вам в Вашей научной работе.
<u>Geometry Calculator</u>	Geometry Calculator позволит Вам быстро подсчитать площадь или объем относительно простых фигур.
<u>Avogadro</u>	Продвинутый молекулярный редактор, разработанный для использования поперечной платформы по вычислительной химии, молекулярного моделирования, биоинформатики, науки материалов. и

	связанных областей. Удобный интерфейс, прост в изучении.
JMathLib	Библиотека математических функций, разработанных, чтобы использоваться в оценке сложных выражений и показать результаты графически.
AnalyticMath	Мультиплатформенная программа с мощным редактором и и интегрированными особенностями 'auto- calc', которые помогут Вам легко и быстро разложить и визуально проанализировать математические выражения.
Электроснабжение	Программа «Конструктор дипломных проектов по электроснабжению» предназначена для выполнения дипломных проектов по электроснабжению. Она позволяет произвести расчет

	электроснабжения района, предприятия или же населенного пункта. На плане района возможно разместить до 290 трансформаторных подстанций, запитанных от РТП.
<u>ElectroDesigning</u>	Программа ElectroDesigning предназначена для проектирования электрической сети до 1000 вольт.
<u>Interactive Demos</u>	Интерактивные Демонстрации - комплект методических материалов предназначенный для виртуальных демонстраций ряда физических явлений, рассматриваемых в курсах оптики и атомной физики. Демонстрационные опыты можно реализовывать на мониторе компьютера, либо демонстрировать с помощью мультимедийного

	проектора на экране в аудитории.
SMath Studio	Бесплатный математический пакет с графическим интерфейсом для вычисления математических выражений и построения двумерных и трёхмерных графиков. Программа для вычисления математических выражений и построения сложных двумерных графиков.
FlatGraph	Программа предназначена для дифференцирования, упрощения и отображения функций в виде графиков.
Slu	ла для решения линейных уравнений. нахождение их членов системы ично, система имеет , проста в обращении, проверена временем. ри решении, когда их членов достаточно

<u>Algebry</u>	Программа по математике, которая решает квадратные уравнения, биквадратные уравнения, системы уравнений, складывает (вычитает) дроби, вычисляет корни любой степени и так далее.
<u>KLAVA</u>	Тренажер для обучения печатанию на клавиатуре слепым методом. Состоит из 4-х уроков на русскую раскладку клавиатуры и 3-х на английскую. Для достижения наилучшего результата рекомендуется проходить их по порядку. Работает без инсталляции
<u>English Trainer</u>	Программа - экзаменатор знания английского языка.
<u>FVords</u>	Программа для изучающих английский и немецкий: советы Longman. тесты.

	словники к подлинникам, параллельные тексты, режим суфлёра, настройка, поиск, печать, статистика и многое другое.
<u>ABC Simulator</u>	Игра-тренажер для запоминания английских слов. Уделяя 10-20 минут в день игре "ABC Simulator" Вы сможете запомнить более 5.000 английских слов и устойчивых словосочитаний.
<u>Teacher - Translator</u>	Программа - Учитель английского. С ней легко научиться разговаривать и переводить различные тексты. Инсталляции не требуется! При первом запуске программа проверяет и, если нужно, сама устанавливает компоненты текстово- речевого синтезатора. Чтобы увидеть перевод и услышать произношение слов.-

	достаточно просто вести указатель мышки от слова к слову.
SchoolBoy	Программа, позволяющая выполнять большое количество математических действий. Пригодится там, где необходимо проводить множественные расчеты (особенно последовательные или однотипные) с использованием сложных выражений. При этом допускается ввод сразу всего задания (состоящего из неограниченного числа подзаданий, разделяемых точкой с запятой) и решение его одним нажатием кнопки, после которого на экран будет выведен окончательный результат и все промежуточные."
Ассоциация	Программа для запоминания

	иностранных слов методом фонетических ассоциаций. Позволяет легко находить ассоциативные ключи к иностранным словам используя словарь из 90.000 слов русского языка.
Stamina	Программа для тех, кто хочет научиться набирать текст на клавиатуре всеми десятью пальцами. Поможет Stamina и тем, кто желает усовершенствовать навыки печати и ускорить скорость ввода, причем раскладка клавиатуры может быть разной - английской /русской /украинской.
Teach2000	Личный преподаватель, который поможет вам выучить иностранный язык, топографию и даже, если захотите историю Др. Рима в датах.
Gradebook Power	Программа Gradebook для учителей –

	разработанная учителями. Программа Gradebook Power создает более 40 отчетов учителей, студентов и администрации, посещаемость и графики мест.
<u>SpagoBI</u>	Система, позволяющая осуществлять многомерный <u>анализ</u> и формировать корпоративную отчетность высокого качества.
<u>iTALC</u>	Мощная программа для учителей работающих в компьютеризированных классах. Она позволяет различными способами контролировать компьютеры учащихся входящих в состав сети.
<u>Teacher Control Panel</u>	Позволяет учителю контролировать, блокировать, и управлять студенческими компьютерами и передавать изображение на своём экране им. Может работать с

	сочетанием рабочих мест на которых установлены ОС - Windows и linux.
Deductor	<u>Аналитическая платформа Deductor реализует практически все современные подходы к анализу структурированной табличной информации: хранилища данных (Data Warehouse), многомерный анализ (OLAP), добыча данных (Data Mining), обнаружение знаний в базах данных (Knowledge Discovery in Databases).</u>
ODE Программный комплекс МВТУ	Предметно ориентированная среда, предназначенная для решения и исследования решений обыкновенных дифференциальных уравнений. <u>Современная среда интеллектуальной</u>

<u>Microsoft Student Graphing Calculator</u>	<u>САПР.</u> Полноценный научный калькулятор с функциями графиков и решения уравнений. Его можно использовать точно также, как и обычный переносной калькулятор.
<u>TOptEls</u>	Визуальный компонент для отрисовки оптических элементов (линз и зеркал). Например, для программ на лазерную и оптическую тематику. Версия августа 2008 с двумя вариантами - для Delphi и Lazarus.
<u>Super Solver</u>	<u>ТОЭ Super Solver</u> <u>программа для</u> <u>студентов и</u> <u>школьников,</u> <u>изучающих физику,</u> <u>теоретические основы</u> <u>электротехники ТОЭ,</u> <u>ТЭЦ и другие</u> <u>предметы связанные с</u> <u>теорией</u> <u>электрических цепей.</u> <u>Комплекс</u> <u>существенно</u> <u>облегчает выполнение</u>

	<u>контрольных,</u> <u>самостоятельных,</u> <u>курсовых работ по</u> <u>вышеперечисленным</u> <u>дисциплинам.</u>
<u>Circuit Magic</u>	Комплекс расчета электрических цепей постоянного и переменного тока в общем виде.
<u>Advanced Grapher</u>	Программа для построения графиков и их анализа.
<u>Infinity</u>	Решение систем нелинейных обыкновенных дифференциальных уравнений.
<u>Mat JV</u>	Решение задач линейной алгебры.
<u>Юниор</u>	Универсальный Механизм - Юниор позволяет описывать плоские механические системы, а также моделировать и исследовать их поведение.
<u>FlatGraph</u>	Программа для дифференцирования и отображения функций в виде графиков.

<u>ADTester</u>	Пакет программ предназначенный для проведения тестирования.
<u>August</u>	Автоматизированная контролирующая система «AUGUST 4» предназначена для создания и использования тестов в контроле знаний в различных областях. «AUGUST 4» представляет собой интегрированный пакет, с помощью которого легко создавать тесты и проводить тестирование, как с использованием компьютера, так и без использования компьютера.
<u>Moodle</u>	Это программный продукт позволяющая создавать <u>курсы</u> и web-сайты, базирующиеся в internet. Это постоянно развивающийся проект, основанный на теории социального

iNstructor	конструктивизма. Инструмент для преподавателей, ведущих занятие в компьютерном классе. С помощью iNstructor преподаватель со своего рабочего места может в режиме реального времени наблюдать за действиями обучаемых, управлять их компьютерами, обмениваться с ними сообщениями.
Competentum Magister	<u>СДО Competentum Magister - полнофункциональная система для организации процесса обучения через Интернет или в локальной сети.</u>
ALGLIB DESIR	<u>Многоязыковая коллекция алгоритмов для решения проблем в области численного анализа и обработки данных.</u> <u>Пакет для исследования динамических систем</u>

Klavaro	Клавиатурный тренажер
Phun	Компьютерная игра-симулятор физики. Представляет собой графический анимационный редактор, основанный на технологии XML, который позволяет создавать объекты «на лету», которые сразу начинают подчиняться законам физики
MODELICA	Визуальная среда для моделирования сложных физических систем
Microsoft Mathematics Graphmatica	Специальный калькулятор для решения математических задач, визуализации двумерных и трехмерных графиков Мощная, простая в использовании программа схематизации уравнений с числовыми функциями и функциями

Convert	вычислений Свободная легкая и удобная в использовании программа которая конвертирует наиболее популярныя единицы длины, температуры, объемы, времени, скорости. массы, силы, энергии и др.
MathGV	Программа предназначена для построения двумерных (X, Y), трехмерных (X,Y,Z), параметрических и полярных графиков
NetSupport School Professional	NetSupport School Professional предназначена для организации работы преподавателя со студентами в компьютерном классе. Программа дает возможность преподавателю контролировать все действия студентов, при необходимости управлять их компьютерами

	удаленно, а также транслировать картинку со своего рабочего стола на мониторы учащихся.
Chemistry Add-in for Word	Позволяет упростить процедуру ввода химических формулы и изображений в документы Microsoft Word.
Article Authoring Add-in for Word	Позволяет вводить большое количество метаданных, относящихся к авторским материалам, при работе с документами Word. Умеет сохранять документы в формат NLM
Creative Commons Add-in for Microsoft Office	Позволяет добавить в документы Microsoft Office лицензию Creative Commons, которая генерируется на лету по заданным пользователем параметрам
NodeXL	<u>Мощный и легкий в использовании инструмент для визуализации и анализа больших сетей</u>
Zentity	<u>Платформа, предоставляющая</u>

<u>The Research Information Centre (RIC)</u>	<u>модули, утилиты и сервисы, которые могут помочь в создании и поддержке цифровой библиотеки (хранилища данных)</u> Это виртуальная исследовательская среда, созданная совместными усилиями группы Microsoft Research и Британской библиотеки.

Литература

1. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.1 Киев: «Освіта України», 2017. 464с.
2. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.2 Киев: «Освіта України», 2017. 564с.
3. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.3 Киев: «Освіта України», 2017. 424с.
4. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 1. Киев: «Освіта України», 2017. 730с
5. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 2. Киев: «Освіта України», 2017. 668 с
6. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.1. Киев: «Освіта України», 2017. 608 с
7. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.2. Киев: «Освіта України», 2017. 645 с
8. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.3. Киев: «Освіта України», 2017. 645 с
9. А.Е. Кононюк. Дискретно-непрерывная математика. Алгоритмы. Кн.10. Ч.1 Киев: «Освіта України», 2017. 588с
10. А.Е. Кононюк. Дискретно-непрерывная математика. Автоматы. Кн.11. Ч.1 Киев: «Освіта України», 2017. 658с
11. Дж. Ллойд. Системы тепловидения. Москва, 1978
12. А.Е. Кононюк. Общая теория познания и созидания. Кн.1. Киев: «Освіта України», 2013. 648 с. ecat.diit.edu.ua:81/ft/index_ru.html
13. А.Е. Кононюк. Общая теория познания и созидания. Кн.2, ч.1. Киев: «Освіта України», 2013. 544 с. ecat.diit.edu.ua:81/ft/index_ru.html
14. А.Е. Кононюк. Общая теория познания и созидания. Кн.2, ч.2. Киев: «Освіта України», 2013. 644 с. ecat.diit.edu.ua:81/ft/index_ru.html
15. А.Е. Кононюк. Информациология. Общая теория информации. Кн.1. Киев: «Освіта України», 2011. 476 с. ecat.diit.edu.ua:81/ft/index_ru.html
16. А.Е. Кононюк. Информациология. Общая теория информации. Кн.2. Киев: «Освіта України», 2011. 476 с. ecat.diit.edu.ua:81/ft/index_ru.html
17. А.Е. Кононюк. Информациология. Общая теория информации. Кн.3. Киев: «Освіта України», 2011. 412 с. ecat.diit.edu.ua:81/ft/index_ru.html
18. А.Е. Кононюк. Информациология. Общая теория информации. Кн.4. Киев: «Освіта України», 2011. 488 с. ecat.diit.edu.ua:81/ft/index_ru.html

19. А.Е. Кононюк. Общая теория понятий. Кн.1. Киев: «Освіта України», 2014. 514с.
20. А.Е. Кононюк. Общая теория понятий. Кн.2. Киев: «Освіта України», 2014. 544с.
21. А.Е. Кононюк. Общая теория понятий. Кн.3. Киев: «Освіта України», 2014. 614с.
22. А.Е. Кононюк. Системология. Общая теория систем. Кн.1. Киев: «Освіта України», 2012. 564с. ecat.diit.edu.ua:81/ft/index_ru.html
23. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.1. Киев: «Освіта України», 2014. 558с. ecat.diit.edu.ua:81/ft/index_ru.html
24. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.2. Киев: «Освіта України», 2014. 658с. ecat.diit.edu.ua:81/ft/index_ru.html
25. А.Е. Кононюк. Системология. Общая теория систем. Кн.2. Ч.1. Киев: «Освіта України», 2014. 558с. ecat.diit.edu.ua:81/ft/index_ru.html
26. А.Е. Кононюк. Общая теория распознавания. Кн.1. Киев: «Освіта України», 2012. 584 с. ecat.diit.edu.ua:81/ft/index_ru.html
27. А.Е. Кононюк. Общая теория распознавания. Кн.2. Киев: «Освіта України», 2012. 588 с. ecat.diit.edu.ua:81/ft/index_ru.html
28. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.1. Киев: «Освіта України», 2013. 448 с. ecat.diit.edu.ua:81/ft/index_ru.html
29. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.2. Киев: «Освіта України», 2013. 412 с. ecat.diit.edu.ua:81/ft/index_ru.html
30. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.3. Киев: «Освіта України», 2013. 520 с. ecat.diit.edu.ua:81/ft/index_ru.html
31. А.Е. Кононюк. Консалтология. Общая теория консалтинга. Кн.4. Киев: «Освіта України», 2013. 508 с. ecat.diit.edu.ua:81/ft/index_ru.html
32. А.Е. Кононюк. Дискретно-непрерывная математика. Начала. Кн.1. Киев: «Освіта України», 2012. 652с. ecat.diit.edu.ua:81/ft/index_ru.html
33. А.Е. Кононюк. Дискретно-непрерывная математика. Множества. Кн.2. Ч.1. Киев: «Освіта України», 2012. 452с. ecat.diit.edu.ua:81/ft/index_ru.html
34. А.Е. Кононюк. Дискретно-непрерывная математика. Множества. Кн.2. Ч.2. Киев: «Освіта України», 2013. 536 с. ecat.diit.edu.ua:81/ft/index_ru.html
35. А.Е. Кононюк. Дискретно-непрерывная математика. Отношения. Кн.3. Ч. 1. Киев: «Освіта України», 2013. 552с. ecat.diit.edu.ua:81/ft/index_ru.html

36. А.Е. Кононюк. Дискретно-непрерывная математика. Отношения. Кн.3. Ч. 2. Киев: «Освіта України», 2013. 548 с.
ecat.diit.edu.ua:81/ft/index_ru.html
37. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.1. Киев: «Освіта України», 2011. 452с.
ecat.diit.edu.ua:81/ft/index_ru.html
38. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.2. Киев: «Освіта України», 2011. 668 с.
ecat.diit.edu.ua:81/ft/index_ru.html
39. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.3. Киев: «Освіта України», 2015. 488 с.
http://lib.sumdu.edu.ua/library/DocDescription?doc_id=640902
40. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.4. Киев: «Освіта України», 2015. 548 с.
41. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.5. Киев: «Освіта України», 2015. 528 с.
42. А.Е. Кононюк. Дискретно-непрерывная математика. Алгебры. Кн.4. Ч.6. Киев: «Освіта України», 2015. 608 с.
43. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.1. Киев: «Освіта України», 2013. 612 с.
ecat.diit.edu.ua:81/ft/index_ru.html
44. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.2. Киев: «Освіта України», 2013. 500 с.
ecat.diit.edu.ua:81/ft/index_ru.html
45. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.3. Киев: «Освіта України», 2013. 520 с.
ecat.diit.edu.ua:81/ft/index_ru.html
46. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.4. Киев: «Освіта України», 2013. 508 с.
ecat.diit.edu.ua:81/ft/index_ru.html
47. А.Е. Кононюк. Дискретно-непрерывная математика. Матрицы. Кн.5. Ч.5. Киев: «Освіта України», 2013. 672 с.
ecat.diit.edu.ua:81/ft/index_ru.html

48. А.Е. Кононюк. Дискретно-непрерывная математика. Поверхности. Кн.6. Ч.1. Киев: «Освіта України», 2012. 652с.
ecat.diit.edu.ua:81/ft/index_ru.html
49. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.1 Киев: «Освіта України», 2014. 652с.
ecat.diit.edu.ua:81/ft/index_ru.html
50. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.2 Киев: «Освіта України», 2014. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html
51. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.3 Киев: «Освіта України», 2015. 512с.
ecat.diit.edu.ua:81/ft/index_ru.html
52. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.4 Киев: «Освіта України», 2015. 552с.
ecat.diit.edu.ua:81/ft/index_ru.html
53. А.Е. Кононюк. Дискретно-непрерывная математика. Графы. Кн.7. Ч.5 Киев: «Освіта України», 2015. 660с.
54. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.1 Киев: «Освіта України», 2012. 602с. ecat.diit.edu.ua:81/ft/index_ru.html
55. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.2 Киев: «Освіта України», 2012. 708с. ecat.diit.edu.ua:81/ft/index_ru.html
56. А.Е. Кононюк. Обобщенная теория моделирования. Кн.1. Ч.3 Киев: «Освіта України», 2012. 568с. ecat.diit.edu.ua:81/ft/index_ru.html
57. А.Е. Кононюк. Обобщенная теория моделирования. Кн.2. Киев: «Освіта України», 2012. 548с. ecat.diit.edu.ua:81/ft/index_ru.html
58. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.1 Киев: «Освіта України», 2012. 636с. ecat.diit.edu.ua:81/ft/index_ru.html
59. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.2 Киев: «Освіта України», 2012. 448с. ecat.diit.edu.ua:81/ft/index_ru.html
60. А.Е. Кононюк. Обобщенная теория моделирования. Кн.3. Ч.3 Киев: «Освіта України», 2013. 588с. ecat.diit.edu.ua:81/ft/index_ru.html
61. А.Е. Кононюк. Основы теории оптимизации. Кн.1. Киев: «Освіта України», 2011. 602с. ecat.diit.edu.ua:81/ft/index_ru.html
62. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.1. Киев: «Освіта України», 2011. 552с. ecat.diit.edu.ua:81/ft/index_ru.html
63. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.2. Киев: «Освіта України», 2011. 616с. ecat.diit.edu.ua:81/ft/index_ru.html
64. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.3. Киев: «Освіта України», 2012. 456с. ecat.diit.edu.ua:81/ft/index_ru.html

65. А.Е. Кононюк. Основы теории оптимизации. Кн.2. Ч.4. Киев: «Освіта України», 2012. 512с. ecat.diit.edu.ua:81/ft/index_ru.html
66. А.Е. Кононюк. Основы научных исследований. Кн.1. Киев: «Освіта України», 2011. 508с. ecat.diit.edu.ua:81/ft/index_ru.html
67. А.Е. Кононюк. Основы научных исследований. Кн.2. Киев: «Освіта України», 2011. 452с. ecat.diit.edu.ua:81/ft/index_ru.html
68. А.Е. Кононюк. Основы научных исследований. Кн.3. Киев: «Освіта України», 2011. 456с. ecat.diit.edu.ua:81/ft/index_ru.html
69. А.Е. Кононюк. Основы научных исследований. Кн.4. Киев: «Освіта України», 2011. 456с. ecat.diit.edu.ua:81/ft/index_ru.html
70. А.Е. Кононюк. Общая теория коммуникаций. Кн.1. Киев: «Освіта України», 2014. 488с.
71. А.Е. Кононюк. Нейроні мережі і генетичні алгоритми. Киев: «Корнійчук», 2010. 448с. ecat.diit.edu.ua:81/ft/index_ru.html
72. Кононюк А. Е. Обобщенная теория познания и созидания. [В 2 кн.] Кн. 1 : Начала / А. Е. Кононюк. — Киев : Освіта України, 2013. ecat.diit.edu.ua:81/ft/index_ru.html
73. Кононюк А. Е. Обобщенная теория познания и созидания. [В 2 кн.] Кн. 2 : Теория познания. Ч. 1 / А. Е. Кононюк. — Киев : Освіта України, 2013 ecat.diit.edu.ua:81/ft/index_ru.html
74. Кононюк А. Ю. Вища математика. (Модульна технологія навчання) : навчальний посібник : в 2 кн. / А. Ю. Кононюк. — Київ : КНТ, 2009 — Кн. 1. — 2009. — 702 с. ecat.diit.edu.ua:81/ft/index_ru.html
75. Кононюк А. Ю. Вища математика. (Модульна технологія навчання) : навчальний посібник : в 2 кн. / А. Ю. Кононюк. — Київ : КНТ, 2009 Кн. 2. — 2009. — 790 с. ecat.diit.edu.ua:81/ft/index_ru.html
76. А.Е. Кононюк. Дискретно-непрерывная математика. Поверхности. Кн.6. Ч.2. Киев: «Освіта України», 2012. 652с. <http://www.dut.edu.ua/ua/lib/127/category/96/view/1297>

77. А.Е. Кононюк. Дискретно-непрерывная математика. Пространства. Кн.8. Ч.1. Киев: «Освіта України», 2016. 748 с.

<http://www.dut.edu.ua/ua/lib/1/category/96/view/1439>

78. А.Е. Кононюк. Дискретно-непрерывная математика. Пространства. Кн.8. Ч.2. Киев: «Освіта України», 2016. 480с.

http://lib.sumdu.edu.ua/library/DocDescription?doc_id=640775

79. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.1. Киев: «Освіта України», 2016. 568с.

80. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.2. Киев: «Освіта України», 2016. 558с.

81. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.3. Киев: «Освіта України», 2016. 588с.

82. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.4. Киев: «Освіта України», 2016. 552с

83. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.5. Киев: «Освіта України», 2016. 836 с

84. А.Е. Кононюк. Истины и информация (фундаментальная теория представления истин и информации). К.6. Киев: «Освіта України», 2016. 576 с

85. Алберг Дж., Нильсон Э., Уолш Дж. Теория сплайнов и ее приложения.—М.: Мир, 1972 (Ahlberg J.H., Nilson E.N., Walsh J.L. The Theory of Splines and their Applications.—New York: Academic Press, 1967).

86. Березин И.С., Жидков Н.П. Методы вычислений. Т. 1,2.—М.: Наука, 1962, 1966.

87. Гардан П., Люка М. Машинная графика и автоматизация конструирования.— М.: Мир, 1987 (Techniques Graphiques Interactives et C.A.O./par Michel Lucas et Yvon Gar dan.—Prance: Hermes Publishing, 1983).

88. Гильберт Д., Кон-Фоссен С. Наглядная геометрия.—М.: Наука, 1981 (Hilbert D., Cohn-Vossen S. Anschauliche Geometrie.—Berlin: 1932).
89. Де Бор К. Практическое руководство по сплайнам.—М.: Радио и связь, 1985 (De Boor C. A Practical Guide to Splines.—Berlin: Springer, 1879).
90. Демидович Б.П., Марон И.А. Основы вычислительной математики.—М.: Наука, 1970.
91. Дубровин Б.А., Новиков С.П., Фоменко А.Т. Современная геометрия.— М.: Наука, 1986.
92. Ильин В. А., Позняк Э.Г. Аналитическая геометрия.—М.: Наука, 1981.
93. Ильин В. А., Позняк Э.Г. Линейная алгебра.—М.: Наука, 1984.
94. Калиткин Н.Н. Численные методы.—М.: Наука, 1978.
95. Ласло М. Вычислительная геометрия и компьютерная графика на C++ К—М.: Бином, 1997 (Laszlo M. J. Computational Geometry and Computer Graphics in C++.—Prentice Hall, 1996).
96. Препарата Ф., Шеймос М. Вычислительная геометрия.—М.: Мир, 1989 (Preparata F. P., Sham os M. Computational Geometry: An Introduction.— New York, Berlin, Tokyo: Springer-Verlag, 1985).
97. Роджерс Д., Адамс Дж. Математические основы машинной графики.—М.: Машиностроение, 1980 (Rogers D.F., Adams J. A. Mathematical Elements for Computer Graphics.—McGraw-Hill, 1976).
98. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.1 Киев: «Освіта України», 2017. 464с.
99. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.2 Киев: «Освіта України», 2017. 564с.
100. А.Е. Кононюк. Дискретно-непрерывная математика. Математическая логика. Кн.9. Ч.3 Киев: «Освіта України», 2017. 424с.
101. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 1. Киев: «Освіта України», 2017. 730с

102. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 2. Киев: «Освіта України», 2017. 668 с
103. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.1. Киев: «Освіта України», 2017. 608 с
104. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.2. Киев: «Освіта України», 2017. 645 с
105. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.3. Киев: «Освіта України», 2017. 645 с
106. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.4. Киев: «Освіта України», 2017. 480с
107. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 3, ч.5. Киев: «Освіта України», 2017. 480с
108. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 4. Киев: «Освіта України», 2017. 790с
109. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 5. Киев: «Освіта України», 2017. 790с
110. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 6. Киев: «Освіта України», 2018. 660с
111. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 7, ч.1. Киев: «Освіта України», 2017. 645 с
112. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 7, ч.2 (начало). Киев: «Освіта України», 2018. 645 с
113. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 7, ч.2 (окончание). Киев: «Освіта України», 2018. 544 с
114. А.Е. Кононюк. Основы фундаментальной теории искусственного интеллекта. Кн. 8. Киев: «Освіта України», 2018. 562 с
115. Кононюк А. Е. Обобщенная теория моделирования. К.З. Ч. 2 (начало) К.4: "Освіта України", 2018. - 522 с.
116. Кононюк А. Е. Обобщенная теория моделирования. К.З. Ч. 2 (окончание) К: "Освіта України", 2018. - 562 с.

- [↑ «Катастрофа облачных вычислений» в 2010 г?–](#)