ВЫСШЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

В. Б. УТКИН, К. В. БАЛДИН

ИНФОРМАЦИОННЫЕ СИСТЕМЫ В ЭКОНОМИКЕ

Рекомендовано
Учебно-методическим объединением по образованию
в области прикладной информатики в качестве учебника
для студентов высших учебных заведений, обучающихся
по специальности 351400 «Прикладная информатика» (по областям)
и другим междисциплинарным специальностям

Москва
ACADEM'A
2004

УДК 681.518(075.8)
ББК 65.050.2я73
У84

Р е ц е н з е н т ы :
профессор кафедры производственного менеджмента МАТИ —РГТУ им. К. Э. Циолковского, канд. техн. наук *В. А. Волочиенко;*
зав. кафедры оценки и управления собственностью Государственного университета управления, д-р экон. наук, профессор *В.И.Бусов*

Уткин В. Б.
У 84 Информационные системы в экономике: Учебник для студ. высш. учеб, заведений / В.Б. Уткин, К.В. Балдин. — М.: Издательский центр «Академия», 2004. — 288 с.

ISBN 5-7695-1447-7

Содержится систематизированное изложение теоретических основ современных информационных систем в области экономики. Основное внимание уделено методологическим основам применения средств автоматизации профессиональной деятельности, теории и практике моделирования экономических информационных систем, а также основам построения и использования систем искусственного интеллекта.

Для студентов высших учебных заведений, обучающихся по специальности 351400 «Прикладная математика» (по областям) и другим междисциплинарным специальностям.

УДК 681.518(075.8)
ББК 65.050.2я73

ISBN 5-7695-1447-7

ПРЕДИСЛОВИЕ

Решение проблем рационального использования современных и перспективных методов и средств обработки информации в практической (профессиональной) деятельности людей приобретает первостепенное значение. Это обусловлено рядом причин. Во-первых, таковы актуальные потребности общества, связанные с необходимостью решения все более усложняющихся политических, экономических, военных и других проблем различного масштаба (глобальных, региональных, государственных, национальных и т.п.). Во-вторых, это единственный путь значительного (а в ряде случаев — кардинального) повышения эффективности профессиональной деятельности человека. В-третьих, широкое распространение получили технические и программные средства, позволяющие реализовать новые технологии при приемлемом расходовании ресурсов. Наконец, пользователями этих технологий становится все большее число людей (по некоторым оценкам к пользователям компьютерных технологий во многих странах может быть отнесено все трудоспособное население).

Естественно, что такой сложный и многообразный процесс, как информатизация, нуждается в методологическом обосновании, являющемся результатом исследований в рамках научно-технического направления и науки, получивших название «информатика».

В широком смысле под информатикой понимается научно-техническое направление, охватывающее все аспекты разработки, проектирования, создания и функционирования систем обработки информации на базе электронно-вычислительных машин (ЭВМ), их применения и воздействия на различные области социальной практики.

Под информатикой в узком смысле понимается научная дисциплина, изучающая цели, способы и средства автоматизации человеческой деятельности на базе современных средств электронно-вычислительной техники (ЭВТ) и связи при решении практических задач, связанных с накоплением, передачей, обработкой и представлением информации.

Предметом изучения информатики являются информационные технологии, которые реализуются на практике в автоматизированных информационных системах (АИС) различного назначения, выступающих в качестве объекта информатики. Таким образом, АИС позволяют автоматизировать ту или иную сферу профессиональной деятельности людей за счет использования компьютерных средств и технологий. Иными словами, в качестве основных средств (инструмента) автоматизации профессиональной деятельности людей сегодня выступают средства ЭВТ и связи.

В учебнике рассматриваются вопросы автоматизации таких важных видов профессиональной деятельности экономиста, как управление и научно-исследовательская работа. Применение совершенных технических и программных средств, реализующих современные и перспективные математические методы, в том числе с использованием достижений теории искусственного интеллекта (ИИ), именно в этих областях позволяет в ряде случаев говорить об интеллектуализации профессиональной деятельности как важнейшем этапе ее автоматизации.

Учебник состоит из четырех частей. В первой части изложены методологические основы создания и применения современных компьютерных систем и технологий в экономической практике: основные определения, классификация АИС, требования к специальному программному обеспечению и принципы его разработки, методика проведения информационного обследования объекта

автоматизации, современные технологии разработки АИС (в частности, CASE-технологии).

Вторая часть содержит методические основы проектирования и использования баз (банков) данных и современных компьютерных сетей, а также организации процессов обработки данных в базе данных (БД).

В третьей части изложены вопросы, связанные с теорией и практикой моделирования сложных экономических систем. Особое внимание уделено имитационным моделям экономических информационных систем и методам учета различных случайных факторов при исследовании эффективности экономических операций.

Четвертая часть посвящена методическим основам построения и использования в деятельности специалистов систем искусственного интеллекта (СИИ), прежде всего экспертных систем с различными методами представления знаний. Подробно рассмотрены механизмы логического вывода в продукционных экспертных системах и диагностических экспертных системах байесовского типа.

Учебник подготовлен при государственной поддержке ведущих научных школ и получил гранд № НШ — 2350.2003.9.

Авторы выражают глубокую благодарность профессору В. А. Волочиенко и профессору В.И.Бусову за рецензирование рукописи и сделанные ими замечания.

СПИСОК СОКРАЩЕНИЙ

АА	—	автоматизированные архивы
АИВС	—	автоматизированная информационно-вычислительная система
АИС	—	автоматизированная информационная система
АИСС	—	автоматизированные информационно-справочные системы
АС	—	автоматизированные системы
АСВЭКМ	—	автоматизированные системы ведения электронных карт
АСД	—	автоматизированные системы делопроизводства
АСО	—	автоматизированная система обучения
АСОДИ	—	автоматизированная система обучения деловым играм
АСПО	—	автоматизированная система программированного обучения
АСУ	—	автоматизированная система управления
БД	—	база данных
ВС	—	вычислительная система
ВЦ	—	вычислительный центр
ДСВ	—	дискретная случайная величина
ЖЦ	—	жизненный цикл
ЖЦПО	—	жизненный цикл программного обеспечения
ИЗ	—	информационная задача
ИИ	—	искусственный интеллект
ИППП	—	интеллектуальные пакеты прикладных программ
ИРЗ	—	информационные и расчетные задачи
ИРС	—	информационно-расчетная система
ИС	—	информационная система
ИТ	—	информационная технология
КУ	—	конфигурационное управление
ЛВС	—	локальная вычислительная сеть
ММ	—	математическая модель
МЦ	—	моделирующий центр
НСД	—	несанкционированный доступ
ОПО	—	общее программное обеспечение
ОППО	—	общее прикладное программное обеспечение
ОС	—	операционная система
ОСПО	—	общее системное программное обеспечение
ПО	—	программное обеспечение
ПОИС	—	проблемно-ориентированная информационная система
ППП	—	пакеты прикладных программ

ПСЧ — псевдослучайные числа
РБД — реляционная база данных
РД — руководящий документ
РЗ — расчетная задача
РПС — разрушающие программные средства
САПР — система автоматизированного проектирования
СВ — случайная величина
СВТ — средства вычислительной техники
СИИ — система искусственного интеллекта
СОН — система общего назначения
СП — система программирования
СПО — специальное программное обеспечение
СППО — специальное прикладное программное обеспечение
СППР — система поддержки принятия решения
СС — специализированная система
ССПО — специальное системное программное обеспечение
СУБД — система управления базами данных
ТЗ — техническая задача
ТТК — тренажеры и тренажерные классы
УЗ — уязвимость защиты
УМ — управляющий модуль
ФД — функциональные действия
ФМ — функциональный модуль
ЭВМ — электронно-вычислительная машина
ЭВТ — электронно-вычислительная техника
ЭИС — экономическая информационная система
ЭС — экспертная система
ЯИМ — язык имитационного моделирования
ЯОН — язык общего назначения

РАЗДЕЛ I. МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ И ПРИМЕНЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Глава 1. АВТОМАТИЗИРОВАННЫЕ ЭКОНОМИЧЕСКИЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ИХ ЭЛЕМЕНТЫ

1.1. Основные понятия и определения

Будучи достаточно сложным процессом, автоматизация любой деятельности человека при решении практических задач должна иметь научное — прежде всего методологическое — обеспечение. Как уже было отмечено в предисловии, наукой, изучающей наиболее общие закономерности внедрения средств автоматизации (компьютеризации) во все сферы жизни общества и последствия этого, является информатика. В рамках этой научной дисциплины автоматизация профессиональной деятельности определяется как процесс создания, внедрения и использования технических, программных средств и математических методов, освобождающих человека от непосредственного участия в получении, преобразовании и передаче энергии, материалов и (или) информации в профессиональной деятельности. Основные виды автоматизируемой профессиональной деятельности: производственные процессы, проектирование, обучение, научные исследования, управление. Основу автоматизации профессиональной деятельности в современных условиях составляют средства ЭВТ и связи.

Весьма важными и особенно интересными для широкого круга специалистов в области организационного управления представляются особенности автоматизации управленческой деятельности как процесса создания, внедрения и использования технических, программных средств и математических методов, предназначенных для автоматизированного сбора, хранения, поиска, переработки и передачи информации, используемой при управлении эргатическими системами, в ходе реализации новых информационных технологий управления. Целью автоматизации управленческой

деятельности является повышение эффективности управления (качества управленческих решений, оперативности, повышения производительности управленческого труда и т.д.) [2].

Информатика является одной из отраслей общей (теоретической) информатики и изучает цели, способы и средства автоматизации деятельности должностных лиц на базе ЭВТ при управлении персоналом, разработке новых систем оружия, совершенствовании видов, форм и способов боевых действий, обучении личного состава.

Как и всякая другая научная дисциплина, информатика имеет свой объект и предмет.

В качестве *объекта информатики* выступает АИС, представляющая собой совокупность технических, программных средств и организационных мероприятий, предназначенных для автоматизации информационных процессов в профессиональной деятельности. Основным техническим средством АИС является ЭВМ.

Используя термин «информация», мы, как правило, не задумываемся о том, что это такое. Надо отметить, что вопрос этот является достаточно сложным (подробнее см. гл. 5). До настоящего времени в науке не выработано строгого определения понятия «информация». Говоря об информационных процессах в АИС, мы пока будем понимать под *информацией* некоторую совокупность данных (текстовых, числовых, графических) и связей между ними.

Под переработкой информации понимаются все возможные информационные процессы, сопровождающие профессиональную деятельность: сбор, хранение, поиск, представление информации на определенном носителе в определенном виде (визуальном, графическом, текстовом, звуковом), получение новой информации (например, в результате проведения расчетов), передача информации по каналам связи различным адресатам и др.

Автоматизированная информационная система должна рассматриваться как инструмент в руках должностных лиц, реализующих переработку информации в процессе профессиональной деятельности. Можно сказать, что наличие этого инструмента фактически определяет новую технологию осуществления профессиональной деятельности.

Понятие *«технология»* означает комплекс знаний о способах, приемах труда, наборах материально-технических факторов, способах их соединения для создания какого-либо продукта или услуги. Применительно к промышленному производству используется понятие «производственная индустриальная технология».

Применение понятия «технология» к информационным процессам привело к возникновению понятия *«информационная технология»* — совокупность знаний о способах автоматизированной переработки информации с использованием ЭВМ для автоматизации управленческой деятельности.

Создание новых информационных технологий и внедрение их в профессиональную деятельность является одной из основных задач информатики. Именно поэтому в качестве *предмета информатики* целесообразно рассматривать информационные технологии, определяющие рациональные способы разработки и применения АИС.

Каждая АИС обеспечивает реализацию некоторой информационной технологии переработки информации в процессе профессиональной деятельности. Таким образом, в качестве *задач информатики* можно рассматривать создание новых информационных технологий и реализующих их АИС или перенесение известных информационных технологий из одной области человеческой деятельности в другую.

1.2. Автоматизированные информационные системы и их классификация

В качестве основного классификационного признака АИС целесообразно рассматривать особенности автоматизируемой профессиональной деятельности — процесса переработки входной информации для получения требуемой выходной информации, в котором АИС выступает в качестве инструмента должностного лица или группы должностных лиц, участвующих в управлении организационной системой [54].

В соответствии с предложенным классификационным признаком можно выделить следующие классы АИС (рис. 1.1):

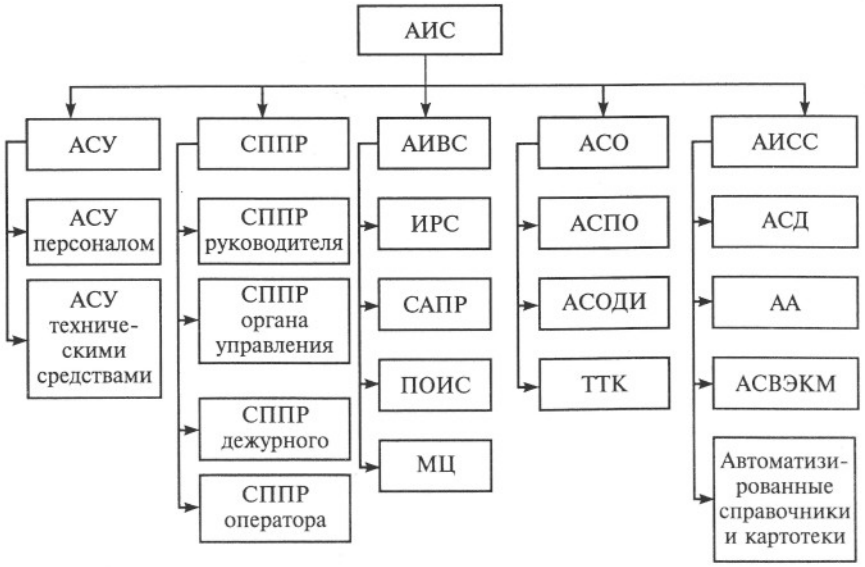


Рис. 1.1. Классификация АИС

- автоматизированные системы управления (АСУ);
- системы поддержки принятия решения (СППР);
- автоматизированные информационно-вычислительные системы (АИВС);
- автоматизированные системы обучения (АСО);
- автоматизированные информационно-справочные системы (АИСС).

Рассмотрим особенности каждого класса АИС и характеристики возможных видов АИС в составе каждого класса.

Автоматизированные системы управления. АСУ представляет собой АИС, предназначенную для автоматизации всех или большинства задач управления, решаемых коллективным органом управления (министерством, дирекцией, правлением, службой, группой управления и т.д.). В зависимости от объекта управления различают АСУ персоналом и АСУ техническими средствами. АСУ является организационной и технической основой реализации рациональной технологии коллективного решения задач управления в различных условиях обстановки. В этой связи разработка рациональной технологии организационного управления является определяющим этапом создания любой АСУ.

АСУ персоналом обеспечивает автоматизированную переработку информации, необходимой для управления организацией в повседневной деятельности, а также при подготовке и реализации программ развития.

АСУ техническими средствами предназначены для реализации соответствующих технологических процессов. Они являются, по сути, передаточным звеном между должностными лицами, осуществляющими управление техническими системами, и самими техническими системами.

В настоящее время АСУ техническими средствами нашли широкое распространение во всех развитых государствах. Объясняется это тем, что управление существующими новейшими технологическими процессами без применения АСУ техническими средствами становится практически невозможным. Что касается АСУ персоналом, то в настоящее время такие системы широко используются в странах Запада, и непрерывно ведутся работы по созданию новых систем, в том числе на базе достижений в области ИИ.

Системы поддержки принятия решений. СППР являются достаточно новым классом АИС, теория создания которых в настоящее время интенсивно развивается.

СППР называется АИС, предназначенная для автоматизации деятельности конкретных должностных лиц при выполнении ими своих должностных (функциональных) обязанностей в процессе управления персоналом и (или) техническими средствами.

Выделяются четыре категории должностных лиц, деятельность которых отличается различной спецификой переработки информации: руководитель, должностное лицо аппарата управления, оперативный дежурный, оператор. В соответствии с четырьмя категориями должностных лиц различают и четыре вида СППР: СППР руководителя, СППР органа управления, СППР дежурного и СППР оператора.

Рассмотрим специфику деятельности должностных лиц, относящихся к каждой выделенной категории.

К категории *руководитель* относятся должностные лица, на которых возложено управление подчиненными должностными лицами (подразделениями) и принятие решений в процессе руководства. Основная форма деятельности руководителя — деловое общение.

Деятельность руководителей характеризуется следующими особенностями:

- при централизации принятия решений резко возрастают объемы информации, уменьшается время на обдумывание и анализ, растут сложности комплексного учета всех факторов;
- велика доля текущих задач, не позволяющих сосредоточиться на стратегических целях;
- в процессе деятельности преобладают приемы, обусловленные привычками, опытом, традициями и другими неформализуемыми обстоятельствами;
- при принятии решения руководитель не всегда в состоянии описать и даже представить достаточно полную умозрительную модель ситуации, а вынужден использовать лишь некоторое представление о ней;
- деятельность руководителя в значительной мере зависит от темперамента и стиля деятельности, от степени знаний причин и следствий, ясности представления взаимосвязей, объема имеющейся информации.

Перечисленные особенности деятельности руководителей обусловливают крайнюю сложность автоматизации их деятельности, которая содержит большое количество неформальных элементов, прежде всего таких, как оперативное и стратегическое управление, принятие решений. Исходя из особенностей деятельности руководителя, можно сформулировать основные требования, предъявляемые к СППР руководителя:

1) наличие широкой информационной базы с возможностью оперативного поиска требуемой информации;

2) наглядность представления информации в форме, адаптированной к запросам конкретного должностного лица (текста, таблиц, графиков, диаграмм и т.д.);

3) обеспечение оперативной связи с другими источниками информации в системе управления и особенно с непосредственными помощниками;

4) наличие диалоговых программных средств обеспечения принятия решений на основе формальных (математических) методов;

5) простота работы при повышенной надежности технических и программных средств;

6) обеспечение возможности накопления в памяти ЭВМ опыта и знаний (в рамках интеллектуальных СППР).

Необходимо отметить, что пп. 2, 3 и 5 являются универсальными и относятся ко всем видам СППР. В настоящее время требования 1, 2, 3 и 5 могут быть полностью удовлетворены с использованием известных информационных технологий. Что касается требований 4 и 6, то их удовлетворение составляет основную теоретическую проблему, возникающую при создании СППР руководителя.

К категории *должностное лицо аппарата управления* относятся специалисты, занимающиеся аналитической работой по подготовке решений руководителя и их документальным оформлением. Основу деятельности должностных лиц аппарата управления составляет оценка различных вариантов решения (проведение оценочных расчетов) и разработка проектов различных документов.

Эффективность функционирования аппарата управления во многом определяется продуктивностью деятельности специалистов, особенно в вопросах создания новой информации. Доля творческого труда в их работе достаточно высока. Именно эти специалисты обеспечивают практически всю информационную подготовку для принятия решения руководителем. Они являются основными исполнителями документов, определяя их качество.

СППР органа управления должна прежде всего создать должностным лицам условия для плодотворного ведения аналитической работы и сведения к минимуму доли рутинных работ (поиск информации, оформление документов, проведение оперативных расчетов и т.д.).

Особенности деятельности должностных лиц аппарата управления определяют следующие основные требования к СППР органа управления:

- обеспечение оперативного поиска и отображения всей информации, необходимой для подготовки решений и формирования проектов документов в пределах его компетентности;
- обеспечение возможности ведения оперативных расчетов и моделирования для оценки ситуации и подготовки вариантов решений;

- обеспечение возможности автоматизированной подготовки проектов документов (текстов, графиков, диаграмм и т.п.).

К основным элементам СППР органа управления следует отнести средства ведения оперативных расчетов и моделирования, поскольку именно эти средства в наибольшей степени обеспечивают повышение эффективности и качества управления.

К категории *оперативный дежурный* относятся должностные лица, выполняющие обязанности по оперативному руководству организационной системой во время дежурства на соответствующих пунктах управления в течение определенного времени.

Основными особенностями деятельности оперативных дежурных являются:

- относительно узкий круг решаемых задач;
- жесткая регламентация деятельности в большинстве вариантов складывающейся обстановки;
- жесткий лимит времени на принятие решений и выполнение различных операций.

Перечисленные особенности деятельности оперативных дежурных определяют в качестве основных требований к СППР дежурного обеспечение оперативного предоставления информации, необходимой оперативному дежурному в заранее определенных ситуациях, а также оперативного анализа складывающейся ситуации. Последнее требование может быть достигнуто с использованием технологии экспертных систем.

К категории *оператор* могут быть отнесены должностные лица, выполняющие техническую работу по заранее определенному алгоритму. Основная особенность деятельности оператора — отсутствие необходимости принимать сложные решения в процессе своей деятельности. СППР оператора должна обеспечивать возможность работы должностного лица со справочной информацией и возможность автоматизированной подготовки документов.

Автоматизированные информационно-вычислительные системы. АИВС предназначены для решения сложных в математическом отношении задач, требующих больших объемов самой разнообразной информации. Таким образом, видом деятельности, автоматизируемым АИВС, является проведение различных (сложных и «объемных») расчетов. Эти системы используются для обеспечения научных исследований и разработок, а также как подсистемы АСУ и СППР в тех случаях, когда выработка управленческих решений должна опираться на сложные вычисления.

В зависимости от специфики области деятельности, в которой используются АИВС, различают следующие виды этих систем.

Информационно-расчетные системы (ИРС). Эти системы предназначены для обеспечения оперативных расчетов и автоматизации обмена информацией между рабочими местами в пределах некоторой организации или системы организаций. ИРС обычно сопрягаются с АСУ и в рамках последней могут рассматриваться как подсистема.

Технической базой ИРС являются, как правило, сети больших, малых и микро-ЭВМ. ИРС имеют сетевую структуру и могут охватывать несколько десятков и даже сотен рабочих мест различных уровней иерархии. Основной сложностью при создании ИРС является обеспечение высокой оперативности расчетов и обмена информации в системе при строгом разграничении доступа должностных лиц к служебной информации.

Системы автоматизации проектирования (САПР). Эти системы предназначены для автоматизации деятельности подразделений проектной организации или коллектива специалистов в процессе разработки проектов изделий на основе применения единой информационной базы, математических и графических моделей, автоматизированных проектных и конструкторских процедур. САПР является одной из систем интегральной автоматизации производства, обеспечивающих реализацию автоматизированного цикла создания нового изделия от предпроектных научных исследований до выпуска серийного образца.

В области экономики САПР могут использоваться при проектировании экономических информационных систем и их элементов. Кроме того, технология САПР может обеспечить создание автоматизированной системы отображения обстановки на экране в процессе ведения экономических операций или в ходе деловых игр различных типов.

Проблемно-ориентированные имитационные системы (ПОИС). Эти системы предназначены для автоматизации разработки имитационных моделей в некоторой предметной области [39]. Например, если в качестве предметной области взять развитие автомобилестроения, то любая модель, создаваемая в этой предметной области, может включать стандартные блоки, моделирующие деятельность предприятий, поставляющих комплектующие; собственно сборочные производства; сбыт,

обслуживание и ремонт автомобилей; рекламу и др. Эти стандартные блоки могут строиться с различной детализацией моделируемых процессов и различной оперативностью расчетов. Пользователь, работая с ПОИС, сообщает ей, какая модель ему нужна (т.е. что необходимо учесть при моделировании и с какой степенью точности), а ПОИС автоматически формирует имитационную модель, необходимую пользователю.

В состав программного обеспечения ПОИС входят блоки типовых моделей предметных областей, планировщик моделей, БД предметных областей, а также средства диалогового общения пользователя с ПОИС.

ПОИС является достаточно сложной АИС, реализуемой, как правило, с использованием технологии ИИ на высокопроизводительных ЭВМ.

Моделирующие центры (МЦ). Это АИС, представляющая собой комплекс готовых к использованию моделей, объединенных единой предметной областью, информационной базой и языком общения с пользователями [39].

МЦ, как и ПОИС, предназначены для обеспечения проведения исследований на различных моделях. Но в отличие от ПОИС МЦ не обеспечивают автоматизацию создания имитационных моделей, а предоставляют пользователю возможность комфортной работы с готовыми моделями.

МЦ могут являться системами как коллективного, так и индивидуального использования и в принципе не требуют для своей реализации мощных ЭВМ.

Автоматизированные системы обучения. Традиционные методы обучения специалистов в различных областях профессиональной деятельности складывались многими десятилетиями, в течение которых накоплен большой опыт. Однако, как свидетельствуют многочисленные исследования, традиционные методы обучения обладают рядом недостатков. К таким недостаткам следует отнести пассивный характер устного изложения, трудность организации активной работы студентов, невозможность учета в полной мере индивидуальных особенностей отдельных обучаемых и т.д.

Одним из возможных путей преодоления этих трудностей является создание АСО — АИС, предназначенных для автоматизации подготовки специалистов с участием или без участия преподавателя и обеспечивающих обучение, подготовку учебных курсов, управление процессом обучения и оценку его результатов [39]. Основными видами АСО являются автоматизированные системы программированного обучения (АСПО), системы обучения деловым играм (АСОДИ), тренажеры и тренажерные классы (ТТК).

АСПО ориентированы на обучение в основном по теоретическим разделам курсов и дисциплин. В рамках АСПО реализуются заранее подготовленные квалифицированными преподавателями компьютерные курсы. При этом учебный материал разделяется на порции (дозы) и для каждой порции материала указывается возможная реакция обучаемого. В зависимости от действий обучаемого и его ответов на поставленные вопросы АСПО формирует очередную дозу представляемой информации.

Наибольшую сложность при создании АСПО составляет разработка компьютерного курса для конкретной дисциплины. Именно поэтому в настоящее время наибольшее распространение получили «компьютерные курсы» по традиционным, отработанным в методическом плане дисциплинам (физике, элементарной математике, программированию и т.д.).

АСОДИ предназначена для подготовки и проведения деловых игр, сущность которых заключается в имитации принятия должностными лицами индивидуальных и групповых решений в различных проблемных ситуациях путем игры по заданным правилам.

В ходе деловой игры на АСОДИ возлагаются следующие задачи:

- хранение и предоставление обучаемым и руководителям игры текущей информации о проблемной среде в процессе деловой игры в соответствии с их компетенцией;
- формирование по заданным правилам реакции проблемной среды на действия обучаемых;
- обмен информацией между участниками (обучаемыми и руководителями игры);
- контроль и обобщение действий обучаемых в процессе деловой игры;
- предоставление руководителям игры возможности вмешательства в ход игры, например для смены обстановки.

Технической базой АСОДИ являются высокопроизводительные ЭВМ или локальные вычислительные сети (ЛВС), а методологической базой, как правило, является имитационное моделирование на ЭВМ.

ТТК предназначены для обучения практическим навыкам работы на конкретных рабочих местах (постах). Они являются средствами индивидуального (тренажеры) и группового (тренажерные

комплексы) обучения.

ТТК являются достаточно дорогостоящими средствами обучения, а их создание требует больших затрат времени. Однако их чрезвычайно высокая эффективность при обучении таких специалистов, как летчики, водители, операторы систем управления и т.д., позволяет считать их достаточно перспективными видами АСО.

Автоматизированные информационно-справочные системы (АИСС). Это АИС, предназначенная для сбора, хранения, поиска и выдачи в требуемом виде потребителям информации справочного характера.

В зависимости от характера работы с информацией различают следующие виды АИСС:

автоматизированные архивы (АА);

автоматизированные системы делопроизводства (АСД);

автоматизированные справочники и картотеки;

автоматизированные системы ведения электронных карт местности (АСВЭКМ) и др.

В настоящее время разработано большое количество разновидностей АИСС, и их количество продолжает увеличиваться. АИСС создаются с использованием технологии БД, достаточно хорошо разработанной и получившей широкое распространение. Для создания АИСС, как правило, не требуется высокопроизводительной вычислительной техники.

Простота создания АИСС и высокий положительный эффект от их использования определили их активное использование во всех сферах профессиональной (в том числе и управленческой) деятельности.

1.3. Информационные и расчетные задачи в составе программного обеспечения

Согласно определению, данному в гл. 1, АИС представляет собой совокупность трех взаимосвязанных компонентов: технических средств, программных средств и организационных мероприятий. Под техническими средствами понимаются ЭВМ, устройства ввода и вывода информации (печатающие устройства, графопостроители, сканеры, плоттеры, мониторы и т.д.), устройства долговременного хранения информации (накопители на магнитной ленте или магнитном диске), сетевое оборудование и каналы связи. Технические средства АИС сами по себе не в состоянии решить какой-либо задачи. Для того чтобы АИС начала функционировать, в ЭВМ необходимо ввести программу, описывающую алгоритм работы технических средств по переработке информации в интересах решения конкретной практической задачи.

Совокупность математических методов, алгоритмических языков и алгоритмов, характеризующих логические и математические возможности ЭВМ, называется *математическим обеспечением ЭВМ.* Алгоритмы, входящие в математическое обеспечение, реализуются в ЭВМ аппаратно или программно. Аппаратная реализация алгоритмов предполагает наличие в составе ЭВМ технических устройств, преобразующих входные сигналы в выходные по жесткому, неизменяемому алгоритму.

Комплекс программ, описаний и инструкций, обеспечивающих создание и отладку программ и решение задач на ЭВМ, называется программным обеспечением (ПО) ЭВМ. По существу, ПО — это записанное на входном языке ЭВМ математическое обеспечение ЭВМ. Одно и то же математическое обеспечение может быть реализовано для различных типов ЭВМ различным ПО.

Поскольку математические методы и алгоритмы неразрывно связаны с программами, их реализующими, на практике вместо терминов «математическое обеспечение» и «программное обеспечение» часто используется термин «математическое и программное обеспечение».

При анализе состава математического и программного обеспечения ЭВМ будем вести речь о ПО, имея в виду, что аналогичный состав имеет и соответствующее математическое обеспечение.

Вариант типовой структуры ПО ЭВМ представлен на рис. 1.2 [53, 54].

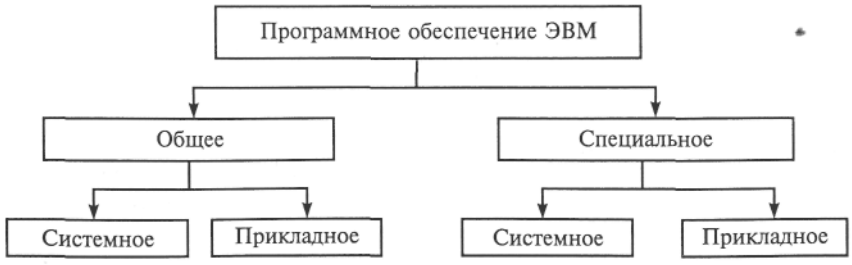


Рис. 1.2. Структура программного обеспечения ЭВМ (АИС)

ПО ЭВМ состоит из двух частей: общего программного обеспечения (ОПО) и специального программного обеспечения (СПО).

Общее программное обеспечение. Представляет собой комплекс программ, предназначенных для обеспечения работы ЭВМ в различных режимах и снижения трудоемкости создания и отладки программ пользователей.

Основные функции ОПО:

- автоматическое управление вычислительным процессом в различных режимах работы ЭВМ при минимальном вмешательстве оператора, программиста, конечного пользователя в этот процесс;
- обеспечение возможности подготовки программ к решению на ЭВМ с помощью средств автоматизации программирования;
- рациональное распределение ресурсов ЭВМ при одновременном решении нескольких задач, что значительно повышает эффективность использования ЭВМ;
- разграничение доступа различных пользователей к данным, хранимым и обрабатываемым в ЭВМ, и обеспечение защиты данных;
- контроль, диагностика и локализация неисправностей ЭВМ и т.д.

По назначению и функциональным особенностям ОПО делится на две взаимосвязанные части: общее системное программное обеспечение (ОСПО) и общее прикладное программное обеспечение (ОППО).

Общее системное программное обеспечение. В состав ОСПО входят операционная система (ОС), системы программирования (СП) и программы контроля и диагностики состояния ЭВМ.

1. *Операционной системой* называется комплекс программ, осуществляющих управление вычислительным процессом, обеспечивающих связь пользователя с ЭВМ на этапах запуска задач и реализующих наиболее общие алгоритмы обработки информации на данной ЭВМ. Главная функция ОС — обеспечение эффективной работы ЭВМ и всех внешних устройств в различных режимах работы.

Под режимом работы понимается способ организации выполнения в ЭВМ задания или нескольких заданий одновременно. Основными режимами работы являются: монопольный, многопрограммный (мультипрограммный) и режим разделения времени.

В монопольном режиме все устройства ЭВМ заняты выполнением только одного задания, являющегося основной единицей работы ЭВМ. Задание может включать несколько пунктов, выполняемых ОС последовательно. Например, задание может включать:

1) трансляцию программы;

2) компоновку оттранслированной программы;

3) запуск программы на счет.

При монопольном режиме все ресурсы ЭВМ используются по мере надобности для отработки очередного пункта задания. С точки зрения загрузки ЭВМ этот режим наименее эффективен, так как в процессе обработки одного задания различные устройства ЭВМ работают с неодинаковой нагрузкой или вообще простаивают значительную часть времени. Однако этот режим наиболее удобен для пользователя, так как время решения задачи при этом минимально. В настоящее время монопольный режим наиболее широко используется в микроЭВМ (прежде всего персональных ЭВМ).

Для увеличения производительности и эффективности использования ЭВМ за счет организации параллельной работы основных устройств применяется мультипрограммный режим работы.

В этом режиме ОС принимает к исполнению сразу несколько заданий. При достаточно большом количестве одновременно находящихся в памяти ЭВМ заданий этот режим обеспечивает практически

полную загрузку всех устройств.

Мультипрограммный режим является основным в работе ЭВМ серий ЕС и СМ (такие ЭВМ по-прежнему используются весьма широко). Кроме того, он находит частичное применение и в персональных ЭВМ высокой производительности, что позволяет пользователю одновременно ввести в ЭВМ несколько заданий. Применение мультипрограммного режима на «больших» ЭВМ позволило обеспечить пакетную обработку задач, при которой пользователи передавали задание оператору, оператор формировал пакеты этих заданий, пропускал их через ЭВМ и затем возвращал пользователям результаты решения сразу по всем заданиям, составляющим очередной пакет.

Пакетная обработка заданий позволяет существенно повысить эффективность работы ЭВМ, но крайне неудобна для пользователя, поскольку он связан с ЭВМ не непосредственно, а через оператора. Наличие этого промежуточного звена приводит к существенному увеличению суммарного времени решения задачи.

Для приближения пользователя к ЭВМ и устранения оператора как промежуточного звена между пользователем и ЭВМ были созданы ОС, реализующие особый вид мультипрограммного режима — режима разделения времени. Реализация режима разделения времени приводит к тому, что пользователи получают связь с ЭВМ поочередно на небольшой промежуток времени. Если этот промежуток времени невелик и невелико количество одновременно работающих пользователей, то каждый работающий пользователь не будет ощущать прерывистой связи с ЭВМ. Таким образом, создается впечатление, что пользователь работает один на некоторой воображаемой ЭВМ.

Недостатком режима разделения времени является уменьшение скорости вычислений пропорционально числу одновременно работающих пользователей. Однако, несмотря на этот недостаток, режим разделения времени является основным режимом работы всех современных ЭВМ, обслуживающих несколько пользователей.

Основными элементами ОС являются: процессор языка управления, супервизор и файловая система.

Процессор языка управления представляет собой программу, предназначенную для распознавания и преобразования команд пользователя и оператора ЭВМ в машинное представление с целью их последующей обработки.

Основными функциями *супервизора* являются: контроль загруженности различных устройств ЭВМ заданиями; распределение оперативной памяти между заданиями; защита одновременно решаемых заданий (задач) друг относительно друга; запуск операций ввода-вывода и т.д. По своему месту в программном обеспечении ЭВМ супервизор занимает положение посредника между аппаратным обеспечением ЭВМ и всем другим программным обеспечением машины.

Файловая система образуется программами, которые поддерживают ведение всей совокупности файлов (наборов данных) в ЭВМ. Основными функциями этой системы являются поиск требуемых файлов, модификация информации в файлах, перемещение файлов, копирование файлов, удаление файлов.

2. *Системы программирования* предназначены для обеспечения создания и отладки программ пользователей, написанных на каком-либо языке программирования (PASCAL, C, C++, FORTRAN и т. д.). В настоящее время для этих целей широко используются так называемые среды программирования (разработки программ) — например, продукты фирмы *Borland DELPHI пли Builder* C++, позволяющие быстро создавать качественные приложения.

3. *Программы контроля и диагностики* состояния ЭВМ предназначены для осуществления непрерывного контроля работы основных устройств ЭВМ, а также поиска неисправных блоков и узлов ЭВМ в случае обнаружения отказов или устойчивых сбоев.

Общее прикладное программное обеспечение. (Включает пакеты прикладных программ (ППП), системы управления базами данных (СУБД), интеграторы и другие прикладные программные системы). Особенностью объектов ОППО является то, что эти средства не требуют от пользователей при решении ими конкретных практических задач на ЭВМ проведения операций, связанных с программированием.

Под *ППП* понимается совокупность готовых к решению программ, объединяемых в пакет по единому содержательному признаку с помощью дополнительной управляющей программы. Данная программа автоматизирует и упрощает стандартную схему использования готовых программ на ЭВМ. Основными функциями управляющей программы являются: поддержание диалоговой (дружественной) формы получения информации от пользователя (обычно это режим меню); вызов соответствующих программ из пакета с целью решения ими содержательных задач, поставленных пользователем; выдача

пользователю выходных данных по решенным задачам в удобной форме.

В настоящее время ППП наряду с СУБД являются самой распространенной формой прикладного программного продукта для массового пользователя. Среди ППП выделяются пакеты трех типов: проблемно-ориентированные, интегрированные и инструментальные.

Проблемно-ориентированные ППП структурно являются наиболее простыми. Они состоят из программ, которые нацелены на решение фиксированного числа задач из относительно узкой предметной области. При этом каждой частной задаче соответствует вполне определенная программа ее решения. В функции управляющей программы входит распознавание в запросе пользователя имени и атрибутов той задачи, которую он выбирает для решения, и запуск соответствующей программы на исполнение.

Интегрированные ППП являются расширением ППП первого типа путем их наращивания такими программами, которые автоматизируют все (или большинство) сопутствующих операций, выполняемых лицом, пользующимся пакетом. К числу указанных программ наиболее часто относятся текстовый редактор, СУБД, графический редактор, реже — электронная таблица и др. В отличие от самостоятельных версий этих программ данные версии носят упрощенный характер, достаточный лишь для решения задач из соответствующей предметной области.

Инструментальные ППП отличаются отсутствием в них программ, строго ориентированных на решение конкретных практических задач. Данные пакеты состоят из программ, каждая из которых может рассматриваться как необходимый элемент решения задач из некоторой предметной области. Таким образом, при использовании данных ППП пользователь в своем запросе указывает структуру элементов, которая реализует прикладную задачу. Управляющая программа пакета на основе заданной структуры элементов создает соответствующую рабочую программу решения требуемой прикладной задачи и затем передает ей управление. В последнее время появились так называемые интеллектуальные ППП (см. разд. IV).

Специальное программное обеспечение. Объекты СПО, как правило, поставляются совместно с ЭВМ. Оно носит универсальный характер в том смысле, что позволяет создать любую программу, совместимую с аппаратными и вычислительными возможностями конкретной ЭВМ, и провести на ней расчеты. ОПО, по существу, является инструментом создания СПО — комплекса программ, предназначенных для решения конкретных управленческих, исследовательских или производственных задач. Конкретное содержание СПО полностью определяет вид конкретной АИС и, конечно, зависит от ее типа.

СПО, так же как и ОПО, состоит из двух частей: специального системного программного обеспечения (ССПО) и специального прикладного программного обеспечения (СППО). ССПО выполняет в АИС функции, аналогичные функциям ОС в ОПО.

Необходимость ССПО в вычислительных комплексах экономического назначения обусловливается двумя причинами: обеспечением требования поддержки особых (специальных) режимов проведения вычислительных работ в этих комплексах; необходимостью управления функционированием специальных устройств.

СППО представляет собой комплекс программ, каждая из которых реализует тот или иной алгоритм переработки информации. Данные программы принято называть задачами, и, хотя это название нельзя признать удачным, оно в настоящее время является общепринятым. Задачи являются основными элементами АИС, в том числе и экономического назначения, поскольку они определяют ее возможности как средства автоматизации деятельности должностных лиц при управлении персоналом.

1.4. Информационные и расчетные задачи и их классификация

Все задачи, входящие в СППО, можно классифицировать по нескольким признакам:

- характеру переработки информации;
- назначению;
- уровню применения.

Необходимость данной классификации определяется различием требований, предъявляемых к задачам каждого класса.

Основным классификационным признаком является *характер переработки информации.* В зависимости от характера переработки информации задачи, входящие в СППО, делятся на информационные и расчетные.

И н ф о р м а ц и о н н о й з а д а ч е й (ИЗ) называется элемент СППО ЭВМ (программа на ЭВМ), алгоритм переработки информации которого не приводит к созданию новой информации, отличной от исходной. Примером ИЗ могут служить задачи поиска информации, хранящейся в памяти ЭВМ, оформления (печати) бухгалтерских и управленческих документов, нанесения обстановки на карту и т.д. Таким образом, ИЗ осуществляют процессы сбора, хранения, поиска информации и преобразования ее из одного вида в другой без изменения существа этой информации и без создания новой информации.

Информационные задачи являются в настоящее время одними из самых простых, имеющими хорошо развитые средства создания, и достаточно эффективных элементов СППО при автоматизации деятельности должностных лиц. Они позволяют полностью исключить или значительно упростить прежде всего рутинные процедуры (хранение, поиск, сортировка информации, составление документов и их тиражирование и т.д.) и тем самым сократить необходимое количество персонала, занятого в основном технической деятельностью (машинистки, делопроизводители, работники библиотек, архивов и т.д.).

Р а с ч е т н о й з а д а ч е й (РЗ) называется элемент СППО ЭВМ (программа на ЭВМ), алгоритм переработки информации которого приводит к созданию новой информации, непосредственно не содержащейся в исходной. К РЗ относятся задачи анализа итогов хозяйственной деятельности; расчета показателей эффективности экономической операции; расчета заработной платы сотрудников и т.д.

В свою очередь, РЗ делятся на вычислительные задачи и математические модели.

1. Вычислительной задачей называется РЗ, алгоритм переработки информации которой построен без использования методов математического моделирования. Обычно алгоритмы вычислительных задач известны до начала их разработки и, как правило, нормативно закреплены в приказах, наставлениях, справочниках, ГОСТах и т. п. Примерами вычислительных задач являются задачи расчета подоходного налога, расчета показателей финансовой отчетности, расчета нормативного расхода средств, подведения итогов работы фирмы и т.д.

2. Математической моделью (ММ) называется РЗ, алгоритм переработки информации которой основан на использовании тех или иных методов математического моделирования.

Классификацию элементов СППО по назначению и уровню применения приведем для тех задач, которые используются в целях автоматизации управленческой деятельности (остальные будут рассматриваться ниже).

По *назначению* информационные и расчетные задачи (ИРЗ) делятся на штатные и исследовательские.

Ш т а т н о й называют ИРЗ, официально включенную в типовой цикл управления организацией и используемую должностными лицами аппаратов управления в процессе служебной, деятельности.

Штатные ИРЗ делятся на одноуровневые (используемые в звеньях управления одного уровня, например, задачи предприятия) и многоуровневые (используемые в звеньях управления нескольких уровней, например, на предприятии, объединении и в министерстве).

Основными особенностями штатных ИРЗ, непосредственно следующими из их назначения, являются высокая достоверность результатов расчетов и оперативность их получения. Кроме того, штатные задачи должны обеспечивать простоту и удобство общения с пользователем в процессе его работы на ЭВМ.

И с с л е д о в а т е л ь с к о й называется ИРЗ, используемая должностными лицами при проведении научно-исследовательских работ, обосновании перспективных программ развития, прогнозирования экономических ситуаций и т.п. Как правило, исследования проводятся с использованием ММ.

Исследовательские модели не имеют жестких требований по оперативности работы, поэтому они позволяют обеспечить широкий учет различных факторов при моделировании. Кроме того, исследовательские задачи должны обеспечивать легкость изменения (при необходимости) алгоритма своей работы в ходе исследований. При этом трудно обеспечить простоту и удобство работы с задачей. Исследовательские задачи в ряде случаев могут рассматриваться в качестве прототипов штатных задач, хотя это возможно далеко не всегда.

Глава 2. ОСНОВЫ ПРОЕКТИРОВАНИЯ ЭЛЕМЕНТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1. Основные требования и принципы разработки ИРЗ и их комплексов

ИРЗ и их комплексы составляют основу любой АИС, определяют ее возможности по автоматизации профессиональной деятельности.

Требования к СПО. Ввиду особой важности и значимости этих элементов СПО их разработка организуется в соответствии с требованиями Федеральных законов, указов, циркуляров, директив, ГОСТов и других руководящих документов [13, 14]:

- достоверность результатов использования ИРЗ и их комплексов;
- оперативность получения результатов;
- соответствие ИРЗ и их комплексов уровню руководства;
- системный подход к созданию и применению СПО;
- обеспечение безопасности информации.

Достоверность результатов. Под достоверностью результатов использования ИРЗ (расчета, моделирования) будем понимать соответствие значений параметров, получаемых в результате решения задачи, их требуемым («истинным») значениям.

Возможными причинами недостоверности получаемых в процессе расчетов результатов являются:

- неадекватность применяемой математической модели операции (процесса, явления);
- низкая точность вычислений;
- ошибки в алгоритме переработки информации, в соответствии с которым работает задача;
- ошибки пользователя при проведении расчетов;
- ошибки (сбои) в работе ЭВМ.

Под адекватностью в теории систем понимается степень соответствия используемой ММ реальному процессу (системе, объекту). Следовательно, для оценки адекватности ММ необходимо провести реальную операцию, осуществить математическое моделирование этой же операции в тех же условиях и сравнить реальные результаты операции с результатами моделирования, используя некоторый показатель, например показатель эффективности операции. Если результаты реальной операции будут хорошо согласовываться с результатами моделирования, то это означает, что используемая ММ в данных условиях проведения операции является адекватной реальному процессу (системе, объекту). Важно отметить, что в этом случае можно количественно оценить адекватность модели в рамках суждений типа «результаты моделирования расходятся с реальными не более чем на 10 %».

Формально оценить адекватность модели не всегда удается, поскольку не всегда возможно проведение реальной операции для сравнения с результатами моделирования (например, для моделей операций, предусматривающих применение ядерного оружия, или крупномасштабных экономических моделей). В таких условиях под адекватностью принято понимать степень доверия должностного лица к результатам моделирования, используемым для принятия решений. При этом невозможно ввести показатель, объективно характеризующий степень адекватности модели. Модель может быть или адекватной, или неадекватной. Должностное лицо должно сделать вывод об адекватности модели на основании анализа существа модели и полноты учета всех факторов, влияющих на проведение операции в конкретных условиях.

Низкая точность вычислений также может стать причиной недостоверности получаемых результатов расчета. Существуют две возможные причины возникновения ошибок вычислений: методические ошибки и ошибки округления. Методические ошибки связаны с использованием приближенных численных методов (например, при использовании метода численного интегрирования или дифференцирования функций). Ошибки округлений связаны с тем, что числа в ЭВМ представляются всегда с некоторой точностью, определяемой количеством значащих цифр в записи числа (для современных ЭВМ такие ошибки практически всегда связаны с неверными действиями пользователей, в частности при программной реализации ИРЗ).

Ошибки в алгоритме переработки информации, в соответствии с которым работает ЭВМ, являются достаточно редким источником недостоверности результатов расчетов и, как правило, бывают связаны с неучетом в алгоритме задачи всех возможных вариантов исходных данных. При некоторых вариантах исходных данных могут возникнуть ситуации, когда алгоритм задачи работает с ошибками. Поэтому при создании алгоритма задачи необходимо тщательно проанализировать возможные значения исходных данных и определить их допустимые значения. Выявление ошибок в алгоритме переработки информации является одной из важнейших целей при проведении контрольных расчетов на этапе приемки ИРЗ.

Ошибки пользователя при проведении расчетов на первый взгляд невозможно исключить за счет создания специальных алгоритмических и программных средств. Тем не менее существуют способы уменьшения возможностей для появления таких ошибок (конечно, имеются в виду непреднамеренные, «случайные» ошибки). Речь идет о программном контроле вводимой пользователем информации. Эта информация может включать значения параметров или команды. Как правило, при вводе параметров можно программно проконтролировать допустимость значения вводимого параметра, причем ограничения на значения параметра могут быть как постоянными, так и изменяться в зависимости от значений других параметров. Например, в задаче планирования транспортной операции по доставке потребителям какой-либо продукции допустимые значения скорости движения зависят от типов транспортных средств, участвующих в операции, и состояния дорог на маршрутах движения.

Что касается контроля команд, вводимых пользователем, то он может включать проверку допустимости данной команды на конкретном этапе работы с задачей (например, проверка наличия всех необходимых исходных данных перед выполнением команды начала расчета), а также выдачу на экран монитора запроса для подтверждения пользователем намерения выполнить какую-либо важную команду (например, при уничтожении каких-либо данных на экран монитора выводится вопрос: «Вы действительно хотите уничтожить эти данные?» и требуется утвердительный ответ пользователя для выполнения команды). Кроме того, особо ответственные команды могут предусматривать запрос на подтверждение полномочий на их проведение (например, ввод пароля).

Ошибки (сбои) в работе ЭВМ могут повлиять на достоверность результатов расчетов, если они не селектируются техническими средствами и операционной системой. Единственным средством исключения неселектируемых ошибок (сбоев) в работе ЭВМ является повторное решение задачи. Поэтому наиболее ответственные расчеты должны дублироваться на другой ЭВМ и (или) с использованием другой задачи, имеющей аналогичный алгоритм.

Оперативность получения результатов. Под оперативностью получения результатов расчетов на ИРЗ понимается возможность практического использования результатов их решения (расчетов, моделирования) либо в реальном ритме работы, либо за заданное время. Задача обладает требуемой оперативностью решения, если время работы пользователя с ней обеспечивает своевременное применение получаемых результатов в профессиональной деятельности. Время работы с задачей включает время на настройку (при необходимости) ПО (а иногда и технических средств), подготовку исходных данных, ввод их в ЭВМ, проведение расчетов и выдачу результатов в виде, удобном для дальнейшего использования.

Таким образом, оперативность получения результатов расчетов является интегральной характеристикой, которая включает в себя не только скорость вычислений по алгоритму задачи, но и скорость ввода исходных данных, а также получение результатов в виде, не требующем какой-либо дополнительной обработки (переписывания, перепечатывания и т.д.). Поэтому при создании ИРЗ необходимо предусматривать минимально необходимый объем исходных данных, вводимый пользователем при использовании задачи, а также удобство их ввода.

Соответствие уровню руководства. Под требованием соответствия ИРЗ и их комплексов уровню руководства понимается:

- использование в них информации с детализацией и точностью, которыми располагает должностное лицо (лица), работающее с задачей;
- представление результатов в наглядном (привычном для пользователя) виде, соответствующем форме и содержанию реальных документов;
- применение показателей, имеющих для конкретного должностного лица ясный технический, оперативный и физический смысл (так называемых транспарентных показателей).

Системный подход. Требование системного подхода означает, что все создаваемые ИРЗ и их комплексы должны быть составными элементами общей системы задач и моделей, т. е. согласованы между собой по цели и назначению; составу учитываемых факторов и ограничений; содержанию и формам входных и выходных документов, показателей и критериев эффективности, нормативов; структуре и содержанию информационной базы, принципам защиты обрабатываемой информации.

Обеспечение безопасности информации. Требование обеспечения безопасности обрабатываемой информации заключается в исключении возможности уничтожения или искажения информации, обрабатываемой на ЭВМ, а также возможности несанкционированного получения этой информации не допущенными к ней лицами. Выполнение данного требования достигается осуществлением комплекса

организационных мероприятий и технических мер.

Принципы разработки и использования СПО. Помимо основных требований к создаваемым ИРЗ и их комплексам, руководящими (нормативными) документами определены и основные принципы разработки и поддержания в работоспособном состоянии элементов СПО. Руководство данными принципами является обязательным и позволяет создавать и применять ИРЗ и их комплексы, отвечающие приведенным в подразд. 2.3 требованиям. Сформулируем эти принципы применительно к средствам автоматизации наиболее сложной области профессиональной деятельности — управлению сложными человеко-машинными системами экономического назначения:

- централизованная разработка по единому плану и замыслу на общих информационных и математических основах;
- конкретность предназначения создаваемых задач и их комплексов;
- непосредственное руководство и участие в создании задач предприятий и фирм (организаций), в интересах которых они создаются;
- обеспечение возможности перестройки задач в процессе их эксплуатации применительно к конкретной обстановке;
- непрерывное сопровождение разработанных ИРЗ и их комплексов представителями заказчика и разработчика.

Централизованная разработка. Принцип централизованной разработки по единому плану и замыслу на общих информационных и математических основах используется при создании ИРЗ и их комплексов в рамках единой АСУ. Этот принцип должен неукоснительно соблюдаться при создании задач, результаты решения которых используются во всех или нескольких звеньях АСУ (например, задач, используемых для автоматизации управления отраслью экономики в министерстве).

Для обеспечения централизованной разработки ИРЗ в вышестоящих организациях формируется и утверждается перспективный план создания элементов СПО. В перспективном плане указывается: название ИРЗ, ее заказчик и разработчик, а также срок создания задачи. Перспективный план, как правило, разрабатывается сроком на пять лет. На основании перспективных планов разрабатываются годовые планы создания ИРЗ.

Принцип централизованной разработки ИРЗ может не учитываться организациями и фирмами, создающими одноуровневые задачи, предназначенные для применения в рамках данной организации, и использующими автономные ЭВМ (например, персональные ЭВМ, не входящие в АСУ). При этом организации выступают в роли заказчика ССПО и осуществляют разработку (совершенствование) ИРЗ на основании своих перспективных планов. Отметим, что с насыщением аппаратов управления современной ЭВТ следовать этому принципу становится все труднее и на первое место при его реализации выдвигаются организационные мероприятия.

Конкретность предназначения. Принцип конкретности предназначения создаваемых задач и их комплексов предполагает необходимость разработки элементов СПО, специально предназначенных для автоматизации решения конкретных задач управления.

На практике достаточно часто встречаются ситуации, когда создается задача для проведения научных исследований или в учебных целях, а затем предпринимаются попытки внедрения этой задачи (как правило, с некоторыми доработками) в той или иной организации.

Однако, поскольку исследовательские и учебные задачи создаются в целях проведения научных исследований или обучения, они не могут эффективно использоваться, а зачастую являются просто непригодными для автоматизации управления предприятиями и фирмами. Исследовательские задачи, обладая обычно высокими показателями достоверности результатов, имеют плохую оперативность расчетов и слабую эргономичность (не отвечают требованиям удобства и простоты работы должностных лиц с задачей). Учебные задачи имеют высокую оперативность расчетов и эргономичность, но достоверность получаемых результатов, как правило, является недостаточной для использования при автоматизации управления на практике.

Таким образом, исследовательские и учебные задачи нуждаются в существенной переработке перед их внедрением в промышленность. Такая переработка является достаточно трудоемкой, причем затраты на доработку задачи соизмеримы с затратами на создание новой задачи.

Поэтому более правильным является путь, когда ИРЗ создается специально для автоматизации деятельности руководителя или должностных лиц предприятия или фирмы при решении конкретной задачи управления персоналом. При этом, конечно, необходимо использовать отдельные математические алгоритмы, фрагменты программ, а также опыт создания и использования

исследовательских и учебных задач, являющихся прототипами разрабатываемых ИРЗ.

Непосредственное руководство заинтересованных предприятий и фирм (организаций). Принцип непосредственного руководства и участия в создании ИРЗ и их комплексов предприятий и фирм (организаций), в интересах которых они создаются, является важнейшим принципом, лежащим в основе всей технологии создания СПО и обеспечивающим создание качественных задач для автоматизации управления персоналом фирм (организаций).

Разработчики задачи, как правило, плохо представляют себе специфику управления персоналом, а также роль создаваемой задачи в процессе управления и предъявляемые к ней требования. Учет этой специфики и соответствующих требований к задаче должен проводиться в процессе разработки ее оперативной постановки, являющейся совместным документом заказчика и разработчика.

Непрерывный контроль со стороны заказчика на всех этапах создания ИРЗ позволяет избежать неправильного толкования разработчиком положений и требований оперативной постановки задачи, своевременно устранить недостатки и тем самым ускорить создание и улучшить качество создаваемых задач.

Кроме того, участие в разработке оперативной постановки и контроля результатов отдельных этапов создания ИРЗ позволит должностным лицам, для которых создается задача, глубже понять механизмы переработки информации в задаче. Понимание должностными лицами этих механизмов обеспечит грамотное и эффективное применение задач в процессе решения задач управления.

Возможность перестройки. Принцип обеспечения возможности перестройки задач в процессе их эксплуатации применительно к конкретной обстановке предполагает, что при создании ИРЗ необходимо более полно учесть возможные изменения обстановки, внешних условий, а также изменения характеристик и условий применения создаваемой продукции, которые вызовут необходимость корректировки алгоритмов и программ ИРЗ.

Конечно, заранее предусмотреть и оговорить какие-либо конкретные изменения (кроме плановых, например, модернизации продукции или договорных ограничений) невозможно. Тем не менее при разработке задач необходимо учитывать возможные направления изменения тех или иных параметров и создавать такие задачи, которые позволили бы с минимумом затрат проводить их корректировку.

Непрерывное сопровождение ИРЗ и их комплексов заказчиком и разработчиком. Непрерывное сопровождение разработанных ИРЗ и их комплексов представителями заказчика и разработчика является основным условием, обеспечивающим поддержание задач и их комплексов в готовности к применению. В функцию представителей заказчика при сопровождении ИРЗ и их комплексов входит обеспечение работоспособности используемых задач, а также анализ процесса их эксплуатации и выработка предложений по их совершенствованию. Представители разработчика при сопровождении ИРЗ устраняют недостатки, выявляемые в процессе эксплуатации, и проводят совершенствование задач в плане повышения их эксплуатационных характеристик.

Организационные мероприятия. Перечисленные выше основные принципы и основные требования являются нормативной базой при разработке и применении СПО в АИС.

Организационные мероприятия включают в себя следующие этапы:

- перспективное централизованное планирование;
- организация сопровождения ПО;
- применение современных технологий программирования;
- разработка полной документации;
- применение современных методов моделирования.

Конечно, уровни обеспечения всех требований существенно зависят от класса задачи, ее назначения и особенностей применения. В зависимости от существа и особенностей применения создаваемых задач перечень требований к ним может расширяться или сужаться. Формирование конкретных требований к создаваемым задачам осуществляется совместными усилиями представителей заказчика и разработчика на этапе разработки технического задания и утверждается заказчиком.

2.2. Содержание работ на этапах создания ИРЗ и их комплексов

Порядок создания ИРЗ и их комплексов определен законодательством Российской Федерации, а также ГОСТами. В создании задач и их комплексов участвуют две стороны: заказчик и разработчик. Если разработчиков несколько, то среди их определяется головной разработчик и соисполнители. Возможно также существование нескольких заказчиков. Тогда среди них выделяется головной заказчик,

а остальные заказчики называются созаказчиками.

Процесс создания ИРЗ и их комплексов в принципе одинаков и включает следующие этапы [54]:

- разработка технического задания;
- эскизное проектирование;
- техническое проектирование;
- рабочее проектирование.

По решению заказчика, сформированному в техническом задании (ТЗ), допускается объединение отдельных этапов разработки, изменение их содержания или введение других этапов. Продолжительность этапов, а также уровень, с которого начинается разработка задач и их комплексов, определяется в каждом конкретном случае исходя из имеющегося научно-методического задела по данной проблеме. Каждый этап завершается в порядке, установленном ТЗ. В частности, итоги каждого этапа рассматриваются заказчиком.

Для повышения оперативности взаимодействия заказчика и разработчика, непрерывного контроля за деятельностью разработчика желательно из состава заказывающей организации выделить сотрудника, которому поручается научно-техническое и организационное сопровождение всех видов работ по созданию ИРЗ и их комплексов. Этот сотрудник называется сотрудником сопровождения.

Разработка технического задания. ТЗ является исходным документом, устанавливающим основное назначение, технические характеристики и требования, предъявляемые к создаваемым задачам и их комплексам, а также порядок работ на всех этапах и сроки их проведения. ТЗ формируется заказчиком совместно с разработчиком.

Для проведения работ на этом этапе заказчик может создавать рабочие группы из своих представителей, представителей разработчика и других специалистов (экспертов) в зависимости от характера создаваемых задач и моделей.

При разработке ТЗ осуществляется:

- проведение информационного обследования объекта автоматизации и уточнение функций и задач управления, подлежащих автоматизации;
- определение необходимого состава комплекса ИРЗ;
- разработка оперативных постановок задач;
- формирование задания (определение разработчиков, сроков и порядка создания задач и их комплексов) и исходных данных.

Проведение информационного обследования объекта автоматизации и уточнение функций и задач управления, подлежащих автоматизации, являются необходимым элементом этапа разработки ТЗ. При информационном обследовании анализируется процесс функционирования объекта автоматизации по переработке информации и определяются те элементы этого процесса, которые могут или должны быть возложены на ЭВМ. Одним из результатов информационного обследования является состав комплекса задач, которые должны быть разработаны.

Основным документом, содержащим всю информацию о создаваемой задаче (или комплексе задач), ее назначении и требованиях к ней, является оперативная постановка задачи (или комплекса задач), которая оформляется как обязательное приложение к ТЗ.

ТЗ в целом и оперативные постановки задач подписываются головным разработчиком, согласовываются с организациями-соисполнителями, аппаратом управления, на котором будет внедряться задача, и утверждаются заказчиком. Проведение разработки задач и их комплексов без утвержденного ТЗ не допускается. В процессе дальнейших работ по созданию АИС или ее элементов при невозможности выполнения требований оперативной постановки она может корректироваться с разрешения заказчика на любом этапе создания и внедрения ИРЗ и их комплексов.

Эскизное и техническое проектирования. После утверждения заказчиком ТЗ разработчик приступает к этапу эскизного проектирования, который часто объединяют с этапом технического проектирования. На этих этапах осуществляются следующие действия:

- определение принципов построения, состава и структуры технических и программных средств ИРЗ и их комплексов (этап эскизного проектирования);
- определение обобщенного алгоритма функционирования, назначения и порядка работы элементов задач и их комплексов (этап эскизного проектирования);
- определение содержания и общих характеристик информационных связей между элементами задач (комплексов задач) (этап эскизного проектирования);

- определение состава необходимого ПО для создания задач и их комплексов (этап эскизного проектирования);
- выбор используемых математических методов и математическое описание моделей экономических операций (этап эскизного проектирования);
- оценка возможности выполнения основных требований оперативной постановки задачи (этап эскизного проектирования);
- разработка детальных алгоритмов задач и комплексов, их информационного и лингвистического обеспечения (этап технического проектирования);
- проектирование и разработка необходимых БД (этап технического проектирования).

Алгоритмы ИРЗ и их комплексов разрабатываются в строгом соответствии с утвержденным ТЗ и оперативными постановками задач и являются определяющими документами для последующего написания программ. Схемы алгоритмов и программ выполняются в соответствии с нормативными требованиями.

Рабочее проектирование. На этом этапе в соответствии с разработанными ранее алгоритмами осуществляется разработка программ, их отладка и экспериментальная проверка (испытания) на ЭВМ и оформление документации по разработанной задаче (или комплексу задач).

Перед сдачей отлаженных программ заказчику разработчик проводит их испытания с целью проверки соответствия программного продукта требованиям ТЗ. В процессе испытаний проверяются:

- достоверность результатов расчетов в различных вариантах исходных данных и, в частности, адекватность ММ операций;
- характер влияния различных исходных данных на результаты расчета (моделирования);
- надежность применяемых технических и программных средств защиты данных;
- оперативность полученных результатов расчетов;
- удобство работы с ЭВМ в процессе расчета или моделирования;
- качество разработанных алгоритмов и программ и т.д.

Проверка достоверности результатов проводится на вариантах исходных данных с реальной или учебной информацией, обеспечивающих проведение всесторонней оценки получаемых результатов путем сравнения с результатами проведенных экономических операций. Для окончательной оценки достоверности результатов расчетов (моделирования) могут привлекаться компетентные эксперты.

Все работы по проверке готовности программного продукта проводятся на технической базе разработчика. В работе по проверке (испытанию) ИРЗ и их комплексов участвует сотрудник сопровождения. Обобщенные результаты экспериментальной проверки разработанных задач и комплексов представляются заказчику вместе с отчетными материалами по программному изделию, подготовленному к сдаче.

На каждую ИРЗ и в целом комплекс задач оформляется отчетная документация в четырех частях.

Часть 1. Оперативная постановка задачи.

Часть 2. Алгоритмы задачи.

Часть 3. Описание программы. Инструкция оператору-программисту по ее применению. Программы на магнитных носителях и их распечатки (тексты программ).

Часть 4. Инструкция должностному лицу по использованию задачи (комплекса задач).

Каждая часть документации оформляется отдельной книгой (или несколькими книгами). Части 1, 2, 4 используются специалистами аппарата управления при изучении сущности задачи и порядка работы с ней. В вычислительный центр (ВЦ) документация передается в полном объеме.

Помимо указанной отчетной документации, после завершения каждого этапа разработки задачи (комплекса) разработчик представляет заказчику отчет. Этим обеспечивается объективный контроль за ходом создания задач. Порядок, сроки выпуска и содержание таких отчетов оговариваются в ТЗ. Анализ отчетов и выдача заключений по результатам каждого этапа работ производятся, как правило, при активном участии сотрудника сопровождения.

Приведенная выше этапность создания задач (комплексов задач) и отчетность в процессе их создания не является строго обязательной (кроме документации по готовым задачам и комплексам задач) и зависит от объема и сложности создаваемой задачи (комплекса задач). В любом случае обязательным документом является ТЗ, в котором оговаривается как содержание этапов создания задач, так и состав документов, разрабатываемых на каждом этапе.

2.3. Порядок внедрения и использования ИРЗ и их комплексов

Прием в эксплуатацию разработанных ИРЗ и их комплексов осуществляется по приказу или директиве заказчика. Этим приказом заказчик назначает комиссию по приемке готового программного продукта, определяет ее состав и задачи, а также сроки и порядок ее работы.

В задачи комиссии входит:

- изучить документацию по задаче (комплексу задач), представленную разработчиком, определить ее качество, соответствие требованиям ТЗ и другим нормативным документам;
- установить соответствие алгоритмов и программ требованиям ТЗ и оперативной постановке задачи;
- проверить достаточность мер по обеспечению безопасности обработки информации и ее выдачи на автоматизированные рабочие места должностных лиц;
- провести контрольные расчеты на внедряемой задаче с целью проверки: работоспособности программ, достоверности, полноты и качества получаемых результатов в широком диапазоне исходных данных; практической пригодности и эффективности использования задачи в деятельности руководителя по управлению персоналом; технологичности, трудоемкости и временных характеристик основных этапов подготовки и ввода исходных данных, проведения расчетов и выдачи результатов;
- подготовить предложения о внесении изменений в методику работы должностных лиц с использованием результатов применения задачи;
- при необходимости определить перечень работ и продолжительность этапа опытной эксплуатации задачи (комплекса), а также составить план-задание на ее проведение.

Приведенный выше перечень задач комиссии по решению заказчика может быть расширен.

При приемке задачи (комплекса) головной разработчик обязан:

- не позднее чем за два месяца до начала работы комиссии представить заказчику и организациям, определенным заказчиком, по одному экземпляру документации в согласованном с заказчиком объеме;
- представить эталонные программы с контрольными вариантами решений на машинных носителях в ВЦ, на котором планируется внедрение задачи (комплекса);
- выполнить контрольные расчеты по указанию заказчика и принять участие в анализе полученных результатов;
- устранить недостатки, выявленные в процессе приемки задачи (комплекса).

По указанию заказчика в приемке задачи (комплекса) могут принять участие заинтересованные фирмы, где планируется использование внедряемых задач и их комплексов. В этом случае заинтересованные организации обязаны:

- принять участие в подготовке контрольных вариантов решений;
- оценить возможности задачи, обеспечить проверку ее работоспособности в условиях, соответствующих особенностям работы данного предприятия;
- представить заказчику в установленные им сроки предложения и замечания о результатах проверки задачи (комплекса).

ВЦ, на котором планируется внедрение задачи, предоставляет технические средства, участвует в контрольных решениях и проверяет эксплуатационно-технические характеристики задачи (комплекса).

По результатам работы комиссия представляет заказчику акт по приемке задачи (комплекса). Копия акта высылается в адрес головного разработчика и является заключением заказчика на выполненное ТЗ.

ИРЗ (или их комплекс) допускается к штатной эксплуатации, если она отвечает требованиям ТЗ и является работоспособной в реальных условиях профессиональной деятельности. ИРЗ (или их комплекс) принимается в опытную эксплуатацию, если в ходе приемки выявлена необходимость в ее доработках, не оказывающих существенного влияния на достоверность результатов расчетов. Схема порядка внедрения ИРЗ и их комплексов следующая.

1. Приказ (директива) заказчика о приеме ИРЗ и их комплексов.
2. Подготовка к приему ИРЗ и их комплексов:

- заказчик создает комиссию по приему ИРЗ и их комплексов; привлекает заинтересованные фирмы (при необходимости);
- разработчик представляет документацию, эталонные программы, результаты контрольных

расчетов.

3. Прием ИРЗ и их комплексов в эксплуатацию:

- анализ полноты и качества документации по задаче;
- установление соответствия алгоритмов и программ требованиям ТЗ;
- проверка достаточности мер по обеспечению безопасности информации;
- проверка работоспособности программ в широком диапазоне исходных данных, достоверности, полноты и качества получаемых результатов;
- проверка технологичности, оперативности и трудоемкости работы с задачей;
- подготовка предложений о внесении изменений в порядок работы должностных лиц с использованием задачи;
- разработка плана-задания на опытную эксплуатацию задачи (при необходимости).

4. Оформление результатов приема ИРЗ и их комплексов:

- подготовка акта комиссии о приеме задачи (комплекса задач);
- подготовка директивы заказчика о приеме задачи (комплекса задач) в эксплуатацию (или об организации опытной эксплуатации).

Для организации опытной эксплуатации заказчиком утверждается план-задание, в котором указываются:

- сроки опытной эксплуатации;
- содержание и порядок работы;
- меры по обеспечению безопасности обработки информации;
- способы оценки эффективности применения данной задачи (комплекса) в практической работе предприятия.

В ходе опытной эксплуатации организуется обучение персонала фирмы порядку работы с задачей (комплексом), изучение ее возможностей и проведение оценки эффективности ее применения. В практике должностных лиц в реальных условиях эксплуатации оценивается надежность средств и методов защиты информации, проводятся все необходимые доработки задачи.

Головной разработчик обязан участвовать на всех этапах опытной эксплуатации и устранять недостатки, выявленные как в процессе приемки задачи, так и в процессе опытной эксплуатации.

Опытная эксплуатация ИРЗ и их комплексов завершается актом, разрабатываемым фирмой, в котором должна использоваться задача. Акт подписывается всеми участниками опытной эксплуатации и представляется заказчику.

Прием задачи (комплекса) осуществляется по директиве (приказу) заказчика. Директивой (приказом) устанавливается порядок применения задачи (комплекса) соответствующими должностными лицами.

Применение ИРЗ и их комплексов организуется и осуществляется на основании указаний руководителя предприятия распоряжений вышестоящих организаций и других руководящих документов.

Документы, регламентирующие применение ИРЗ и их комплексов, должны включать:

- цели и сроки применения каждой задачи;
- порядок проведения расчетов (моделирования) в различных условиях обстановки, подготовки исходных данных, анализа промежуточных и конечных результатов, выдачи их в соответствующие аппараты управления и использования результатов решения в процессе управления;
- порядок обобщения опыта применения задач и их комплексов, разработки и реализации предложений по совершенствованию методов работы с использованием результатов расчетов;
- список сотрудников, выделенных для оперативного сопровождения задач и их комплексов;
- перечень мероприятий по исключению утечки информации в процессе производства расчетов и анализа их результатов;
- порядок подготовки и допуска персонала фирмы к работе с задачей и с использованием результатов расчетов, а также порядок проведения необходимых периодических тренировок с лицами, допущенными к работе с задачей;
- перечень мероприятий по поддержанию в работоспособном состоянии средств программного, технического и других видов обеспечения.

Ответственность за внедрение, освоение оперативным составом, применение, совершенствование задач и их комплексов, а также обеспечение безопасности информации в процессе ее обработки

возлагается на руководителя соответствующего предприятия. Ответственность за поддержание задач и их комплексов в работоспособном состоянии возлагается на начальника ВЦ.

Оперативное сопровождение задач и их комплексов осуществляется выделенными для этой цели сотрудниками фирмы и включает:

- поддержание программ и средств их информационного обеспечения в работоспособном состоянии;
- подготовку предложений по совершенствованию оперативных постановок, алгоритмов и инструкций по использованию задач и их комплексов в связи с изменением взглядов на проведение экономического анализа, появлением новых технических автоматизированных средств, новой организационной структуры предприятия, отрасли и т.д.;
- подготовку рекомендаций по совершенствованию методики работы фирмы с использованием разработанных ИРЗ и их комплексов.

В процессе эксплуатации задач и их комплексов разработчик осуществляет научно-техническое сопровождение (авторский надзор), которое включает совершенствование математических методов, алгоритмов, программ и информационного обеспечения в целях повышения оперативно-технических характеристик задач и их комплексов.

Глава 3. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ ЭКОНОМИЧЕСКИХ СИСТЕМ

3.1. Сравнительный анализ стандартов информационной безопасности

Опыт эксплуатации существующих компьютерных систем обработки информации показывает, что проблема обеспечения безопасности еще далека от своего решения, а предлагаемые производителями различных систем средства защиты сильно различаются как по решаемым задачам и используемым методам, так и по достигнутым результатам. Это определяет актуальность проблемы построения защищенных систем обработки информации, решение которой следует начать с анализа причин сложившейся ситуации.

Проблема защиты машинной информации на современном уровне развития информатизации общества столь важна и многогранна, что заслуживает более подробного рассмотрения, чем другие аспекты автоматизации профессиональной деятельности. Более подробные сведения можно найти в других источниках (например, [14, 19, 20, 31, 33]).

Для того чтобы объединить усилия всех специалистов в направлении конструктивной работы над созданием защищенных систем, необходимо определить, что является целью исследований, что мы хотим получить в результате и чего в состоянии достичь. Для ответа на эти вопросы и согласования всех точек зрения на проблему создания защищенных систем разработаны и продолжают разрабатываться стандарты информационной безопасности. Это документы, регламентирующие основные понятия и концепции информационной безопасности на государственном или межгосударственном уровне, определяющие понятие «защищенная система» посредством стандартизации требований и критериев безопасности, образующих шкалу оценки степени защищенности вычислительных систем (ВС). В соответствии с этими документами защищенная система обработки информации представляет собой систему, отвечающую тому или иному стандарту информационной безопасности. Этот факт позволяет сопоставлять степени защищенности различных систем относительно установленного стандарта.

Основные понятия и определения. *Политика безопасности* — совокупность норм и правил, обеспечивающих эффективную защиту системы обработки информации от заданного множества угроз.

Модель безопасности — формальное представление политики безопасности.

Дискреционное, или произвольное, управление доступом — управление доступом, основанное на совокупности правил предоставления доступа, определенных на множестве атрибутов безопасности субъектов и объектов, например, в зависимости от грифа секретности информации и уровня допуска пользователя.

Ядро безопасности — совокупность аппаратных, программных и специальных компонентов ВС, реализующих функции защиты и обеспечения безопасности.

Идентификация — процесс распознавания сущностей путем присвоения им уникальных меток (идентификаторов).

Аутентификация — проверка подлинности идентификаторов сущностей с помощью различных

(преимущественно криптографических) методов.

Адекватность — показатель реально обеспечиваемого уровня безопасности, отражающий степень эффективности и надежности реализованных средств защиты и их соответствия поставленным задачам (в большинстве случаев это задача реализации политики безопасности).

Квалификационный анализ, квалификация уровня безопасности — анализ ВС с целью определения уровня ее защищенности и соответствия требованиям безопасности на основе критериев стандарта безопасности.

Таксономия — наука о систематизации и классификации сложноорганизованных объектов и явлений, имеющих иерархическое строение. Таксономия основана на декомпозиции явлений и поэтапном уточнении свойств объектов (иерархия строится сверху вниз).

Прямое взаимодействие — принцип организации информационного взаимодействия (как правило, между пользователем и системой), гарантирующий, что передаваемая информация не подвергается перехвату или искажению.

Защищенная система обработки информации для определенных условий эксплуатации обеспечивает безопасность (доступность, конфиденциальность и целостность) обрабатываемой информации и поддерживает свою работоспособность в условиях воздействия на нее заданного множества угроз.

Под защищенной системой обработки информации предлагается понимать систему, которая:

- осуществляет автоматизацию некоторого процесса обработки конфиденциальной информации, включая все аспекты этого процесса, связанные с обеспечением безопасности обрабатываемой информации;
- успешно противостоит угрозам безопасности, действующим в определенной среде;
- соответствует требованиям и критериям стандартов информационной безопасности.

Отсюда вытекают следующие задачи, которые необходимо и достаточно решить, для того чтобы создать защищенную систему обработки информации, а именно:

- в ходе автоматизации процесса обработки конфиденциальной информации реализовать все аспекты этого процесса, связанные с обеспечением безопасности обрабатываемой информации;
- обеспечить противодействие угрозам безопасности, действующим в среде эксплуатации защищенной системы;
- реализовать необходимые требования соответствующих стандартов информационной безопасности.

Угрозы безопасности компьютерных систем. Под *угрозой безопасности* компьютерных систем понимаются воздействия на систему, которые прямо или косвенно могут нанести ущерб ее безопасности. Приведем наиболее общую классификацию возможных угроз безопасности. Все угрозы можно разделить по их *источнику* и *характеру* проявления.

Случайные угрозы возникают независимо от воли и желания людей. Данный тип угроз связан прежде всего с прямым физическим воздействием на элементы компьютерной системы (чаще всего природного характера) и ведет к нарушению работы этой системы и/или физическому уничтожению носителей информации, средств обработки и передачи данных, физических линий связи.

Причиной возникновения технических угроз случайного характера могут быть как сбои вследствие ошибок персонала (порожденные людьми), так и случайные нарушения в работе оборудования системы (например, вследствие поломки какого-либо узла или устройства, сбоя в работе ПО или элементарное короткое замыкание). Последствиями подобных событий могут быть отказы и сбои аппаратуры, искажение или уничтожение информации, нарушение линий связи, ошибки и физический вред персоналу.

Примером реализации случайной угрозы, созданной людьми, может быть физическое нарушение проводных линий связи из-за проведения строительных работ. Другими словами, угрозы данного типа возникают вследствие каких-либо действий людей, целью которых не является нанесение физического вреда и нарушение функционирования работы компьютерной системы и/или отдельных ее сегментов и ресурсов, однако побочный эффект данных действий приводит к нарушениям и сбоям в работе системы.

Преднамеренные угрозы в отличие от случайных могут быть созданы только людьми и направлены именно на дезорганизацию компьютерной системы. Примером реализации такой угрозы может быть как физическое уничтожение аппаратуры и сетевых коммуникаций системы, так и нарушение ее целостности и доступности, а также конфиденциальности обрабатываемой и хранимой ею информации с применением средств и ресурсов самой системы, а также с использованием дополнительного оборудования.

Ниже приведена более подробная классификация угроз информационной безопасности в зависимости от их источника.

1. Природные угрозы.
   1.1. Стихийные бедствия.
   1.2. Магнитные бури.
   1.3. Радиоактивное излучение и осадки.
   1.4. Другие.
2. Угрозы техногенного характера.
   2.1. Отключения или колебания электропитания и сбои в работе других средств обеспечения функционирования системы.
   2.2. Отказы и сбои в работе аппаратно-программных средств компьютерной системы.
   2.3. Электромагнитные излучения и наводки.
   2.4. Утечки через каналы связи: оптические, электрические, звуковые.
   2.5. Другие.
3. Угрозы, созданные людьми.
   3.1. Непреднамеренные действия.
      3.1.1. Обслуживающего персонала.
      3.1.2. Управленческого персонала.
      3.1.3. Программистов.
      3.1.4. Пользователей.
      3.1.5. Архивной службы.
      3.1.6. Службы безопасности.
   3.2. Преднамеренные действия.
      3.2.1. Обслуживающего персонала.
      3.2.2. Управленческого персонала.
      3.2.3. Программистов.
      3.2.4. Пользователей.
      3.2.5. Архивной службы.
      3.2.6. Службы безопасности.
      3.2.7. Хакерские атаки.

Угрозы техногенного характера связаны с надежностью работы аппаратно-программных средств компьютерной системы. При этом угрозы подгруппы 2.1 связаны с внезапным временным прекращением работы системы и ведут к потерям информации и управления объектами системы. Угрозы подгруппы 2.2 связаны с надежностью работы аппаратно-программных средств и ведут к искажению и потерям информации, нарушениям в управлении объектами. Угрозы подгруппы 2.3 связаны с наличием электромагнитных излучений, за счет которых может происходить несанкционированный перенос информации за пределы защищаемой системы. Угрозы подгруппы 2.4 связаны с утечкой информации через легальные каналы связи за счет использования специального оборудования.

Угрозы группы 3 связаны с людьми, непосредственно работающими с компьютерной системой. Непреднамеренные угрозы связаны со случайными действиями пользователей, ошибками операторов, программистов, управленческого персонала, сотрудников архивной службы и службы безопасности и ведут к искажению или уничтожению информации, нарушению функционирования, управления и безопасности системы, а также ошибкам и сбоям в работе программно-аппаратных средств.

Угрозы, «носителями» которых являются хакерские атаки, связаны с преднамеренными действиями людей, направленными на нанесение ущерба системе с использованием средств и возможностей штатного оборудования системы и любых других возможностей, которые могут быть получены с применением всех имеющихся на данный момент времени информационных технологий. Данная группа угроз является наиболее многочисленной.

Необходимо особо отметить такой вид угроз, как внедрение компьютерных вирусов, программ — «троянских коней», логических бомб и т.д. Данный вид угроз может относиться как к группе 3.1, так и к группе 3.2 в связи с тем, что программы такого типа могут быть специально разработанными «боевыми вирусами» или специально внедренными программными закладками для выведения из строя объектов системы, однако схожими по возможным последствиям могут быть и результаты проявления так называемых недокументированных возможностей вполне «мирного» ПО (например, сетевой ОС),

являющиеся следствием непреднамеренных ошибок, допущенных создателями программно-аппаратных средств. Самым ярким примером проявления недокументированных возможностей является инцидент с «червем» Морриса, первым сетевым компьютерным вирусом. Изначально данная программа предназначалась для удаленного тестирования UNIX-машин, однако после запуска 2 ноября 1988 г. программа вышла из-под контроля автора и начала быстро перемещаться по сети, загружая ОС хостов сети своими копиями и вызывая отказы в обслуживании. Формально данное программное средство не наносило ущерба информации на «зараженных» им хостах, однако вызывало необходимость проведения комплекса профилактических работ по восстановлению работоспособности данных хостов. Общие потери от описанного выше инцидента составили почти 100 млн долл. США.

Таким образом, перед защитой систем обработки информации стоит довольно сложная задача — противодействие бурно развивающимся угрозам безопасности. Следовательно, безопасная, или защищенная, система — это система, обладающая средствами защиты, которые успешно и эффективно противостоят угрозам безопасности.

Главная задача стандартов информационной безопасности — создать основу для взаимодействия между производителями, потребителями и экспертами по квалификации продуктов информационных технологий (ИТ). Каждая из этих групп имеет свои интересы и свои взгляды на проблему информационной безопасности. Таким образом, перед стандартами информационной безопасности стоит непростая задача — примирить взгляды этих сторон и создать эффективный механизм взаимодействия между ними.

Критерии безопасности компьютерных систем Министерства обороны США («Оранжевая книга»). Они были разработаны Министерством обороны США в 1983 г. с целью определения требований безопасности, предъявляемых к аппаратному, программному и специальному обеспечению компьютерных систем и выработки соответствующей методологии и технологии анализа степени поддержки политики безопасности в компьютерных системах экономического назначения.

Согласно «Оранжевой книге», безопасная компьютерная система — это система, поддерживающая управление доступом к обрабатываемой в ней информации так, что только соответствующим образом авторизованные пользователи или процессы, действующие от их имени, получают возможность читать, писать, создавать и удалять информацию.

В «Оранжевой книге» предложены три категории требований безопасности — политика безопасности, аудит и корректность, в рамках которых сформулированы шесть базовых требований безопасности.

Требование 1. *Политика безопасности.* Система должна поддерживать точно определенную политику безопасности.

Требование 2. *Метки.* С объектами должны быть ассоциированы метки безопасности, используемые в качестве атрибутов контроля доступа.

Требование 3. *Идентификация и аутентификация.* Все субъекты должны иметь уникальные идентификаторы.

Требование 4. *Регистрация и учет.* Для определения степени ответственности пользователей за действия в системе все происходящие в ней события, имеющие значение с точки зрения безопасности, должны отслеживаться и регистрироваться в защищенном протоколе.

Требование 5. *Контроль корректности функционирования средств защиты.* Средства защиты должны содержать независимые аппаратные и/или программные компоненты, обеспечивающие работоспособность функций защиты.

Требование 6. *Непрерывность защиты.* Все средства защиты (в том числе и реализующие данное требование) должны быть защищены от несанкционированного вмешательства и/или отключения, причем эта защита должна быть постоянной и непрерывной в любом режиме функционирования системы защиты и компьютерной системы в целом.

Приведенные выше базовые требования к безопасности служат основой для критериев, образующих единую шкалу оценки безопасности компьютерных систем, определяющую семь классов безопасности.

«Оранжевая книга» предусматривает четыре группы критериев, которые соответствуют различной степени защищенности: от минимальной (группа *D)* до формально доказанной (группа *A).* Уровень безопасности возрастает при движении от группы *D* к группе *A,* а внутри группы — с возрастанием номера класса.

Г р у п п а *D.* Минимальная защита.

Класс *D.* Минимальная защита. К этому классу относятся все системы, которые не удовлетворяют

требованиям других классов.

Г р у п п а С. Дискреционная защита.

Класс *C1*. Дискреционная защита. Системы этого класса удовлетворяют требованиям обеспечения разделения пользователей и информации и включают средства контроля и управления доступом, позволяющие задавать ограничения для индивидуальных пользователей, что дает им возможность защищать свою приватную информацию от других пользователей.

Класс *C2*. Управление доступом. Системы этого класса осуществляют более избирательное управление доступом, чем системы класса *C*1, с помощью применения средств индивидуального контроля за действиями пользователей, регистрацией, учетом событий и выделением ресурсов.

Г р у п п а *B*. Мандатная защита.

Класс *B*1. Защита с применением меток безопасности.

Класс *B*2. Структурированная защита. Угроза безопасности системы должна поддерживать формально определенную и четко документированную модель безопасности, предусматривающую произвольное нормативное управление доступом, которое распространяется по сравнению с системами класса *B*1 на все субъекты.

Класс *B*3. Домены безопасности. Угроза безопасности системы должна поддерживать монитор взаимодействий, который контролирует все типы доступа субъектов к объектам и который невозможно обойти. Кроме того, угроза безопасности должна быть структурирована с целью исключения из нее подсистем, не отвечающих за реализацию функций защиты, и быть достаточно компактной для эффективного тестирования и анализа.

Г р у п п а *A*. Верифицированная защита.

Класс *A*1. Формальная верификация. Системы класса *A*1 функционально эквивалентны системам класса *B*3, к ним не предъявляются никакие дополнительные функциональные требования. В отличие от систем класса *B*3 в ходе разработки должны применяться формальные методы верификации, что позволяет с высокой уверенностью получить корректную реализацию функций защиты.

Приведенные классы безопасности надолго определили основные концепции безопасности и ход развития средств защиты.

Для того чтобы исключить возникшую в связи с изменением аппаратной платформы некорректность некоторых положений «Оранжевой книги», адаптировать их к современным условиям и сделать адекватными нуждам разработчиков и пользователей программного обеспечения, и была проделана огромная работа по интерпретации и развитию положений этого стандарта. В результате возник целый ряд сопутствующих «Оранжевой книге» документов, многие из которых стали ее неотъемлемой частью. К наиболее часто упоминаемым относятся:

1) руководство по произвольному управлению доступом в безопасных системах;

2) руководство по управлению паролями;

3) руководство по применению критериев безопасности компьютерных систем в специфических средах.

В 1995 г. Национальным центром компьютерной безопасности США был опубликован документ под названием «Интерпретация критериев безопасности компьютерных систем», объединяющий все дополнения и разъяснения.

Европейские критерии безопасности информационных технологий. Для того чтобы удовлетворить требованиям конфиденциальности, целостности и работоспособности, в Европейских критериях впервые вводится понятие «адекватность средств защиты».

Адекватность включает в себя:

- эффективность, отражающую соответствие средств безопасности решаемым задачам;
- корректность, характеризующую процесс их разработки и функционирования.

Общая оценка уровня безопасности системы складывается из функциональной мощности средств защиты и уровня адекватности их реализации.

Набор функций безопасности может специфицироваться с использованием ссылок на заранее определенные классы-шаблоны. В Европейских критериях таких классов десять. Пять из них *(F-C*1, *F-C2, F-B1, F-B2*, *F-B3)* соответствуют соответствующим классам безопасности «Оранжевой книги» с аналогичными обозначениями. Рассмотрим другие пять классов, так как их требования отражают точку зрения разработчиков стандарта на проблему безопасности.

Класс F-IN предназначен для систем с высокими потребностями в обеспечении целостности, что типично для систем управления базами данных.

Класс F-A V характеризуется повышенными требованиями к обеспечению работоспособности.

Класс F-DI ориентирован на распределенные системы обработки информации.

Класс F-DC уделяет особое внимание требованиям конфиденциальности передаваемой информации.

Класс F-DX предъявляет повышенные требования и к целостности, и к конфиденциальности информации.

Европейские критерии определяют семь уровней адекватности — от *E0* до *E6* (в порядке возрастания). Уровень *EO* обозначает минимальную адекватность (аналог уровня *D* «Оранжевой книги»). При проверке адекватности анализируется весь жизненный цикл системы — от начальной фазы проектирования до эксплуатации и сопровождения. Уровни адекватности от *E1* до *E6* выстроены по нарастанию требований тщательности контроля. Так, на уровне *E1* анализируется общая архитектура системы, а адекватность средств защиты подтверждается функциональным тестированием. На уровне *E3* к анализу привлекаются исходные тексты программ и схемы аппаратного обеспечения. На уровне *E6* требуется формальное описание функций безопасности, общей архитектуры, а также политики безопасности.

Руководящие документы Гостехкомиссии России. В 1992 г. Гостехкомиссия при Президенте РФ опубликовала пять руководящих документов (РД), посвященных вопросам защиты от несанкционированного доступа (НСД) к информации. Важнейшие из них:

1) «Концепция защиты средств вычислительной техники от НСД к информации»;

2) «Средства вычислительной техники. Защита от НСД к информации. Показатели защищенности от НСД к информации»;

3) «Автоматизированные системы. Защита от НСД к информации. Классификация автоматизированных систем и требования по защите информации».

Идейной основой этих документов является «Концепция защиты средств вычислительной техники от НСД к информации», содержащая систему взглядов Гостехкомиссии на проблему информационной безопасности и основные принципы защиты компьютерных систем.

Основная и едва ли не единственная задача средств безопасности в этих документах — это обеспечение защиты от НСД к информации. Если средствам контроля и обеспечения целостности еще уделяется некоторое внимание, то поддержка работоспособности систем обработки информации вообще не упоминается. Все это объясняется тем, что эти документы были разработаны в расчете на применение в информационных системах Министерства обороны Российской Федерации и спецслужб, а также недостаточно высоким уровнем информационных технологий этих систем по сравнению с современным.

Руководящие документы Гостехкомиссии предлагают две группы критериев безопасности:

- показатели защищенности средств вычислительной техники (СВТ) от НСД;
- критерии защищенности автоматизированных систем (АС) обработки данных.

Данный РД устанавливает классификацию СВТ по уровню защищенности от НСД к информации на базе перечня показателей защищенности и совокупности описывающих их требований.

Данные показатели содержат требования защищенности СВТ от НСД к информации и применяются к общесистемным программным средствам и операционным системам. Конкретные перечни показателей определяют классы защищенности СВТ и описываются совокупностью требований.

Установлено семь классов защищенности СВТ от НСД к информации. Самый низкий класс — седьмой, самый высокий — первый.

В отличие от остальных стандартов отсутствует раздел, содержащий требования по обеспечению работоспособности системы, зато присутствует раздел, посвященный криптографическим средствам.

Требования к средствам защиты АС от НСД включают следующие подсистемы.

1. Подсистема управления доступом.
2. Подсистема регистрации и учета.
3. Криптографическая подсистема.
4. Подсистема обеспечения целостности.

Документы Гостехкомиссии устанавливают девять классов защищенности АС от НСД, каждый из которых характеризуется определенной совокупностью требований к средствам защиты. Классы подразделяются на три группы, отличающиеся спецификой обработки информации в АС. Группа АС определяется на основании следующих признаков:

- наличие в АС информации различного уровня конфиденциальности.
- уровень полномочий пользователей АС на доступ к конфиденциальной информации.

- режим обработки данных в АС (коллективный или индивидуальный).

Федеральные критерии безопасности информационных технологий. Это первый стандарт информационной безопасности, в котором определяются три независимые группы требований: функциональные требования к средствам защиты, требования к технологии разработки и к процессу квалификационного анализа. Авторами этого стандарта впервые предложена концепция Профиля защиты — документа, содержащего описание всех требований безопасности как к самому продукту информационных технологий (ИТ-продукту), так и к процессу его проектирования, разработки, тестирования и квалификационного анализа.

Функциональные требования безопасности хорошо структурированы и описывают все аспекты функционирования угрозы безопасности. Требования к технологии разработки, впервые появившиеся в этом документе, побуждают производителей использовать современные технологии программирования как основу для подтверждения безопасности своего продукта.

Разработчики Федеральных критериев отказались от используемого в «Оранжевой книге» подхода к оценке уровня безопасности ИТ-продукта на основании обобщенной универсальной шкалы классов безопасности. Вместо этого предлагается независимое ранжирование требований каждой группы, т.е. вместо единой шкалы используется множество частных шкал-критериев, характеризующих обеспечиваемый уровень безопасности. Данный подход позволяет разработчикам и пользователям ИТ-продукта выбрать наиболее приемлемое решение и точно определить необходимый и достаточный набор требований для каждого конкретного ИТ-продукта и среды его эксплуатации.

Этот стандарт рассматривает устранение недостатков существующих средств безопасности как одну из задач защиты наряду с противодействием угрозам безопасности и реализацией модели безопасности.

Единые критерии безопасности информационных технологий. Представляют собой результат обобщения всех достижений последних лет в области информационной безопасности. Впервые документ такого уровня содержит разделы, адресованные потребителям, производителям и экспертам по квалификации ИТ-продуктов.

Предложенные Едиными критериями механизмы Профиля защиты и Проекта защиты позволяют потребителям и производителям в полной мере выразить свой взгляд на требования безопасности и задачи защиты и дают возможность экспертам по квалификации проанализировать взаимное соответствие между требованиями, нуждами потребителей, задачами защиты и средствами защиты ИТ-продукта.

В отличие от Профиля защиты Федеральных критериев, который ориентирован исключительно на среду применения ИТ-продукта, Профиль защиты Единых критериев предназначен непосредственно для удовлетворения запросов потребителей.

Разработчики Единых критериев отказались (как и разработчики Федеральных критериев) от единой шкалы безопасности и усилили гибкость предложенных в них решений путем введения частично упорядоченных шкал, благодаря чему потребители и производители получили дополнительные возможности по выбору требований и их адаптации к своим прикладным задачам.

Особое внимание этот стандарт уделяет адекватности реализации функциональных требований, которая обеспечивается как независимым тестированием и анализом ИТ-продукта, так и применением соответствующих технологий на всех этапах его проектирования и реализации. Таким образом, требования Единых критериев охватывают практически все аспекты безопасности ИТ-продуктов и технологии их создания, а также содержат все исходные материалы, необходимые потребителям и разработчикам для формирования Профилей и Проектов защиты.

Кроме того, требования Единых критериев являются практически всеобъемлющей энциклопедией информационной безопасности, поэтому их можно использовать в качестве справочника безопасности информационных технологий.

Анализ стандартов информационной безопасности. Главная задача стандартов информационной безопасности — согласовать позиции и цели производителей, потребителей и аналитиков-классификаторов в процессе создания и эксплуатации продуктов информационных технологий. Каждая из перечисленных категорий специалистов оценивает стандарты и содержащиеся в них требования и критерии по своим собственным параметрам.

В качестве обобщенных показателей, характеризующих стандарты информационной безопасности и имеющих значение для всех трех сторон, предлагается использовать универсальность, гибкость, гарантированность, реализуемость и актуальность.

Универсальность. «Оранжевая книга» предназначалась для систем военного времени, ее адаптация

для распределенных систем и баз данных потребовала разработки дополнительных документов.

В Европейские критерии вошли распределенные системы, сети, системы телекоммуникаций и СУБД, но в нем по-прежнему явным образом оговаривается архитектура и назначение систем, к которым он может быть применен, и никак не регламентируется среда их эксплуатации.

Документы Гостехкомиссии имеют довольно ограниченную сферу применения — это персональные и многопользовательские системы, причем ориентация системы на обслуживание конечных пользователей является обязательным условием.

Федеральные критерии подняли область применения стандартов на новый уровень, начав рассматривать в качестве объекта их применения любые продукты информационных технологий независимо от их назначения, проводя различие только между характеристиками среды их эксплуатации.

Канадские критерии рассматривают в качестве области своего применения все типы компьютерных систем.

Единые критерии предложили такую технологию создания ИТ-продуктов, при которой использование данного стандарта является неотъемлемым компонентом.

Гибкость. Требования «Оранжевой книги» оказались слишком абстрактными для непосредственного применения во многих случаях, что потребовало их дополнения.

Европейские критерии предусмотрели специальные уровни и требования, рассчитанные на типовые системы (СУБД, телекоммуникации и т.д.).

Документы Гостехкомиссии подробно регламентируют реализацию функций защиты (например, это единственный стандарт, который в ультимативной форме требует применения криптографии), что значительно снижает удобство их использования — в конкретных ситуациях многие требования часто оказываются избыточными и ненужными.

Федеральные критерии впервые предложили механизм Профилей защиты, с помощью которых можно создавать специальные наборы требований, соответствующие запросам потребителей конкретного продукта и угрозам среды его эксплуатации.

Канадские критерии не рассматривают Профиль защиты в качестве обязательного элемента безопасности информационных технологий, и также обладают определенной спецификой в своем подходе к основным понятиям безопасности, поэтому их гибкость можно оценить только как достаточную.

Единые критерии обладают практически совершенной гибкостью, так как позволяют потребителям выразить свои требования с помощью механизма Профилей защиты, в форме инвариантной к механизмам реализации, а производителям — продемонстрировать с помощью Проекта защиты, как эти требования преобразуются в задачи и реализуются на практике.

Гарантированность. «Оранжевая книга» предусматривала обязательное применение формальных методов верификации только при создании систем высшего класса защищенности (класс *А).*

В Европейских критериях появляется специальный раздел требований — требования адекватности, которые регламентируют технологию и инструментарий разработки, а также контроль за процессами проектирования и разработки.

Документы Гостехкомиссии практически полностью проигнорировали этот ключевой аспект безопасности информационных технологий и обходят данный вопрос молчанием.

Федеральные критерии содержат два специальных раздела требований, посвященных этому аспекту безопасности, содержащие требования к технологии разработки и к процессу квалификационного анализа.

Канадские критерии включают раздел требований адекватности, количественно и качественно ни в чем не уступающий разделу функциональных требований.

Единые критерии рассматривают гарантированность реализации защиты как самый важный компонент информационной безопасности и предусматривают многоэтапный контроль на каждой стадии разработки ИТ-продукта, позволяющий подтвердить соответствие полученных результатов поставленным целям путем доказательства адекватности задач защиты требованиям потребителей, адекватности Проекта защиты Единым критериям и адекватности ИТ-продукта Проекту защиты.

Реализуемость. Плохие показатели реализуемости говорят о практической бесполезности стандарта, поэтому все документы отвечают этому показателю в достаточной или высокой степени.

Единые критерии и здесь оказались на практически недосягаемой для остальных стандартов высоте за счет потрясающей степени подробности функциональных требований (135 требований), фактически

служащих исчерпывающим руководством по разработке защищенных систем.

Отметим, что это единственный показатель, по которому документы Гостехкомиссии не отстают от остальных стандартов информационной безопасности.

Актуальность. «Оранжевая книга» содержит требования, в основном направленные на противодействие угрозам конфиденциальности, что объясняется ее ориентированностью на системы экономического назначения.

Европейские критерии находятся примерно на том же уровне, хотя и уделяют угрозам целостности гораздо больше внимания.

Документы Гостехкомиссии с точки зрения этого показателя выглядят наиболее отсталыми — уже в самом их названии определена единственная рассматриваемая в них угроза — НСД.

Федеральные критерии рассматривают все виды угроз достаточно подробно и предлагают механизм Профилей защиты для описания угроз безопасности, присущих среде эксплуатации конкретного ИТ-продукта, что позволяет учитывать специфичные виды угроз.

Канадские критерии ограничиваются типовым набором угроз безопасности.

Единые критерии ставят во главу угла удовлетворение нужд пользователей и предлагают для этого соответствующие механизмы (Профиль и Проект защиты), что дает возможность выстроить на их основе динамичную и постоянно адаптирующуюся к новым задачам технологию создания безопасных информационных систем.

3.2. Исследование причин нарушений безопасности

Проведение анализа успешно реализовавшихся угроз безопасности (атак) с целью их обобщения, классификации и выявления причин и закономерностей их появления и существования позволяет при разработке и создании защищенных систем сконцентрировать основные усилия именно на устранении этих причин путем исправления выявленных в механизмах защиты недостатков, что позволяет эффективно противостоять угрозам безопасности.

Уязвимость защиты (УЗ) — совокупность причин, условий и обстоятельств, наличие которых в конечном итоге может привести к нарушению нормального функционирования ВС и нарушению безопасности (НСД, ознакомление, уничтожение или искажение данных).

В 70-х гг. XX в. были предприняты попытки формального описания и систематизации информации об УЗ. Исследования проводились по проектам *RISOS* (Исследование безопасности защищенных операционных систем) и *PA* (Анализ защиты).

Предлагаемые методики поиска ошибок безопасности в ОС достаточно ограничены в практическом применении. Это можно объяснить предпринятой попыткой обеспечить универсальность методик, что отрицательно сказалось на возможности их развития и адаптации для новых ОС. Усилия исследователей слишком рано были перенаправлены от изучения УЗ в сторону разработки универсальной технологии создания защищенных ОС, свободных от подобных ошибок.

С точки зрения технологии создания защищенных систем наибольшее значение имеют следующие вопросы, на которые должна дать ответ таксономия УЗ.

1. Каким образом ошибки, приводящие к появлению УЗ, вносятся в систему защиты?
2. Когда, на каком этапе они вносятся?
3. Где, в каких компонентах системы защиты (или ВС в целом) они возникают и проявляются?

Ошибки в системах защиты, служащие источником появления УЗ, могут быть следующими.

I. Преднамеренные.

1. С наличием деструктивных функций (активные):

а) разрушающие программные средства (РПС) (несамовоспроизводящиеся РПС («троянские кони»), самовоспроизводящиеся РПС (вирусы));

б) черные ходы, люки, скрытые возможности проникновения в систему.

2. Без деструктивных функций (пассивные),

а) скрытые каналы утечки информации:

- с использованием памяти (для кодирования передаваемой информации в этом случае используется либо область памяти, не имеющая важного значения (например, установление характеристик признаков в имени и атрибутах файла), либо вообще неиспользуемая область (например, зарезервированные поля в заголовке сетевого пакета));
- с использованием времени (в этом случае информация кодируется определенной

последовательностью и длительностью событий, происходящих в системе (например, с помощью модуляции интервалов обращения к устройствам, введения задержек между приемом и посылкой сетевых пакетов и т.д.));

б) другие (к их появлению обычно приводят расхождения между требованиями безопасности и требованиями к функциональным возможностям ВС).

II. Непреднамеренные.

1. Ошибки контроля допустимых значений параметров.
2. Ошибки определения областей (доменов).
3. Ошибки последовательностей действий и использования нескольких имен для одного объекта.
4. Ошибки идентификации/аутентификации.
5. Ошибки проверки границ объектов.
6. Другие ошибки в логике функционирования.

Этап внедрения ошибки и возникновения УЗ может происходить:

- на стадии разработки (ошибки при проектировании; ошибки при написании программ);
- стадии настройки систем;
- стадии сопровождения;
- стадии эксплуатации.

Классификация УЗ по размещению в системе:

1. Программное обеспечение:

а) операционная система:

- инициализация (загрузка);
- управление выделением памяти;
- управление процессами;
- управление устройствами;
- управление файловой системой;
- средства идентификации и аутентификации;
- другие;

б) сервисные программы и утилиты:

- привилегированные утилиты;
- непривилегированные утилиты;

в) прикладные программы.

2. Аппаратное размещение.

Таксономия причин возникновения УЗ должна дать ответ на имеющий ключевое значение с практической точки зрения вопрос: что явилось причиной успешного осуществления нарушения безопасности в том или ином случае?

Для ответа на этот вопрос необходимо выявить те свойства и особенности архитектуры ВС, которые привели к возможности успешного осуществления соответствующих атак. Только знание природы этих причин позволит оценить способность системы противостоять атакам на ее безопасность, а также понять природу недостатков, присущих существующим средствам обеспечения безопасности, которые привели к соответствующим нарушениям, и построить защищенную систему, лишенную этих недостатков. К причинам нарушения безопасности ВС относятся:

1) предопределенный на стадии разработки требований выбор модели безопасности, не соответствующей назначению или архитектуре ВС;

2) причины, обусловленные принципами организации системы обеспечения безопасности:

- неправильное внедрение модели безопасности;
- отсутствие идентификации и/или аутентификации субъектов и объектов;
- отсутствие контроля целостности средств обеспечения безопасности;

3) причины, обусловленные реализацией:

- ошибки, допущенные в ходе программной реализации средств обеспечения безопасности;
- наличие средств отладки и тестирования в конечных продуктах;

4) ошибки администрирования.

Предложенный подход к классификации причин нарушения безопасности в отличие от существующих подходов позволяет определить полное множество независимых первопричин нарушений безопасности, образующих ортогональное пространство факторов, определяющих реальную

степень безопасности системы.

Сопоставление таксономии причин нарушений безопасности и классификации источников появления УЗ демонстрирует тот факт, что источником появления наибольшего количества категорий УЗ является неправильное внедрение модели безопасности и ошибки в ходе программной реализации. Это означает, что эти причины являются более значимыми и должны быть устранены в первую очередь.

Сопоставление между причинами нарушений безопасности и классификацией УЗ по этапу внесения показывает, что появление основных причин нарушения безопасности закладывается на этапе разработки, причем в основном на стадии задания спецификаций. Это вполне ожидаемый результат, так как именно этап составления спецификаций является одним из самых трудоемких, а последствия ошибок в спецификациях сказываются на всех последующих этапах разработки и распространяются на все взаимосвязанные компоненты системы.

Перекрестный анализ таксономии причин нарушений безопасности и классификация УЗ по источнику появления и этапу внесения показывают, что наиболее значимые причины (неправильное внедрение модели безопасности и ошибки программной реализации) действуют в ходе процесса разработки и реализации.

Следовательно, именно на этих этапах должны быть сосредоточены основные усилия при создании защищенных систем.

3.3. Способы и средства защиты информации

Необходимость обеспечения скрытности (секретности) отдельных замыслов, действий, сообщений возникла в глубокой древности, практически вместе с началом осмысленной человеческой деятельности. Иными словами, организация защиты части информации от нежелательного (несанкционированного) доступа к ней — проблема столь же древняя, как и само понятие «информация».

На современном этапе существуют следующие предпосылки сложившейся кризисной ситуации обеспечения безопасности информационных систем (ИС):

- современные компьютеры за последние годы приобрели большую вычислительную мощность, но одновременно с этим стали гораздо проще в эксплуатации;
- прогресс в области аппаратных средств сочетается с еще более бурным развитием ПО;
- развитие гибких и мобильных технологий обработки информации привело к тому, что практически исчезает грань между обрабатываемыми данными и исполняемыми программами за счет появления и широкого распространения виртуальных машин и интерпретаторов;
- несоответствие бурного развития средств обработки информации и медленной проработки теории информационной безопасности привело к появлению существенного разрыва между теоретическими моделями безопасности, оперирующими абстрактными понятиями типа «объект», «субъект» и реальными категориями современных ИТ;
- необходимость создания глобального информационного пространства и обеспечение безопасности протекающих в нем процессов потребовали разработки международных стандартов, следование которым может обеспечить необходимый уровень гарантии обеспечения защиты.

Вследствие совокупного действия всех перечисленных факторов перед разработчиками современных ИС, предназначенных для обработки конфиденциальной информации, стоят следующие задачи, требующие немедленного и эффективного решения:

- обеспечение безопасности новых типов информационных ресурсов;
- организация доверенного взаимодействия сторон (взаимной идентификации/аутентификации) в информационном пространстве;
- защита от автоматических средств нападения;
- интеграция в качестве обязательного элемента защиты информации в процессе автоматизации ее обработки.

В настоящее время с ростом объемов обработки информации в компьютерных сетях, расширением круга ее потребителей, распространением многопрограммных режимов работы ЭВМ, внедрением перспективных ИТ данная проблема приобрела новый аспект в связи с возрастанием роли программно-технического посредника между человеком-пользователем и информационными объектами, что, в свою очередь, вызвало создание дополнительных способов закрытия информации.

Понятно, что проблема защиты информации приобретает особое значение в экономической области, характеризующейся повышенными требованиями одновременно к скрытности и оперативности обработки информации.

Учитывая наибольшую сложность обеспечения защиты информации, обрабатываемой и передаваемой в компьютерных сетях различных типов, в дальнейшем рассматривается прежде всего эта часть общей проблемы (если иное не будет оговорено специально).

Анализ уязвимости машинной информации позволяет выделить две группы возможных причин ее искажения или уничтожения:

- непреднамеренные действия (сбои технических средств, ошибки обслуживающего персонала и пользователей, аварии и т.п.);
- несанкционированные действия, к которым относятся: НСД и ознакомление субъектов с информацией; прямое хищение информации в электронном виде непосредственно на носителях или копирование информации на другие носители; запрещенная передача информации в линии связи или на терминалы; перехват электромагнитных излучений и информации по различным каналам связи и т. п.

Понятно, что такое большое число причин искажения (уничтожения) информации определяет необходимость использования и значительного числа способов и средств ее защиты, причем эти способы и средства только тогда эффективны, когда они применяются комплексно. Названные обстоятельства обусловливают содержание понятия «защита электронной информации».

Под *защитой информации в компьютерных системах* принято понимать создание и поддержание организованной совокупности средств, способов, методов и мероприятий, предназначенных для предупреждения искажения, уничтожения и несанкционированного использования информации, хранимой и обрабатываемой в электронном виде [54].

Наряду с определением понятия «защита информации» важным вопросом является классификация имеющихся способов и средств защиты, которые позволяют воспрепятствовать запрещенному (незаконному) ее использованию. На рис. 3.1 приведены наиболее часто используемые *способы защиты информации* в компьютерных сетях и средства, которыми они могут быть реализованы (на рисунке эти возможности изображены стрелками от способа к средствам).

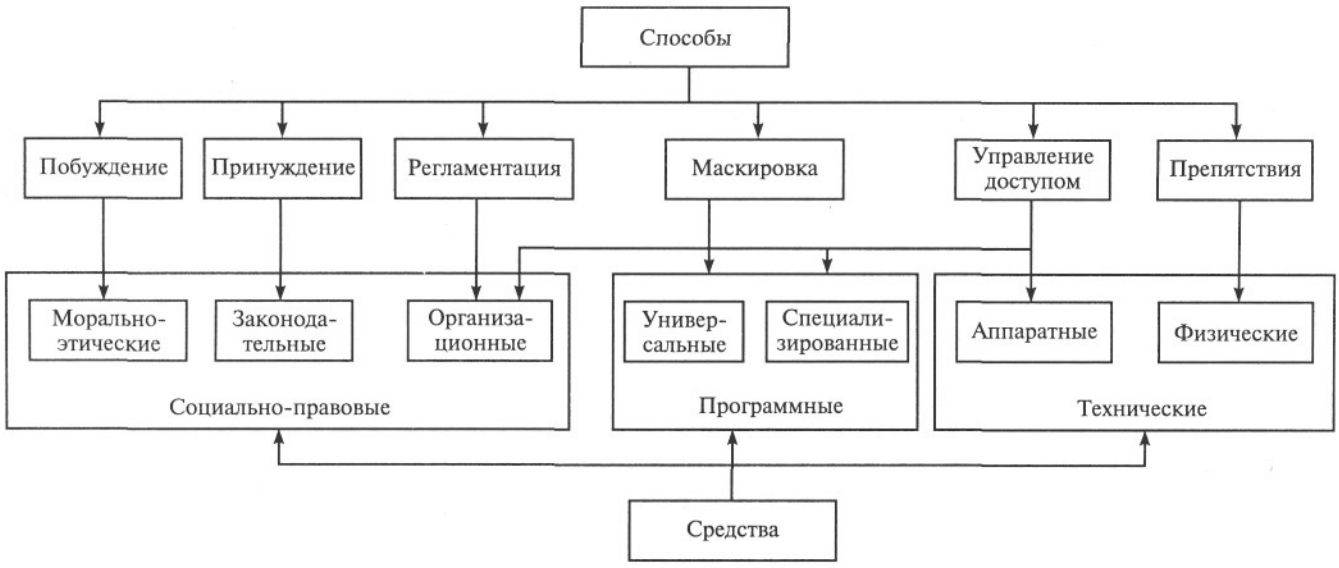


Рис. 3.1. Способы и средства защиты информации

Рассмотрим каждый из способов. П р е п я т с т в и я предусматривают создание преград, физически не допускающих к информации. У п р а в л е н и е д о с т у п о м — способ защиты информации за счет регулирования использования всех ресурсов системы (технических, программных, временных и др.). М а с к и р о в к а информации, как правило, осуществляется путем ее криптографического закрытия. Р е г л а м е н т а ц и я заключается в реализации системы организационных мероприятий, определяющих все стороны обработки информации. П р и н у ж д е н и е заставляет соблюдать

определенные правила работы с информацией под угрозой материальной, административной или уголовной ответственности. Наконец, п о б у ж д е н и е основано на использовании действенности морально-этических категорий (например, авторитета или коллективной ответственности).

Средства защиты информации, хранимой и обрабатываемой в электронном виде, разделяют на три самостоятельные группы: технические, программные и социально-правовые. В свою очередь, среди технических средств защиты выделяют физические и аппаратные.

К ф и з и ч е с к и м с р е д с т в а м защиты относятся:

- механические преграды, турникеты (заграждения); специальное остекление;
- сейфы, шкафы;
- механические и электромеханические замки, в том числе с дистанционным управлением;
- замки с кодовым набором;
- датчики различного типа;
- теле- и фотосистемы наблюдения и регистрации;
- СВЧ, ультразвуковые, радиолокационные, лазерные, акустические системы и др.;
- устройства маркировки;
- устройства с идентификационными картами;
- устройства идентификации по физическим признакам;
- устройства пространственного заземления;
- системы физического контроля доступа;
- системы охранного телевидения и охранной сигнализации;
- системы пожаротушения и оповещения о пожаре и др.

Под а п п а р а т н ы м и с р е д с т в а м и з а щ и т ы понимают технические устройства, встраиваемые непосредственно в системы (аппаратуру) обработки информации. Наиболее часто используют:

- регистры хранения реквизитов защиты (паролей, грифов секретности и т.п.);
- устройства для измерения индивидуальных характеристик человека (например, цвета и строения радужной оболочки глаз, овала лица и т.д.);
- схемы контроля границ адреса имен для определения законности обращения к соответствующим полям (областям) памяти и отдельным программам;
- схемы прерывания передачи информации в линии связи с целью периодического контроля адресов выдачи данных;
- экранирование ЭВМ;
- установка генераторов помех и др.

П р о г р а м м н ы е с р е д с т в а з а щ и т ы данных в настоящее время получили значительное развитие. По целевому назначению их можно разделить на несколько больших классов (групп):

- программы идентификации пользователей;
- программы определения прав (полномочий) пользователей (технических устройств);
- программы регистрации работы технических средств и пользователей (ведение так называемого системного журнала);
- программы уничтожения (затирания) информации после решения соответствующих задач или при нарушении пользователем определенных правил обработки информации;
- криптографические программы (программы шифрования данных).

Программные средства защиты информации часто делят на средства, реализуемые в стандартных ОС (универсальные), и средства защиты в специализированных ИС (специализированные). К первым следует отнести:

- динамическое распределение ресурсов и запрещение задачам пользователей работать с «чужими» ресурсами;
- разграничение доступа пользователей к ресурсам по паролям;
- разграничение доступа к информации по ключам защиты;
- защита таблицы паролей с помощью главного пароля и др.

Средства защиты в экономических ИС, в том числе банковских системах, позволяют реализовать следующие функции защиты данных:

- опознавание по идентифицирующей информации пользователей и элементов ИС, разрешение на

этой основе работы с информацией на определенном уровне;
- ведение многоразмерных таблиц профилей доступа пользователей к данным;
- управление доступом по профилям полномочий;
- уничтожение временно фиксируемых областей информации при завершении ее обработки;
- формирование протоколов обращений к защищаемым данным с идентификацией данных о пользователе и временных характеристик;
- программная поддержка работы терминала лица, отвечающего за безопасность информации;
- подача сигналов при нарушении правил работы с системой или правил обработки информации;
- физическая или программная блокировка возможности работы пользователя при нарушении им определенной последовательности правил или совершении определенных действий;
- подготовка отчетов о работе с различными данными — ведение подробных протоколов работы и др.

Криптографические программы основаны на использовании методов шифрования (кодирования) информации. Данные методы остаются достаточно надежными средствами ее защиты и более подробно будут рассмотрены ниже.

В перспективе можно ожидать развития программных средств защиты по двум основным направлениям:
- создание централизованного ядра безопасности, управляющего всеми средствами защиты информации в ЭВМ (на первом этапе — в составе ОС, затем — вне ее);
- децентрализация защиты информации вплоть до создания отдельных средств, управляемых непосредственно только пользователем. В рамках этого направления находят широкое применение методы «эстафетной палочки» и «паспорта», основанные на предварительном расчете специальных контрольных кодов (по участкам контролируемых программ) и их сравнении с кодами, получаемыми в ходе решения задачи.

Организационные и законодательные средства защиты информации предусматривают создание системы нормативно-правовых документов, регламентирующих порядок разработки, внедрения и эксплуатации информации, а также ответственность должностных и юридических лиц за нарушение установленных правил, законов, приказов, стандартов и т.п.

Морально-этические средства защиты информации основаны на использовании моральных и этических норм, господствующих в обществе, и требуют от руководителей всех рангов особой заботы о создании в коллективах здоровой нравственной атмосферы.

3.4. Формальные модели безопасности

Наибольшее развитие получили два подхода, каждый из которых основан на своем видении проблемы безопасности и нацелен на решение определенных задач, — это формальное моделирование политики безопасности и криптография. Причем эти различные по происхождению и решаемым задачам подходы дополняют друг друга: криптография может предложить конкретные методы защиты информации в виде алгоритмов идентификации, аутентификации, шифрования и контроля целостности, а формальные модели безопасности предоставляют разработчикам основополагающие принципы, лежащие в основе архитектуры защищенной системы и определяющие концепцию ее построения.

Модель политики безопасности — формальное выражение политики безопасности. Формальные модели используются достаточно широко, потому что только с их помощью можно доказать безопасность системы, опираясь при этом на объективные и неопровержимые постулаты математической теории.

Основная цель создания политики безопасности ИС и описания ее в виде формальной модели — это определение условий, которым должно подчиняться поведение системы, выработка критерия безопасности и проведение формального доказательства соответствия системы этому критерию при соблюдении установленных правил и ограничений. На практике это означает, что только соответствующим образом уполномоченные пользователя получат доступ к информации и смогут осуществлять с ней только санкционированные действия.

Среди моделей политик безопасности можно выделить два основных класса: дискреционные (произвольные) и мандатные (нормативные). В данном подразделе в качестве примера изложены

основные положения наиболее распространенных политик безопасности, основанных на контроле доступа субъектов к объектам.

Дискреционная модель Харрисона—Руззо — Ульмана. Модель безопасности Харрисона—Руззо—Ульмана, являющаяся классической дискреционной моделью, реализует произвольное управление доступом субъектов к объектам и контроль за распространением прав доступа.

В рамках этой модели система обработки информации представляется в виде совокупности активных сущностей — субъектов (множество *S),* которые осуществляют доступ к информации, пассивных сущностей — объектов (множество *O),* содержащих защищаемую информацию, и конечного множества прав доступа $R = \{r_1, ..., r_n\}$, означающих полномочия на выполнение соответствующих действий (например, чтение, запись, выполнение).

Причем для того чтобы включить в область действия модели и отношения между субъектами, принято считать, что все субъекты одновременно являются и объектами. Поведение системы моделируется с помощью понятия «состояние». Пространство состояний системы образуется декартовым произведением множеств составляющих ее объектов, субъектов и прав — *OSR.* Текущее состояние системы *Q* в этом пространстве определяется тройкой, состоящей из множества субъектов, множества объектов и матрицы прав доступа *M,* описывающей текущие права доступа субъектов к объектам, — *Q = (S, O, M).* Строки матрицы соответствуют субъектам, а столбцы — объектам, поскольку множество объектов включает в себя множество субъектов, матрица имеет вид прямоугольника. Любая ячейка матрицы *M*[*s, o*] содержит набор прав субъекта *s* к объекту *o,* принадлежащих множеству прав доступа *R.* Поведение системы во времени моделируется переходами между различными состояниями. Переход осуществляется путем внесения изменений в матрицу *M* с помощью команд следующего вида:

command a (x_1, ..., x_k)
If r_1 in $M[x_{S_1}, x_{O_1}]$ and (*условия выполнения команды*)
r_1 in $M[x_{S_2}, x_{O_2}]$ and
.
.
r_m in $M[x_{S_m}, x_{O_m}]$
then
op_1, op_2, ..., op_n (*операции, составляющие команду*)

Здесь *a* — имя команды; x_i — параметры команды, являющиеся идентификаторами субъектов и объектов; s_i и o_i — индексы субъектов и объектов в диапазоне от 1 до *k;* op_i — элементарные операции.

Элементарные операции, составляющие команду, выполняются только в том случае, если все условия, означающие присутствие указанных прав доступа в ячейках матрицы *M,* являются истинными. В классической модели допустимы только следующие элементарные операции:

enter *r* into *M[s, o] (добавление субъекту s права r для объекта o)*
delete *r* from *M[s, o] (удаление у субъекта s права r для объекта o)*
create subject *s (создание нового субъекта s)*
create object *o (создание нового объекта o)*
destroy subject *s (удаление существующего субъекта s)*
destroy object *o (удаление существующего объекта o)*

Критерий безопасности модели Харрисона — Руззо — Ульмана формулируется следующим образом.

Для заданной системы начальное состояние $Q_0 = (S_0, O_0, M_0)$ является безопасным относительно права *r,* если не существует применимой к Q_0 последовательности команд, в результате которой право *r* будет занесено в ячейку матрицы *M,* в которой оно отсутствовало в состоянии Q_0.

Нужно отметить, что все дискреционные модели уязвимы по отношению к атаке с помощью «троянского коня», поскольку в них контролируются только операции доступа субъектов к объектам, а не потоки информации между ними. Поэтому, когда «троянская» программа, которую нарушитель подсунул некоторому пользователю, переносит информацию из доступного этому объекта в объект, доступный нарушителю, то формально никакое правило дискреционной политики безопасности не нарушается, но утечка информации происходит.

Таким образом, дискреционная модель Харрисона—Руззо — Ульмана в своей общей постановке не

дает гарантий безопасности системы, однако именно она послужила основой для целого класса моделей политик безопасности, которые используются для управления доступом и контроля за распределением прав во всех современных системах.

Типизованная матрица доступа. Другая дискреционная модель, получившая название «Типизованная матрица доступа» (*Type Access Matrix* — далее *TAM),* представляет собой развитие модели Харрисона—Руззо—Ульмана, дополненной концепцией типов, что позволяет несколько смягчить те условия, для которых возможно доказательство безопасности системы.

Состояние системы описывается четверкой $Q = (S, O, t, M)$, где S, O и M обозначают соответственно множество субъектов, объектов и матрицу доступа, а $t: O \rightarrow T$ — функция, ставящая в соответствие каждому объекту некоторый тип.

Состояние системы изменяется с помощью команд из множества C. Команды *TAM* имеют тот же формат, что и в модели Харрисона—Руззо—Ульмана, но всем параметрам приписывается определенный тип:

command a($x_1: t_1$, $x_k: t_k$)
  If r_1 in $M[x_{S_1}, x_{O_1}]$ and
    r_1 in $M[x_{S_2}, x_{O_2}]$ and
  .
  .
  r_m in $M[x_{S_m}, x_{O_m}]$
  then
  $op_1, op_2, ..., op_n$

(*условия выполнения команды*)

(*операции, составляющие команду*)

Перед выполнением команды происходит проверка типов фактических параметров, и, если они не совпадают с указанными в определении, команда не выполняется. Фактически введение контроля типов для параметров команд приводит к неявному введению дополнительных условий, так как команды могут быть выполнены только при совпадении типов параметров. В модели используются следующие шесть элементарных операций, отличающихся от аналогичных операций модели Харрисона—Руззо — Ульмана только использованием типизованных параметров:

enter r into $M[s, o]$
create subject s of type t
create object o of type t

(*монотонные операции*)

delete r from $M[s, o]$
destroy subject s
destroy object o

(*немонотонные операции*)

Смысл элементарных операций совпадает со смыслом аналогичных операций их классической модели Харрисона — Руззо — Ульмана с точностью до использования типов.

Таким образом, *TAM* является обобщением модели Харрисона—Руззо—Ульмана, которую можно рассматривать как частный случай *TAM* с одним-единственным типом, к которому относятся все объекты и субъекты. Появление в каждой команде дополнительных неявных условий, ограничивающих область применения команды только сущностями соответствующих типов, позволяет несколько смягчить жесткие условия классической модели, при которых критерий безопасности является разрешимым.

Несмотря на различающиеся подходы, суть всех моделей безопасности одинакова, поскольку они предназначены для решения одних и тех же задач. Целью построения модели является получение формального доказательства безопасности системы, а также определения достаточного критерия безопасности.

Безопасность системы определяется в равной степени тремя факторами: свойствами самой модели, ее адекватностью угрозам, воздействующим на систему, и тем, насколько корректно она реализована. Поскольку при существующем разнообразии теоретических наработок в области теории информационной безопасности выбор модели, адекватной заданным угрозам, не является проблемой, последнее, решающее слово остается за ее реализацией в защищенной системе.

3.5. Шифрование — специфический способ защиты информации

Шифрование информации, хранимой и обрабатываемой в электронном виде, — это нестандартная кодировка данных, исключающая или серьезно затрудняющая возможность их прочтения (получения в открытом виде) без соответствующего программного или аппаратного обеспечения и, как правило, требующая для открытия данных предъявления строго определенного ключа (пароля, карты, отпечатка и т.д.). Шифрование условно объединяет четыре аспекта защиты информации: управление доступом, регистрацию и учет, криптографическую защиту, обеспечение целостности информации. Оно включает в себя непосредственное шифрование информации, электронную подпись и контроль доступа к информации.

Шифрование. Оно направлено на достижение четырех основных целей.

1. Статическая защита информации, хранящейся на жестком диске компьютера или дискетах (шифрование файлов, фрагментов файлов или всего дискового пространства), исключает или серьезно затрудняет доступ к информации лицам, не владеющим паролем (ключом), т. е. защищает данные от постороннего доступа в отсутствие владельца информации. Статическое шифрование применяется в целях информационной безопасности на случай похищения файлов, дискет или компьютеров целиком (жестких дисков компьютеров) и исключения возможности прочтения данных любыми посторонними (не владеющими паролем) лицами. Наиболее продвинутой формой статической защиты информации является прозрачное шифрование (рис. 3.2), при котором данные, попадающие на защищенный диск, автоматически шифруются (кодируются) вне зависимости от природы операции записи, а при считывании с диска в оперативную память автоматически дешифрируются, так что пользователь вообще не ощущает, что находится под неусыпной защитой невидимого стража информации.

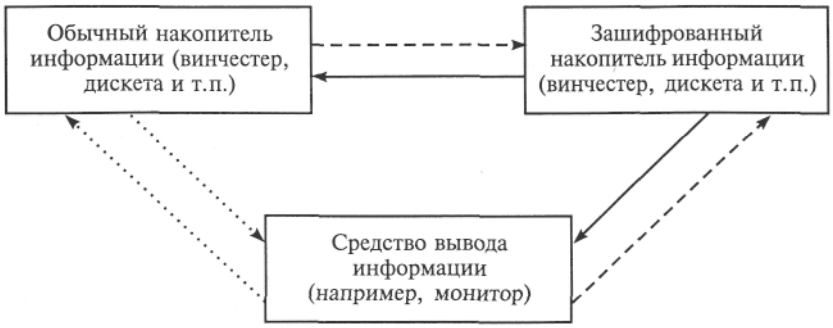


Рис. 3.2. Общая схема прозрачной дисковой защиты:
- - → шифрование; ⟶ расшифровка; ····→ передача без изменений

2. Разделение прав и контроль доступа к данным. Пользователь может владеть своими личными данными (разными компьютерами, физическими или логическими дисками одного компьютера, просто разными директориями и файлами), не доступными другим пользователям.

3. Защита отправляемых (передаваемых) данных через третьи лица, в том числе по электронной почте или в рамках локальной сети.

4. Идентификация подлинности (аутентификация) и контроль целостности переданных через третьи лица документов.

Шифровальные методы подразделяются на два принципиальных направления:

- симметричные классические методы с секретным ключом, в которых для зашифровки и дешифрации требуется предъявление одного и того же ключа (пароля);
- асимметричные методы с открытым ключом, в которых для зашифровки и дешифрации требуется предъявление двух различных ключей, один из которых объявляется секретным (приватным), а второй — открытым (публичным), причем пара ключей всегда такова, что по публичному невозможно восстановить приватный, и ни один из них не подходит для решения обратной задачи.

Как правило, шифрование производится путем выполнения некоторой математической (или логической) операции (серии операций) над каждым блоком битов исходных данных (так называемая криптографическая обработка). Применяются также методы рассеивания информации, например обыкновенное разделение данных на нетривиально собираемые части, или стеганография, при которой исходные открытые данные размещаются определенным алгоритмом в массиве случайных данных, как бы растворяясь в нем. От произвольной трансформации данных шифрование отличается тем, что

выполняемое им преобразование всегда обратимо при наличии симметричного или асимметричного ключа дешифрации.

Идентификация подлинности и контроль целостности основываются на том, что дешифрация данных с определенным ключом возможна только в случае, если они были зашифрованы с соответствующим (тем же или парным) ключом и не подверглись изменению в зашифрованном виде. Таким образом, если в случае симметричного метода обеспечена секретность (уникальность) двух копий одного ключа, а в случае асимметричного метода — секретность (уникальность) одного из пары ключей, успех операции дешифрации данных гарантирует их подлинность и целостность (разумеется, при условии надежности используемого метода и чистоты его программной или аппаратной реализации).

Шифрование — наиболее общий и надежный при достаточном качестве программной или аппаратной системы способ защиты информации, обеспечивающий практически все его аспекты, включая разграничение прав доступа и идентификацию подлинности (электронную подпись). Однако существует два обстоятельства, которые необходимо учитывать при использовании программных средств, реализующих данное направление. Во-первых, любое зашифрованное сообщение в принципе всегда может быть расшифровано (хотя время, затрачиваемое на это, подчас делает результат расшифровки практически бесполезным). Во-вторых, перед непосредственной обработкой информации и выдачей ее пользователю производится расшифровка — при этом информация становится открытой для перехвата.

С точки зрения качества защиты информации шифрование можно условно разделить на «сильное», или «абсолютное», практически не вскрываемое без знания пароля, и «слабое», затрудняющее доступ к данным, но практически (при использовании современных ЭВМ) вскрываемое тем или иным способом за реальное время без знания исходного пароля. Способы вскрытия информации в современных компьютерных сетях включают:

- подбор пароля или рабочего ключа шифрования перебором *(brute-force attack);*
- угадывание пароля *(key-guessing attack);*
- подбор или угадывание пароля при известной части пароля;
- взлом собственно алгоритма шифрования.

Вне зависимости от метода шифрования любой шифр является слабым (т. е. вскрываемым за реальное время), если длина пароля недостаточно велика. Приводимые в табл. 3.1 данные показывают время, требуемое на подбор пароля на ЭВМ класса Pentium/200 МГц в зависимости от длины пароля и допустимых при его формировании знаков при вскрытии информации.

Таблица 3.1

Время на подбор пароля на ЭВМ Pentium/200 МГц

Состав пароля	Число знаков пароля						
	4	5	6	7	8	9	10
Только цифры	0 с	0,01 с	0,08 с	0,83 с	8 с	4 мин	14 мин
Латинские буквы без учета регистра	0,04 с	0,9 с	25 с	12 мин	4,9 ч	5,2 дня	0,4 лет
Латинские буквы без учета регистра и цифры	0,14 с	5,5 с	3 мин	1,8 ч	2,7 дня	0,27 лет	9,7 лет
Латинские буквы с учетом регистра и цифры	1,2 с	1,3 мин	1,3 ч	3,4 дня	0,58 лет	35,7 лет	2220 лет
Все возможные символы	6 мин	1,06 дня	0,74 лет	190 лет	48,7 тыс. лет	12 млн лет	3,2 млрд лет

В зависимости от сложности применяемого алгоритма указанные времена могут быть увеличены в фиксированное число раз (в среднем в 10—1000). Микропроцессор Pentium II/450 МГц или даже Pentium III превосходят Pentium/200 МГц по производительности не более чем в 10 раз, использование суперЭВМ (например, «Эльбрус») позволяет сократить время перебора не более чем в 10000 раз, что, учитывая порядок приведенных в таблице чисел, абсолютно непринципиально.

Таким образом, если пароль включает только латинские буквы без различения регистра, то любой шифр является слабым при длине пароля менее 10 знаков (очень слабым при длине пароля менее 8 знаков); если пароль включает только латинские буквы с различением регистра и цифры, то шифр является слабым при длине пароля менее 8 знаков (очень слабым при длине пароля менее 6 знаков); если же допускается использование всех возможных 256 знаков, то шифр является слабым при длине пароля менее 6 знаков.

Однако длинный пароль сам по себе еще не означает высокой степени защиты, поскольку защищает данные от взлома подбором пароля, но не угадыванием. Угадывание пароля основано на специально разработанных таблицах ассоциации, построенных на статистических и лингвопсихологических свойствах словообразования, словосочетаний и буквосочетаний того или иного языка, и способно на порядки сократить пространство полного перебора. Так, если для подбора пароля «Мама мыла раму» полным перебором требуются миллиарды лет на сверхмощных ЭВМ, то угадывание этого же пароля по таблицам ассоциации займет считанные дни или даже часы.

Подбор или угадывание пароля при известной части пароля также существенно упрощает взлом. Например, зная особенности работы человека за компьютером или видя (или даже слыша) издали, как он набирает пароль, можно установить точное число знаков пароля и приблизительные зоны клавиатуры, в которых нажимаются клавиши. Такие наблюдения также могут сократить время подбора с миллиардов лет до нескольких часов.

Даже если примененный пароль и рабочий ключ достаточно сложны, возможность взлома алгоритма шифрования поистине не знает границ. Из наиболее известных подходов можно выделить:

- математическое обращение применяемого метода;
- взлом шифра по известным парам открытых и соответствующих закрытых данных (метод *plaintext attack);*
- поиск особых точек метода (метод *singularity attack)* — дублирующих ключей (различных ключей, порождающих одинаковые вспомогательные информационные массивы при шифровании различных исходных данных), вырожденных ключей (порождающих тривиальные или периодические фрагменты вспомогательных информационных массивов при шифровании различных исходных данных), а также вырожденных исходных данных;
- статистический, в частности дифференциальный, анализ — изучение закономерностей зашифрованных текстов и пар открытых/зашифрованных текстов.

Наиболее привычным и доступным каждому пользователю средством шифрования информации, хранимой и обрабатываемой в электронном виде, являются программы-архиваторы, как правило содержащие встроенные средства шифрования.

Согласно проведенным исследованиям [6, 21], максимальный рейтинг по степени сжатия и скорости имеет архиватор RAR, незначительно отстает от него архиватор PKZ1P (несколько худшая компрессия при выдающейся скорости).

Защита данных с помощью электронной подписи. Электронная подпись — вставка в данные (добавление) фрагмента инородной зашифрованной информации. Применяется для идентификации подлинности переданных через третьи лица документов и произвольных данных. Сама передаваемая информация при этом никак не защищается, т.е. остается открытой и доступной для ознакомления тем лицам, через которых она передается (например, администраторам сети и инспекторам почтовых узлов электронной связи).

Как правило, электронная подпись включает в себя специальным образом вычисляемую контрольную сумму от данных, с которыми она соотносится, за счет чего обеспечивается контроль целостности данных.

В электронных подписях может использоваться симметричное шифрование, однако по сложившейся традиции почти все системы электронной подписи базируются на шифровании с открытым ключом. В этом случае для зашифрования контрольной суммы от данных применяется секретный ключ пользователя, публичный ключ дешифрации может быть добавлен непосредственно к подписи, так что вся информация, необходимая для аутентификации и контроля целостности данных, может находиться в одном (передаваемом) «конверте».

Достоверность собственно электронной подписи целиком и полностью определяется качеством шифрующей системы. Однако на самом деле с электронной подписью все не так просто, и число уязвимых точек электронной подписи, базирующейся на шифровании с открытым ключом, также велико. С точки зрения решения задачи идентификации подлинности и контроля целостности

полностью зашифрованный файл и открытый файл с добавочной зашифрованной информацией, включающей контрольную сумму данных («электронной подписью»), абсолютно эквивалентны.

Шифрование для обеспечения контроля прав доступа. *Контроль права доступа* — простейшее средство защиты данных и ограничения (разграничения) использования компьютерных ресурсов, предназначенное для ограждения паролем определенной информации и системных ресурсов ЭВМ от лиц, не имеющих к ним отношения и не имеющих специального умысла получить к ним доступ или не обладающих достаточной для этого квалификацией. Сами данные хранятся на дисках в открытом (незащищенном) виде и всегда могут быть востребованы (похищены) в обход системы контроля, сколь бы изощренной она ни была.

Примерами систем, осуществляющих парольный контроль доступа, являются системы Norton's partition security system, Stacker, Fastback, Quicken, Microsoft Money, системы парольного контроля доступа при загрузке BIOS и т.д. Слабые шифры, реализуемые в известных программах Norton's Diskreet, PKZIP, Unix crypt, Novell Netware, MS Excel, MS Word и другие, для которых известны эффективные способы взлома, также можно отнести к системам контроля доступа.

Несмотря на богатый научный потенциал России в области криптографии и особенно бурное ее развитие в начале 90-х гг. прошлого века, на настоящий момент единственным лицензированным Федеральным агентством правительственной связи и информации (ФАПСИ) шифром является шифр по ГОСТ 28147 — 89, самому же ФАПСИ и принадлежащий. Все остальные системы шифрования, предлагаемые зарубежными и отечественными фирмами (системы Symantec, RSA Data Security, AT&T, PGP, ЛАН Крипто, Аладдин, Novex, Элиас, Анкад и многие другие) в виде законченных продуктов или библиотек, начиная с устоявшихся зарубежных стандартов (алгоритмов шифрования DES, FEAL, IDEA) и кончая оригинальными новейшими разработками, являются в равной степени незаконными и подводят наиболее активных инициаторов их разработки и использования на грань уголовной ответственности. Право на хождение на территории России имеет только указанный ГОСТ, причем только в исполнении организации, обладающей сертификатом ФАПСИ.

Что же касается непосредственно надежности шифрования, то практически все используемые коммерческие и индивидуально разработанные алгоритмы шифрования являются слабыми. Кроме того, существуют коммерческие и некоммерческие версии дешифраторов для всех известных архиваторов (PKZIP, arj и др.). Зарубежные «стандарты» шифрования (с учетом многообразия предлагаемых модификаций), экспортируемые некоторыми технологически развитыми странами (в частности, США — алгоритм DES, Япония — алгоритм FEAL), на самом деле являются стандартами соответствующих разведслужб, предлагаемыми и внедряемыми на территориях дружеских государств. Исключения в списке заведомо ненадежных систем шифрования, потенциально доступных для пользователя, являются лишь некоторые — две или три — оригинальные российские разработки.

Разделение систем шифрозащиты на сильные и слабые (как по длине используемого пароля, так и по надежности самой системы) имеет принципиальное значение, обусловливающее возможность реального применения как слабых, так и сильных шифров в условиях их юридического запрета. Дело в том, что если используется заведомо слабая шифрозащита (например, программа PKZIP с паролем), для которой существует эффективный взлом, то невозможно наверняка утверждать, что выбранное средство является криптосистемой. Скорее, речь идет о шифрообразном ограничении и контроле прав доступа. С другой стороны, любая программа шифрования может потенциально рассматриваться как слабый шифр, т.е. шифрообразный контроль доступа к данным. Наконец, каким бы шифром вы ни пользовались, применение коротких паролей, безусловно, переводит шифры в разряд слабых, не обеспечивающих должный уровень защиты информации.

3.6. Защита информации от компьютерных вирусов

Рассмотренные в предыдущих пунктах способы и средства защиты информации являются в значительной мере универсальными и могут быть использованы в компьютерных сетях с любой машинной базой и для любой операционной платформы. Вместе с тем существует необходимость разработки совершенно новых средств защиты информации, предназначенных для противодействия так называемым компьютерным вирусам, способным уничтожать или искажать информацию, обрабатываемую на персональных ЭВМ. Важность и значимость таких методов определяются несколькими основными факторами. Во-первых, персональные ЭВМ получили весьма широкое распространение во всех отраслях человеческой деятельности, в том числе в экономической, причем

круг непосредственных пользователей машин постоянно расширяется и в скором времени охватит, очевидно, практически все трудоспособное население. Во-вторых, к настоящему времени разработаны разнообразные программные средства, существенно облегчающие использование новых информационных технологий, и расширилась область их сбыта и применения, что в ряде случаев способствует распространению «зараженных» программных продуктов. В-третьих, появилось значительное число программистов, способных самостоятельно создавать достаточно сложные версии компьютерных вирусов и по различным причинам делающих это (в качестве причин могут выступать и весьма необычные — от желания испытать свои силы, прославиться, сделать рекламу своей квалификации до сознательных попыток сорвать работу тех или иных ИС).

Следует отметить, что проблема борьбы с компьютерными вирусами возникла тогда, когда программисты стали создавать и использовать свои программы в некоторой системной среде, т.е. стали в определенном смысле по отношению друг к другу обезличенными. Также необходимо иметь в виду, что разработка компьютерных вирусов (и соответственно способов и средств борьбы с ними) не является вопросом науки, а имеет чисто инженерный («программистский») уровень.

История компьютерных вирусов ведет начало с 1984 г., когда Фред Коэн (США) написал программу, которая автоматически активизировалась и проводила деструктивные изменения информации при незаконном обращении к другим программам автора. После этого было создано большое количество программ, предназначенных для вызова тех или иных «вредных» действий, и постепенно сформировалось самостоятельное инженерное направление по борьбе с ними.

Компьютерные вирусы были и остаются одной из наиболее распространенных причин потери информации. Данное положение особенно важно для локальных и глобальных компьютерных сетей, объединяющих работу сотен и миллионов пользователей. Известны случаи, когда вирусы блокировали работу организаций и предприятий. Более того, несколько лет назад был зафиксирован случай, когда компьютерный вирус стал причиной гибели человека — в одном из госпиталей Нидерландов пациент получил летальную дозу морфия по той причине, что компьютер был заражен вирусом и выдавал неверную информацию.

Несмотря на огромные усилия конкурирующих между собой антивирусных фирм, убытки, приносимые компьютерными вирусами, не падают и достигают астрономических величин, измеряемых сотнями миллионов долларов ежегодно. Особенно эти потери актуальны в больших компьютерных сетях. Так, например, в апреле 1999 г. только по причине деструктивного действия одного компьютерного вируса под названием Win95.CIH во Франции было поражено более 100 тыс. банковских компьютеров с общим ущербом более 300 млн долл. Надо полагать, что эти оценки явно занижены, поскольку в средства массовой информации попадает лишь часть сведений о подобных инцидентах.

При этом следует иметь в виду, что используемые антивирусные программы и аппаратная часть не дают полной гарантии защиты от вирусов. Примерно так же плохо обстоят дела на другой стороне тандема «человек—компьютер». Как прикладные пользователи, так и профессионалы-программисты часто не имеют даже навыков «самообороны» от вирусов, а их представления о проблеме порой весьма и весьма поверхностны.

Что же представляет собой компьютерный вирус? *Компьютерный вирус* — это специально написанная, как правило, на языке программирования низкого уровня небольшая по размерам программа, которая может «приписывать» себя к другим программам и выполнять различные нежелательные для пользователя действия на компьютере.

Жизненный цикл вируса включает четыре основных этапа [53]:

- внедрение;
- инкубационный период (прежде всего для скрытия источника проникновения);
- «репродуцирование (саморазмножение);
- деструкция (искажение и/или уничтожение информации).

Заметим, что использование медицинской (вирусологической) терминологии связано с тем, что по характеру жизнедеятельности и проявлениям программы-вирусы сходны с биологическими вирусами. В этой связи естественным является применение той же терминологии и для способов и средств борьбы с компьютерными вирусами: «антивирус», «карантин», «вакцина», «фаг», о чем более подробно будет сказано далее.

Для реализации каждого из этапов цикла жизни вируса в его структуру включают несколько взаимосвязанных элементов (частей):

- ответственная за внедрение и инкубационный период;
- осуществляющая его копирование и добавление к другим файлам (программам);
- часть, в которой реализуется проверка условия активизации его деятельности;
- содержащая алгоритм деструктивных действий;
- реализующая алгоритм саморазрушения.

Часто названные части вируса хранятся отдельно друг от друга, что затрудняет борьбу с ними.

Объекты воздействия компьютерных вирусов можно условно разделить на две группы:

- с целью продления своего существования вирусы поражают другие программы, причем не все, а те, которые наиболее часто используются и/или имеют высокий приоритет в ВС (заметим, что сами программы, в которых находятся вирусы, с точки зрения реализуемых ими функций, как правило, не портятся);
- с деструктивными целями вирусы воздействуют чаще всего на данные, реже — на программы.

Назовем наиболее широко распространившиеся деструктивные функции вирусов:

- изменение данных в соответствующих файлах;
- изменение назначенного магнитного диска, когда запись информации осуществляется на неизвестный пользователю диск;
- уничтожение информации путем форматирования диска или отдельных треков (дорожек) на нем;
- уничтожение каталога файлов или отдельных файлов на диске;
- уничтожение (выключение) программ, постоянно находящихся в ОС;
- нарушение работоспособности ОС, что требует ее периодической перезагрузки;
- уничтожение специальных файлов ОС и др. Классифицировать компьютерные вирусы можно по различным признакам. Укажем четыре из них [53]:
- среда обитания;
- ОС;
- особенности алгоритма работы;
- деструктивные возможности.

По среде обитания вирусы можно разделить на файловые; загрузочные; макровирусы; сетевые.

Файловые вирусы либо различными способами внедряются в выполняемые файлы (наиболее распространенный тип вирусов), либо создают файлы-двойники (компаньон-вирусы), либо используют особенности организации файловой системы (link-вирусы).

Загрузочные вирусы записывают себя либо в загрузочный сектор диска (boot-сектор), либо в сектор, содержащий системный загрузчик винчестера (Master Boot Record), либо меняют указатель на активный boot-сектор.

Макровирусы заражают файлы-документы, электронные таблицы и презентации ряда популярных редакторов.

Сетевые вирусы используют для своего распространения протоколы или команды компьютерных сетей и электронной почты.

Существует большое количество сочетаний — например, файлово-загрузочные вирусы, заражающие как файлы, так и загрузочные сектора дисков. Такие вирусы, как правило, имеют довольно сложный алгоритм работы, часто применяют оригинальные методы проникновения в операционную систему, используют стелс- и полиморфик-технологии. Другой пример такого сочетания — сетевой макровирус, который не только заражает редактируемые документы, но и рассылает свои копии по электронной почте (например, «I love you» или «Анна Курникова»).

Заражаемая ОС (вернее, ОС, объекты которой подвержены заражению) является вторым уровнем деления вирусов на классы. Каждый файловый или сетевой вирус заражает файлы какой-либо одной или нескольких ОС — DOS, Win95/98/NT, OS/2 и т.д. Макровирусы заражают файлы программ Word, Excel, Power Point, Office. Загрузочные вирусы также ориентированы на конкретные форматы расположения системных данных в загрузочных секторах дисков.

Среди особенностей алгоритма работы вирусов выделяются следующие:

- резидентность;
- использование стелс-алгоритмов;
- самошифрование и полиморфичностъ;
- использование нестандартных приемов.

Резидентный вирус при инфицировании компьютера оставляет в оперативной памяти свою

резидентную часть, которая затем перехватывает обращения ОС к объектам заражения и внедряется в них. Резидентные вирусы находятся в памяти и являются активными вплоть до выключения компьютера или перезагрузки ОС. *Нерезидентные вирусы* не заражают память компьютера и сохраняют активность ограниченное время. Некоторые вирусы оставляют в оперативной памяти небольшие резидентные программы или функции, которые не распространяют вирус. Такие вирусы считаются нерезидентными.

Резидентными можно считать макровирусы, поскольку они постоянно присутствуют в памяти компьютера на все время работы зараженного редактора. При этом роль ОС берет на себя редактор, а понятие «перезагрузка операционной системы» трактуется как выход из редактора.

В многозадачных ОС время «жизни» резидентного DOS-вируса также может быть ограничено моментом закрытия зараженного DOS-окна, а активность загрузочных вирусов в некоторых операционных системах ограничивается моментом инсталляции дисковых драйверов ОС.

Использование *стелс-алгоритмов* позволяет вирусам полностью или частично скрыть себя в системе. Наиболее распространенным стелс-алгоритмом является перехват запросов ОС на чтение/запись зараженных объектов. Стелс-вирусы при этом либо временно лечат их, либо «подставляют» вместо себя незараженные участки информации. В случае макровирусов наиболее популярный способ — запрет вызовов меню просмотра макросов. Один из первых файловых стелс-вирусов — вирус Frodo, первый загрузочный стелс-вирус — Brain.

Самошифрование и *полиморфичностъ* используются практически всеми типами вирусов для того, чтобы максимально усложнить процедуру детектирования вируса. *Полиморфик-вирусы* — это достаточно труднообнаружимые вирусы, не имеющие сигнатур, т.е. не содержащие ни одного постоянного участка кода. В большинстве случаев два образца одного и того же полиморфик-вируса не будут иметь ни одного совпадения. Это достигается шифрованием основного тела вируса и модификациями программы-расшифровщика.

Различные *нестандартные приемы* часто используются в вирусах для того, чтобы как можно глубже спрятать себя в ядре ОС (как это делает вирус «ЗАРАЗА»), защитить от обнаружения свою резидентную копию (вирусы TPVO, Trout2), затруднить лечение от вируса (например, поместив свою копию в Flash-BIOS) и т.д.

По деструктивным возможностям вирусы можно разделить:

- на безвредные, т. е. никак не влияющие на работу компьютера (кроме уменьшения свободной памяти на диске в результате своего распространения);
- неопасные, влияние которых ограничивается уменьшением свободной памяти на диске и графическими, звуковыми и прочими подобными эффектами;
- опасные вирусы, которые могут привести к серьезным сбоям в работе компьютера;
- очень опасные, в алгоритм работы которых заведомо заложены процедуры, способные привести к потере программ, уничтожению данных, стиранию необходимой для работы компьютера информации, записанной в системных областях памяти, и даже, как гласит одна из непроверенных компьютерных легенд, на быстрый износ движущихся частей механизмов — вводить в резонанс и разрушать головки некоторых типов винчестеров.

На сегодняшний день сложились установившиеся названия для некоторых типов вирусов. Так, *«ловушками»* называют вирусы, использующие имеющиеся неточности в действующих программах или их несовершенство (например, изменяющие адрес входа из программы в подпрограммы и обратно). *«Логические бомбы»,* или «бомбы замедленного действия», осуществляют длительную и разнообразную подготовку к проведению деструктивных действий и затем срабатывают при выполнении определенного комплекса условий (например, выполнении определенного этапа работ, наступлении заданного времени, обращении к программе определенного пользователя и т.п.). Эти вирусы особенно опасны в силу длительности периода времени, при котором они себя практически не обнаруживают, хотя уже ведут разрушительную работу. Факт проявления этих вирусов сопряжен с такой степенью порчи данных, что в установленной версии ОС оказываются неработоспособными практически все (или большинство) программы. Таким образом, машина становится полностью неработоспособной. *Вирусы-«черви»* вызывают неуправляемое функционирование, например, сетевых или периферийных устройств (бесконечный «прогон» бумаги в принтере, постоянную перезагрузку ОС и т. п.). «*Троянскими конями»* называют вирусы, распространяемые вместе с ПО специального назначения, причем для пользователя оказываются крайне неожиданными их деструктивные действия (например, таким вирусом могут быть заражены сами антивирусные программы). Внешне «троянцы» могут даже

выполнять некие полезные функции (фиктивно оптимизировать распределение памяти на компьютере, проводить уплотнение информации на диске и т.д.), но на самом деле либо разрушают систему (например, форматируют ваш винчестер на низком уровне или операцией многократного и оперативного считывания-записи информации способствуют механическому выведению из строя дисковода вашего компьютера), либо отдают контроль в руки другого человека. Пород «троянцев» множество: некоторые из них вообще не выполняют полезных функций, а просто скрытно «живут» на диске ЭВМ и совершают различные деструктивные действия, а некоторые, наоборот, совершенно не скрываются от пользователя, при этом производя некоторые манипуляции, о которых никто не подозревает (или не должен подозревать). Пример вируса первого типа — известный вирус Back Orifice, дающий «врагу» почти полный контроль над вашим компьютером в компьютерной сети и для вас невидимый. Пример вируса второго типа — подделки под браузер MS Internet Explorer, который при соединении с сайтом фирмы *Microsoft* развивает небывалую активность по пересылке данных с компьютера на сервер *Microsoft,* объем которых явно превосходит простой запрос загружаемого HTML документа (Web-страницы Интернета).

Для того чтобы применить тот или иной способ борьбы с конкретным вирусом, нужно сначала осуществить его диагностику (хотя следует признать, что все современные методы диагностики вирусов являются несовершенными). Названной цели служат программы трех типов:

- для выявления наличия вируса;
- обнаружения и удаления вируса;
- защиты от вирусов (препятствующие проникновению новых вирусов).

Борьба с вирусами ведется путем применения программ-антивирусов. *Антивирус* — программа, осуществляющая обнаружение или обнаружение и удаление вируса. Самыми популярными и эффективными антивирусными программами являются антивирусные сканеры (другие названия: фаги, полифаги) [25]. Следом за ними по эффективности и популярности следуют CRC-сканеры (также ревизоры, checksumer, integrity checker). Часто оба приведенных метода объединяются в одну универсальную антивирусную программу, что значительно повышает ее мощность. Применяются также различного типа блокировщики и иммунизаторы.

Проблема защиты от макровирусов. Поскольку проблема макровирусов в последнее время перекрывает все остальные проблемы, связанные с вирусами, на ней следует остановиться подробнее.

Существует несколько приемов и специальных функций, встроенных в Word/Excel и другие программы функций, направленных на предотвращение запуска вируса. Наиболее действенной из них является защита от вирусов, встроенная в Word и Excel (начиная с версий 7.0). Эта защита при открытии файла, содержащего любой макрос, сообщает о его присутствии и предлагает запретить этот макрос. В результате макрос не только не выполняется, но и не виден средствами Word/Excel.

Такая защита является достаточно надежной, однако абсолютно бесполезна, если пользователь работает с макросами (любыми): она не отличает «чистые» и зараженные макросы и выводит предупреждающее сообщение при открытии практически любого файла. По этой причине защита в большинстве случаев оказывается отключенной, что дает возможность вирусу проникнуть в систему. К тому же включение защиты от вирусов в уже зараженной системе не во всех случаях помогает — некоторые вирусы, однажды получив управление, при каждом запуске отключают защиту от вирусов и таким образом полностью блокируют ее.

Существуют другие методы противодействия вирусам, например функция DisableAutoMacros, однако она не запрещает выполнение прочих макросов и блокирует только те вирусы, которые для своего распространения используют один из автомакросов.

Запуск редактора Word с опцией /M (или с нажатой клавишей Shift) не является локацией от всех бед, а только отключает один макрос AutoExec и, таким образом, также не может служить надежной защитой от вируса.

Сетевые вирусы. Среди множества известных вирусов особое место занимают сетевые вирусы, как правило, они считаются и наиболее опасными [24].

Возможность инфицирования компьютерными вирусами корпоративных компьютерных сетей становится все более серьезной проблемой. Опасность действия вирусов определяется возможностью частичной или полной потери ценной информации, а также потерей времени и средств, направленных на восстановление нормального функционирования ИС.

Обычная корпоративная компьютерная сеть включает в себя сотни рабочих станций, десятки серверов и, как правило, имеет очень сложную структуру. Стоимость обслуживания такой сети растет

вместе с ростом числа подключенных рабочих станций. При этом расходы на антивирусную защиту являются не последним пунктом в списке общих расходов. Основной возможностью их снижения является централизация управления работой антивирусной системы, установленной в корпоративной компьютерной сети. Администратор должен иметь возможность с одного рабочего места вести мониторинг всех точек проникновения вирусов и управлять всеми антивирусными продуктами, перекрывающими эти точки. Компьютерные вирусы проникают в сеть вместе с файлами. Основные пути, которыми файлы, зараженные вирусами, попадают в корпоративную сеть:

- копирование инфицированных файлов или при запуске программ и других файлов с переносимых источников (гибких дисков, оптических дисков, Zip, Jaz, Floptical и т.д.);
- программное обеспечение, полученное через Web или FTP и сохраненное на локальных рабочих станциях;
- файлы, получаемые удаленными пользователями, которые соединяются с корпоративной сетью по модему;
- файлы, получаемые при соединении удаленного сервера с сетью для обмена с файловым сервером, сервером приложений или сервером баз данных;
- электронная почта, содержащая приложенные зараженные файлы.

Глобальная компьютерная сеть Интернет на сегодняшний день также является основным источником вирусов. Наибольшее число заражений вирусом происходит при обмене письмами. Пользователь зараженного макровирусом редактора, сам того не подозревая, рассылает зараженные письма адресатам, которые, в свою очередь, отправляют новые зараженные письма и т.д. К сетевым относятся вирусы, которые для своего распространения активно используют протоколы и возможности локальных и глобальных сетей. Основным принципом работы сетевого вируса является возможность самостоятельно передать свой код на удаленный сервер или рабочую станцию. «Полноценные» сетевые вирусы при этом обладают еще и возможностью запустить на выполнение свой код на удаленном компьютере или, по крайней мере, «подтолкнуть» пользователя к запуску зараженного файла.

Бытует ошибочное мнение, что сетевым является любой вирус, распространяющийся в компьютерной сети. Но в таком случае практически все вирусы были бы сетевыми, даже наиболее примитивные из них: ведь самый обычный нерезидентный вирус при заражении файлов не разбирается — сетевой (удаленный) это диск или локальный. В результате такой вирус способен заражать файлы в пределах сети, но отнести его к сетевым вирусам нельзя.

Наибольшую известность приобрели сетевые вирусы конца 80-х гг. XX в., их также называют *сетевыми червями* (worms). К ним относятся вирус Морриса, вирусы Cristmas Tree и Wank Worm&. Для своего распространения они использовали ошибки и недокументированные функции глобальных сетей того времени — вирусы передавали свои копии с сервера на сервер и запускали их на выполнение. В случае с вирусом Морриса эпидемия захватила даже несколько глобальных сетей в США.

После некоторого затишья проблема сетевых вирусов вновь обострилась в начале 1997 г. с появлением вирусов Macro.Word. ShareFun и Win.Homer. Первый из них использует возможности электронной почты Microsoft Mail — он создает новое письмо, содержащее зараженный файл-документ (ShareFun является макровирусом), затем выбирает из списка адресов службы MSMail (из вашей компьютерной записной книги) три случайных адреса и рассылает по ним зараженное письмо. Поскольку многие пользователи устанавливают параметры MSMail таким образом, что при получении письма автоматически запускается MS Word, то вирус «автоматически» внедряется в компьютер адресата зараженного письма.

Этот вирус иллюстрирует первый тип современного сетевого вируса, который объединяет возможности встроенного в Word/Excel языка Basic, протоколы и особенности электронной почты и функции автозапуска, необходимые для распространения вируса.

Второй вирус (Homer) использует для своего распространения протокол FTP (File Transfer Protocol) и передает свою копию на удаленный FTP-сервер в каталог Incoming. Поскольку сетевой протокол FTP исключает возможность запуска файла на удаленном сервере, этот вирус можно охарактеризовать как «полусетевой», однако это реальный пример возможностей вирусов по использованию современных сетевых протоколов и поражению глобальных сетей.

Приведем некоторые правила защиты от компьютерных вирусов [19].

Правило первое. Крайне осторожно относитесь к программам и документам, изготовленным с помощью пакета Microsoft Office (с помощью приложений Word/Excel/Power Point/Access), которые вы получаете из глобальных сетей. Перед тем как запустить файл на выполнение или открыть

документ/таблицу/презентацию/ базу данных, обязательно проверьте их на наличие вирусов. Используйте специализированные антивирусы для проверки всех файлов, приходящих по электронной почте (из локальных и глобальных сетей в целом). К сожалению, на сегодняшний день пока нельзя однозначно назвать хотя бы один антивирус, который достаточно надежно ловил бы вирусы в приходящих из Интернета файлах, но не исключено, что такие антивирусы появятся в ближайшем будущем.

Правило второе. Касается защиты локальных сетей. Для уменьшения риска заразить файл на сервере администраторам сетей следует активно использовать стандартные возможности защиты сети: ограничение прав пользователей; установку атрибутов «только на чтение» или даже «только на запуск» для всех выполняемых файлов (к сожалению, это не всегда оказывается возможным); скрытие (закрытие) важных разделов диска и директорий и т.д.

Используйте специализированные антивирусы, проверяющие все файлы, к которым идет обращение. Если это по какой-либо причине невозможно, регулярно проверяйте сервер обычными антивирусными программами. Значительно уменьшается риск заражения компьютерной сети при использовании бездисковых рабочих станций. Желательно также перед тем как запустить новое ПО, проверить его на тестовом компьютере, не подключенном к общей сети.

Правило третье. Лучше приобретать дистрибутивные копии ПО у официальных продавцов, чем бесплатно или почти бесплатно копировать их из других источников или покупать нелицензионные «пиратские» копии. При этом значительно снижается вероятность заражения.

Как следствие из этого правила вытекает необходимость хранения дистрибутивных копий ПО (в том числе копий ОС), причем копии желательно хранить на защищенных от записи дискетах (дисках).

Пользуйтесь только хорошо зарекомендовавшими себя источниками программ и прочих файлов, хотя это не всегда спасает (например, на сервере *Microsoft* довольно долгое время находился документ, зараженный макровирусом Wazzu). По-видимому, единственными надежными с точки зрения защиты от вирусов являются BBS/ftp/WWW антивирусных фирм-разработчиков.

Правило четвертое. Старайтесь не запускать непроверенные файлы, в том числе полученные из компьютерной сети. Желательно использовать только программы, полученные из надежных источников. Перед запуском новых программ обязательно проверьте их одним или несколькими антивирусами. Если даже ни один антивирус не среагировал на файл, который был снят с BBS или электронной конференции, не торопитесь его запускать. Подождите неделю, если этот файл вдруг окажется заражен новым неизвестным вирусом, то, скорее всего, кто-либо потерпит неудобства раньше вас и своевременно сообщит об этом.

Желательно также, чтобы при работе с новым ПО в памяти резидентно находился какой-либо антивирусный монитор. Если запускаемая программа заражена вирусом, то такой монитор поможет обнаружить вирус и остановить его распространение.

Все рекомендации приводят к необходимости ограничения круга лиц, допущенных к работе на конкретном компьютере. Как правило, наиболее часто подвержены заражению «многопользовательские» персональные компьютеры.

Правило пятое. Пользуйтесь утилитами проверки целостности информации. Такие утилиты сохраняют в специальных БД информацию о системных областях дисков (или целиком системные области) и информацию о файлах (контрольные суммы, размеры, атрибуты, даты последней модификации файлов и т.д.). Периодически сравнивайте информацию, хранящуюся в подобной БД, с реальным содержимым винчестера, так как практически любое несоответствие может служить сигналом о появлении вируса или «троянской» программы.

Правило шестое. Периодически сохраняйте на внешнем носителе файлы, с которыми ведется работа. Такие резервные копии носят название *backup-копий.* Затраты на копирование файлов, содержащих исходные тексты программ, БД, документацию, значительно меньше затрат на восстановление этих файлов при проявлении вирусом агрессивных свойств или при сбое компьютера.

При наличии стримера или какого-либо другого внешнего носителя большого объема имеет смысл делать *backup*-копии всего содержимого винчестера. Но поскольку времени на создание подобной копии требуется значительно больше, чем на сохранение только рабочих файлов, такие копии делаются гораздо реже.

В качестве вывода отметим, что проблему защиты от вирусов целесообразно рассматривать в общем контексте проблемы защиты информации от несанкционированного доступа. Основной принцип, который должен быть положен в основу разработки технологии защиты от вирусов, состоит в создании

многоуровневой распределенной системы защиты, включающей регламентацию доступа к ЭВМ, проводимых операций на ЭВМ, применение специальных программных и аппаратных средств и т.п.

В заключение подчеркнем три важнейших обстоятельства: во-первых, защита информации от несанкционированных действий эффективна только тогда, когда комплексно (системно) применяются все известные способы и средства; во-вторых, защита должна осуществляться непрерывно; и в-третьих, на защиту информации не нужно жалеть затрат денежных, материальных, временных и других ресурсов, ибо они многократно окупятся сохранением целостности информации.

Глава 4. CASE-ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ АВТОМАТИЗИРОВАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

4.1. Общие положения CASE-технологий

За последние десятилетия сформировалось новое направление в программотехнике — CASE (Computer-Aided Software/System Engineering), в дословном переводе — разработка ПО ИС при поддержке (с помощью) компьютера. В настоящее время не существует общепринятого определения CASE, этот термин используется в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь ПО, в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных АИС в целом. Теперь под термином CASE-средства понимаются ПС, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и БД, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы [23, 24]. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки АИС.

CASE-средства позволяют не только создавать «правильные» продукты, но и обеспечить «правильный» процесс их создания. Основная цель CASE состоит в том, чтобы отделить проектирование ПО от его кодирования и последующих этапов разработки, а также скрыть от разработчиков все детали среды разработки и функционирования ПО. При использовании CASE-технологий изменяются все этапы жизненного цикла ПО (подробнее об этом будет сказано ниже) ИС, при этом наибольшие изменения касаются этапов анализа и проектирования. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры ПС. Такие методологии обеспечивают строгое и наглядное описание проектируемой системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру со все большим числом уровней.

CASE-технологии успешно применяются для построения практически всех типов систем ПО, однако устойчивое положение они занимают в следующих областях:

- обеспечение разработки делового и коммерческого ПО — широкое применение CASE-технологий обусловлено массовостью этой прикладной области, в которой CASE применяется не только для разработки ПО, но и для создания моделей систем, помогающих решать задачи стратегического планирования, управления финансами, определения политики фирм, обучения персонала и др. (это направление получило свое собственное название — бизнес-анализ);
- разработка системного и управляющего ПО — активное применение CASE-технологий связано с большой сложностью данной проблематики и со стремлением повысить эффективность работ.

CASE не революция в программотехнике, а результат естественного эволюционного развития всей отрасли средств, называемых ранее инструментальными, или технологическими. С самого начала CASE-технологии развивались с целью преодоления ограничений при использовании структурных методологий проектирования 60 —70-х гг. прошлого века (сложности понимания, большой трудоемкости и стоимости использования, трудности внесения изменений в проектные спецификации и т.д.) за счет их автоматизации и интеграции поддерживающих средств. Таким образом, CASE-технологии не могут считаться самостоятельными методологиями, они только развивают структурные методологии и делают более эффективным их применение за счет автоматизации.

Помимо автоматизации структурных методологий и, как следствие, возможности применения современных методов системной и программной инженерии, CASE-средства обладают следующими

основными достоинствами:

- улучшают качество создаваемого ПО за счет средств автоматического контроля (прежде всего контроля проекта);
- позволяют за короткое время создавать прототип будущей системы, что дает возможность на ранних этапах оценить ожидаемый результат;
- ускоряют процесс проектирования и разработки;
- освобождают разработчика от рутинной работы, позволяя ему целиком сосредоточиться на творческой части разработки;
- поддерживают развитие и сопровождение разработки;
- поддерживают технологии повторного использования компонентов разработки.

Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т. д. В 70—80-х гг. прошлого века стала на практике применяться структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания АИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей АИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяли разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако широкое применение этой методологии и следование ее рекомендациям при разработке конкретных АИС встречалось достаточно редко, поскольку при неавтоматизированной (ручной) разработке это практически невозможно. Это и способствовало появлению программно-технологических средств особого класса — CASE-средств, реализующих CASE-технологию создания и сопровождения АИС.

Необходимо понимать, что успешное применение CASE-средств невозможно без понимания базовой технологии, на которой эти средства основаны. Сами по себе программные CASE-средства являются средствами автоматизации процессов проектирования и сопровождения ИС. Без понимания методологии проектирования ИС невозможно применение CASE-средств.

4.2. Жизненный цикл программного обеспечения информационной системы

Одним из базовых понятий методологии проектирования АИС является понятие жизненного цикла ее программного обеспечения (ЖЦПО). ЖЦПО — это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации [6].

Структура ЖЦПО базируется на трех группах процессов:

- основные процессы ЖЦПО (приобретение, поставка, разработка, эксплуатация, сопровождение);
- вспомогательные процессы, обеспечивающие выполнение основных процессов (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем);
- организационные процессы (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого ЖЦ, обучение).

Разработка включает в себя все работы по созданию ПО и его компонентов в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала и т.д. Разработка ПО включает в себя, как правило, анализ, проектирование и реализацию (программирование).

Эксплуатация включает в себя работы по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование БД и рабочих мест пользователей, обеспечение эксплуатационной документацией, проведение обучения персонала и т.д., и непосредственно эксплуатацию, в том числе локализацию проблем и устранение причин их возникновения, модификацию ПО в рамках установленного регламента, подготовку предложений по совершенствованию, развитию и модернизации системы.

Управление проектом связано с вопросами планирования и организации работ, создания коллективов разработчиков и контроля за сроками и качеством выполняемых работ. Техническое и организационное обеспечение проекта включает выбор методов и инструментальных средств для реализации проекта, определение методов описания промежуточных состояний разработки, разработку методов и средств испытаний ПО, обучение персонала и т.п. Обеспечение качества проекта связано с проблемами верификации, проверки и тестирования ПО. Верификация — это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа. Проверка позволяет оценить соответствие параметров разработки с исходными требованиями. Проверка частично совпадает с тестированием, которое связано с идентификацией различий между действительными и ожидаемыми результатами и оценкой соответствия характеристик ПО исходным требованиям. В процессе реализации проекта важное место занимают вопросы идентификации, описания и контроля конфигурации отдельных компонентов и всей системы в целом.

Управление конфигурацией является одним из вспомогательных процессов, поддерживающих основные процессы ЖЦ ПО, прежде всего процессы разработки и сопровождения ПО. При создании проектов сложных ИС, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, возникает проблема учета их связей и функций, создания унифицированной структуры и обеспечения развития всей системы. Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях ЖЦ. Общие принципы и рекомендации конфигурационного учета, планирования и управления конфигурациями ПО отражены в проекте стандарта *ISO* 12207 — 2.

Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами. Результатами анализа, в частности, являются функциональные модели, информационные модели и соответствующие им диаграммы. ЖЦПО носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах.

Существующие модели ЖЦ определяют порядок исполнения этапов в ходе разработки, а также критерии перехода от этапа к этапу. В соответствии с этим наибольшее распространение получили три следующие модели ЖЦ:

- *каскадная модель* (70—80 гг. XX в.) — предполагает переход на следующий этап после полного окончания работ по предыдущему этапу;
- *поэтапная модель с промежуточным контролем* (80 — 85 гг. XX в.) — итерационная модель разработки ПО с циклами обратной связи между этапами. Преимущество такой модели заключается в том, что межэтапные корректировки обеспечивают меньшую трудоемкость по сравнению с каскадной моделью, однако время жизни каждого из этапов растягивается на весь период разработки;
- *спиральная модель* (86 — 90 гг. XX в.) — делает упор на начальные этапы ЖЦ: анализ требований, проектирование спецификаций, предварительное и детальное проектирование. На этих этапах проверяется и обосновывается реализуемость технических решений путем создания прототипов. Каждый виток спирали соответствует поэтапной модели создания фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

Специалистами отмечаются следующие преимущества спиральной модели:

- накопление и повторное использование программных средств, моделей и прототипов;
- ориентация на развитие и модификацию ПО в процессе его проектирования;
- анализ риска и издержек в процессе проектирования.

Главная особенность индустрии создания ПО состоит в концентрации сложности на начальных этапах ЖЦ (анализ, проектирование) при относительно невысокой сложности и трудоемкости последующих этапов. Более того, нерешенные вопросы и ошибки, допущенные на этапах анализа и проектирования, порождают на последующих этапах трудные, часто неразрешимые проблемы и в конечном счете приводят к неуспеху всего проекта.

4.3. RAD-технологии быстрого создания приложений

Одним из возможных подходов к разработке ПО в рамках спиральной модели ЖЦ является получившая в последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development). Под этим термином обычно понимается процесс разработки ПО, содержащий три элемента:

- небольшую команду программистов (от 2 до 10 чел.);
- короткий, но тщательно проработанный производственный график (от 2 до 6 мес);
- повторяющийся цикл, при котором разработчики по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.

Команда разработчиков должна представлять собой группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ПО с использованием CASE-средств. Члены коллектива должны также уметь трансформировать в рабочие прототипы предложения конечных пользователей.

Жизненный цикл ПО по методологии RAD состоит из четырех фаз:

анализа и планирования требований;

проектирования;

построения;

внедрения.

На *фазе анализа и планирования требований* пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности. Определение требований выполняется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта, определяются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации данного проекта в установленных рамках финансирования, на данных аппаратных средствах и т.п. Результатом данной фазы должны быть список и приоритетность функций будущей АИС, предварительные функциональные и информационные модели ИС.

На *фазе проектирования* часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков. CASE-средства используются для быстрого получения работающих прототипов приложений. Пользователи, непосредственно взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. Анализируется и корректируется функциональная модель. Каждый процесс рассматривается детально. При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности. Определяются требования разграничения доступа к данным. На этой же фазе происходит определение набора необходимой документации.

После детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении АИС на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время — порядка 60 — 90 дней. С использованием CASE-средств проект распределяется между различными командами (делится функциональная модель). Результатом данной фазы должны быть:

- общая информационная модель системы;
- функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
- точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;
- построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что в традиционном подходе при передаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой опасности.

В отличие от традиционного подхода, при котором использовались специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасывались после того, как выполняли задачу устранения неясностей в проекте, в подходе RAD каждый прототип развивается в часть будущей системы. Таким образом, на следующую фазу

передается более полная и полезная информация.

На *фазе построения* непосредственно выполняется быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется при помощи автоматических генераторов, получающих информацию непосредственно из репозитория CASE-средств. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется непосредственно в процессе разработки.

После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом. В завершение физического проектирования системы:

- определяется необходимость распределения данных;
- производится анализ использования данных;
- производится физическое проектирование БД;
- определяются требования к аппаратным ресурсам;
- определяются способы увеличения производительности;
- завершается разработка документации проекта.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

На *фазе внедрения* производится обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы.

Приведенная схема разработки АИС не является абсолютной. Возможны различные варианты, зависящие, например, от начальных условий, в которых ведется разработка: разрабатывается ли совершенно новая система; было ли проведено информационное обследование организации и существует ли модель ее деятельности; существует ли в организации некоторая АИС, которая может быть использована в качестве начального прототипа или должна быть интегрирована с разрабатываемой, и т.п.

Следует, однако, отметить, что методология RAD, как и любая другая, не может претендовать на универсальность, она хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика. Если же разрабатывается типовая система, которая не является законченным продуктом, а представляет собой комплекс типовых компонентов, централизованно сопровождаемых, адаптируемых к программно-техническим платформам, СУБД, средствам телекоммуникации, организационно-экономическим особенностям объектов внедрения и интегрируемых с существующими разработками, на первый план выступают такие показатели проекта, как управляемость и качество, которые могут войти в противоречие с простотой и скоростью разработки. Для таких проектов необходимы высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки.

Методология RAD неприменима для построения сложных расчетных программ, ОС или программ управления космическими кораблями, т. е. программ, требующих написания большого объема (сотни тысяч строк) уникального кода.

Не подходят для разработки по методологии RAD приложения, в которых отсутствует ярко выраженная интерфейсная часть, наглядно определяющая логику работы системы (например, приложения реального времени), и приложения, от которых зависит безопасность людей (например, управление самолетом или атомной электростанцией), так как итеративный подход предполагает, что первые несколько версий наверняка не будут полностью работоспособны, что в данном случае исключается.

Основные принципы методологии RAD:

- разработка приложений итерациями;
- необязательность полного завершения работ на каждом из этапов ЖЦ;
- обязательное вовлечение пользователей в процесс разработки АИС;

- необходимое применение CASE-средств, обеспечивающих целостность проекта;
- применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
- необходимое использование генераторов кода;
- использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
- тестирование и развитие проекта, осуществляемые одновременно с разработкой;
- ведение разработки немногочисленной, хорошо управляемой командой профессионалов;
- грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

4.4. Структурный метод разработки программного обеспечения

Сущность структурного подхода к разработке АИС заключается в ее *декомпозиции* (разбивке) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые, в свою очередь, делятся на подфункции, подразделяемые на задачи, и т.д. Процесс разбивки продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны. При разработке системы «снизу вверх» от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов.

Все методологии структурного анализа базируются на ряде общих принципов, часть из которых регламентирует организацию работ на начальных этапах ЖЦ, а часть используется при выработке рекомендаций по организации работ [6]. В качестве двух базовых принципов используются следующие: принцип «разделяй и властвуй» и принцип иерархического упорядочивания. Первый является принципом решения трудных проблем путем разбиения их на множество меньших независимых задач, более легких для понимания и решения. Второй принцип декларирует, что устройство этих частей также существенно для понимания. Уровень уяснения проблемы резко повышается при представлении ее частей в виде древовидных иерархических структур, т.е. система может быть понята и построена по уровням, каждый из которых добавляет новые детали.

Выделение двух базовых принципов инженерии ПО не означает, что остальные принципы являются второстепенными, игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к неуспеху всего проекта). Отметим основные из таких принципов.

1. Принцип абстрагирования заключается в выделении существенных с некоторых позиций аспектов системы и отвлечении от несущественных с целью представления проблемы в простом общем виде.

2. Принцип формализации заключается в необходимости строгого методического подхода к решению проблемы.

3. Принцип «упрятывания» заключается в упрятывании несущественной на конкретном этапе информации: каждая часть «знает» только необходимую ей информацию.

4. Принцип концептуальной общности заключается в следовании единой философии на всех этапах ЖЦ (структурный анализ — структурное проектирование — структурное программирование — структурное тестирование).

5. Принцип полноты заключается в контроле присутствия лишних элементов.

6. Принцип непротиворечивости заключается в обоснованности и согласованности элементов.

7. Принцип логической независимости заключается в концентрации внимания на логическом проектировании для обеспечения независимости от физического проектирования.

8. Принцип независимости данных заключается в том, что модели данных должны быть проанализированы и спроектированы независимо от процессов их логической обработки, а также от их физической структуры и распределения.

9. Принцип структурирования данных заключается в том, что данные должны быть структурированы и иерархически организованы.

10. Принцип доступа конечного пользователя заключается в том, что пользователь должен иметь средства доступа к БД, которые он может использовать непосредственно (без программирования).

Соблюдение указанных принципов необходимо при организации работ на начальных этапах ЖЦ независимо от типа разрабатываемого ПО и используемых при этом методологий. Руководствуясь всеми принципами в комплексе, можно на более ранних стадиях разработки понять, что будет представлять собой создаваемая система, обнаружить промахи и недоработки, что, в свою очередь,

облегчит работы на последующих этапах ЖЦ и понизит стоимость разработки.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой, и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются следующие:

- SADT (Structured Analysis and Design Technique) — модели и соответствующие функциональные диаграммы;
- DFD (Data Flow Diagrams) — диаграммы потоков данных;
- ERD (Entity-Relationship Diagrams) — диаграммы «сущность— связь»;
- STD (State Trasition Diagrams) — диаграммы переходов состояний.

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру ПО: архитектуру ПО, структурные схемы программ и диаграммы экранных форм.

Перечисленные модели в совокупности дают полное описание АИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

Методология SADT. Методология SADT разработана Дугласом Россом, на ее основе разработана, в частности, известная методология IDEFO (Icam DEFinition), которая является основной частью программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой по инициативе США. Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями. Основные элементы этой методологии основываются на следующих концепциях:

- графическое представление блочного моделирования. Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описывается посредством интерфейсных дуг, выражающих «ограничения», которые, в свою очередь, определяют, когда и каким образом функции выполняются и управляются;
- строгость и точность. Выполнение правил SADT требует достаточной строгости и точности, не накладывая в то же время чрезмерных ограничений на действия аналитика.

Правила SADT включают:

- ограничение количества блоков на каждом уровне декомпозиции (как правило, 3 — 6 блоков);
- связность диаграмм (номера блоков);
- уникальность меток и наименований (отсутствие повторяющихся имен);
- синтаксические правила для графики (блоков и дуг);
- разделение входов и управлений (правило определения роли данных).
- отделение организации от функции, т. е. исключение влияния организационной структуры на функциональную модель.

Методология SADT может использоваться для моделирования широкого круга систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям и реализует эти функции. Для уже существующих систем SADT может быть использована для анализа функций, выполняемых системой, а также для указания механизмов, посредством которых они осуществляются.

Результатом применения методологии SADT является модель, которая состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга. Диаграммы — главные компоненты модели, все функции ИС и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке, показана с левой стороны блока, а результаты выхода показаны с правой стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рис. 4.1).

Рис. 4.1. Функциональный блок и интерфейсные дуги

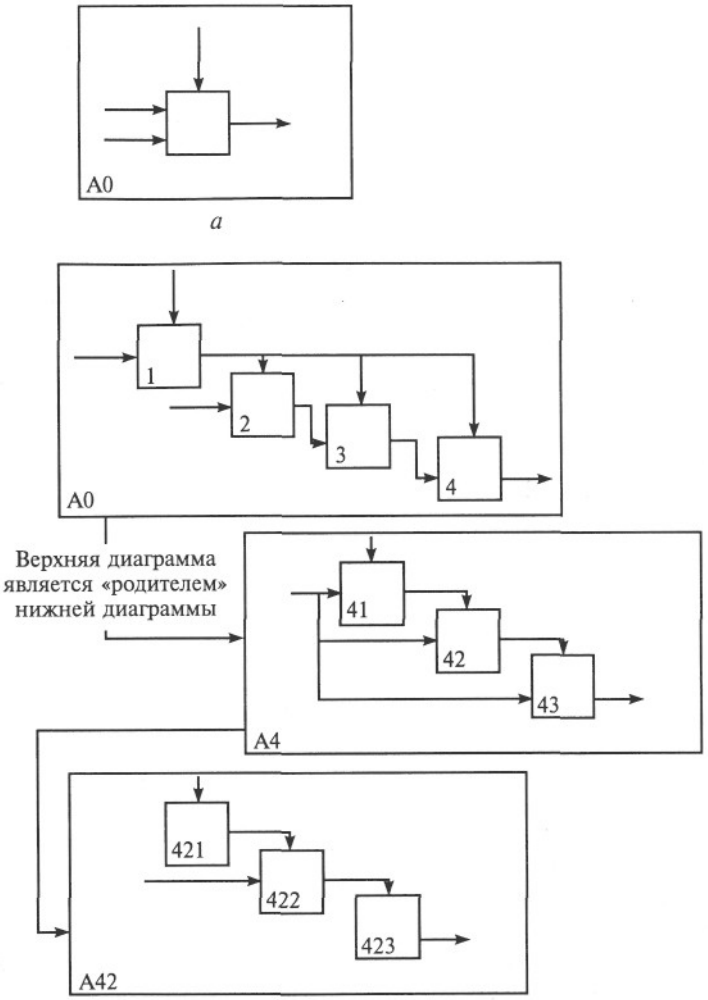
Одной из наиболее важных особенностей методологии SADT является постепенное введение все больших уровней детализации по мере создания диаграмм, отображающих модель.

Построение SADT-модели начинается с представления всей системы в виде простейшего компонента — одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг — они также представляют полный набор внешних интерфейсов системы в целом.

Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления.

Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, т.е., как уже отмечалось, так называемый родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить, и из него не может быть ничего удалено.

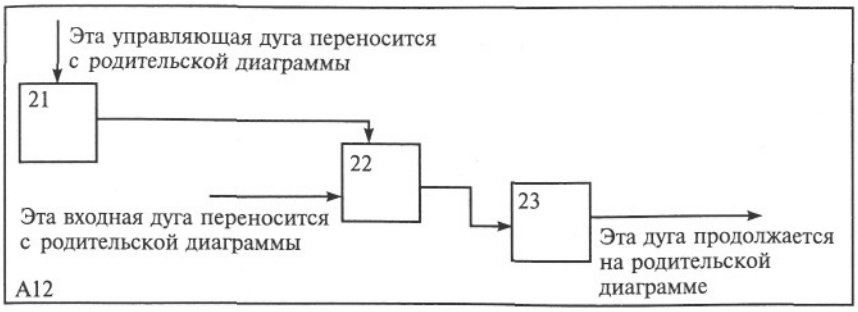
Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах. На каждом шаге декомпозиции более общая диаграмма называется родительской для более детальной диаграммы (рис. 4.2 и 4.3).



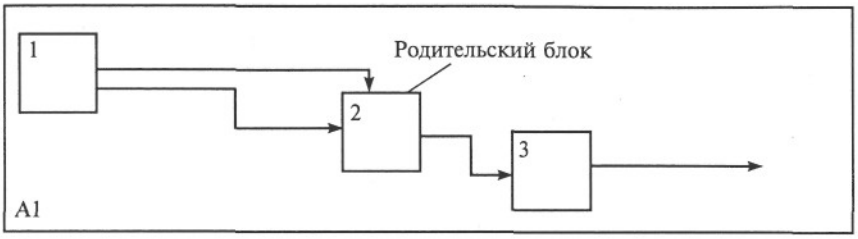
а

б

Рис. 4.2. Структура SADT-модели (декомпозиция диаграмм):
а — общее представление; *б* — детальное представление



а



б

Рис. 4.3. Иерархия диаграмм:
а — детальная диаграмма; *б* — родительская диаграмма

Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается неприсоединенным. Неприсоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме. Неприсоединенные концы должны соответствовать дугам на исходной диаграмме. Все граничные дуги должны продолжаться на родительской диаграмме, чтобы она была полной и непротиворечивой.

На SADT-диаграммах не указаны явно ни последовательность, ни время. Обратные связи, итерации, продолжающиеся процессы и перекрывающиеся (по времени) функции могут быть изображены с помощью дуг. Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т.д.

Как было отмечено, механизмы (дуги с нижней стороны) показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию.

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, которая, в свою очередь, может быть далее детализирована с помощью необходимого числа диаграмм. Таким образом формируется иерархия диаграмм.

Для того чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Например, A21 является диаграммой, которая детализирует блок 1 на диаграмме A2. Аналогично A2 детализирует блок 2 на диаграмме A0, которая является самой верхней диаграммой модели.

Одним из важных моментов при проектировании ИС с помощью методологии SADT является точная согласованность типов связей между функциями. Различают по крайней мере семь типов связи по относительной значимости.

Ниже каждый тип связи кратко определен и проиллюстрирован с помощью типичного примера из SADT (табл. 4.1).

Таблица 4.1

Типы связности для функций и данных

Значимость	Тип связности	Для функций	Для данных
0	Случайная	Случайная	Случайная
1	Логическая	Функции одного и того же множества или типа (например, «редактировать все входы»)	Данные одного и того же множества или типа
2	Временная	Функции одного и того же периода времени (например, «операции инициализации»)	Данные, используемые в каком-либо временном интервале
3	Процедурная	Функции, работающие в одной и той же фазе или итерации (например, «первый проход компилятора»)	Данные, используемые во время одной и той же фазы или итерации
4	Коммуникационная	Функции, использующие одни и те же данные	Данные, на которые воздействует одна и та же деятельность
5	Последовательная	Функции, выполняющие последовательные преобразования одних и тех же данных	Данные, преобразуемые последовательными функциями
6	Функциональная	Функции, объединяемые для выполнения одной функции	Данные, связанные с одной функцией

(0) *Тип случайной связности.* Наименее желательный. Случайная связность возникает, когда конкретная связь между функциями мала или полностью отсутствует. Это относится к ситуации, когда имена данных на SADT-дугах в одной диаграмме имеют малую связь друг с другом.

(1) *Тип логической связности.* Логическое связывание происходит тогда, когда данные и функции собираются вместе вследствие того, что они попадают в общий класс или набор элементов, но необходимых функциональных отношений между ними не обнаруживается.

(2) *Тип временной связности.* Связанные по времени элементы возникают вследствие того, что они представляют функции, связанные во времени, когда данные используются одновременно или функции включаются параллельно, а не последовательно.

(3) *Тип процедурной связности.* Процедурно-связанные элементы появляются сгруппированными вместе вследствие того, что они выполняются в течение одной и той же части цикла или процесса.

(4) *Тип коммуникационной связности.* Диаграммы демонстрируют коммуникационные связи, когда блоки группируются вследствие того, что они используют одни и те же входные данные и/или производят одни и те же выходные данные.

(5) *Тип последовательной связности.* На диаграммах, имеющих последовательные связи, выход одной функции служит входными данными для следующей функции. Связь между элементами на диаграмме является более тесной, чем на рассмотренных выше уровнях связок, поскольку моделируются причинно-следственные зависимости.

(6) *Тип функциональной связности.* Диаграмма отражает полную функциональную связность при наличии полной зависимости одной функции от другой. Диаграмма, которая является чисто функциональной, не содержит чужеродных элементов, относящихся к последовательному или более слабому типу связности. Одним из способов определения функционально-связанных диаграмм является рассмотрение двух блоков, связанных через управляющие дуги.

Моделирование потоков данных (процессов). Основным средством моделирования функциональных требований АИС являются диаграммы потоков данных (DFD — Data Flow Diagrams). С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств — продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Для изображения DFD традиционно используются две различные *нотации:* Йодана *(Yourdon)* и Гейна — Сарсона *(Gane—Sarson)* (рис. 4.4).

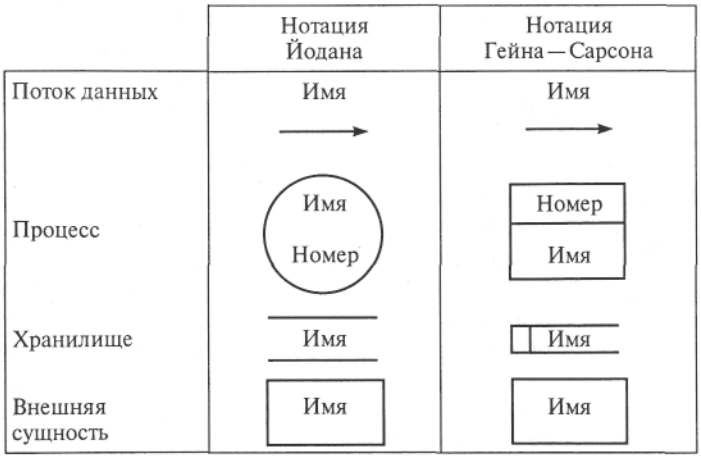


Рис. 4.4. Представление нотаций Йодана и Гейна — Сарсона

В соответствии с методологией модель системы определяется как иерархия диаграмм потоков данных (ДПД или DFD), описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы ИС с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процессы становятся элементарными и детализировать их далее невозможно.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те, в свою очередь, преобразуют

информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям — потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются:

- внешние сущности;
- системы/подсистемы (см. ранее);
- процессы;
- накопители данных (хранилище);
- потоки данных.

Внешняя сущность — это материальный предмет или физическое лицо, являющееся источником или приемником информации, например заказчики, персонал, поставщики, клиенты, склад. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой АИС.

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т.д. В различных нотациях процесс может изображаться на диаграммах по-разному. Номер процесса служит для его идентификации. В поле имени вводится наименование процесса в виде предложения с активным недвусмысленным глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже, например:

- «Ввести сведения о клиентах»;
- «Выдать информацию о текущих расходах»;
- «Проверить кредитоспособность клиента».

Использование таких глаголов, как «обработать», «модернизировать» или «отредактировать», означает, как правило, недостаточно глубокое понимание данного процесса и требует дальнейшего анализа.

В последнее время принято использовать еще и поле физической реализации, информация в котором показывает, какое подразделение организации, программа или аппаратное устройство выполняет данный процесс.

Хранилище (накопитель данных) представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

Накопитель данных может быть реализован физически в виде микрофиши, ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т.д. Накопитель данных идентифицируется буквой *D* и произвольным числом. Имя накопителя выбирается из соображения наибольшей информативности для проектировщика.

В общем случае накопитель данных является прообразом будущей БД, и описание хранящихся в нем данных должно быть увязано с информационной моделью. Поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой, и т.д.

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление. Каждый поток данных имеет имя, отражающее его содержание.

Первым шагом при построении иерархии DFD является построение контекстных диаграмм. Обычно при проектировании относительно простых АИС строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс, соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы. Для сложных АИС строится иерархия контекстных диаграмм. При этом контекстная диаграмма верхнего уровня содержит не единственный главный процесс, а набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем.

Иерархия контекстных диаграмм определяет взаимодействие основных функциональных подсистем проектируемой АИС как между собой, так и с внешними входными и выходными потоками данных и внешними объектами (источниками и приемниками информации), с которыми взаимодействует АИС.

Разработка контекстных диаграмм решает проблему строгого определения функциональной структуры АИС на самой ранней стадии ее проектирования, что особенно важно для сложных многофункциональных систем, в разработке которых участвуют разные организации и коллективы разработчиков.

После построения контекстных диаграмм полученную модель следует проверить на полноту исходных данных об объектах системы и изолированность объектов (отсутствие информационных связей с другими объектами). Для каждой подсистемы, присутствующей на контекстных диаграммах, выполняется ее детализация при помощи DFD. Каждый процесс на DFD, в свою очередь, может быть детализирован при помощи DFD или мини-спецификации. При детализации должны выполняться следующие правила:

- правило балансировки — при детализации подсистемы или процесса детализирующая диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеет информационную связь детализируемая подсистема или процесс на родительской диаграмме;
- правило нумерации — при детализации процессов должна поддерживаться их иерархическая нумерация. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т.д.

Мини-спецификация (описание логики процесса) должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу.

Мини-спецификация является конечной вершиной иерархии DFD. Решение о завершении детализации процесса и использовании мини-спецификации принимается аналитиком, исходя из следующих критериев:

- наличие у процесса относительно небольшого количества входных и выходных потоков данных (2 — 3 потока);
- возможность описания преобразования данных процессом в виде последовательного алгоритма;
- выполнение процессом единственной логической функции преобразования входной информации в выходную;
- возможность описания логики процесса при помощи мини-спецификации небольшого объема (не более 20 — 30 строк).

При построении иерархии DFD переходить к детализации процессов следует только после определения содержания всех потоков и накопителей данных, которое описывается при помощи структур данных. Структуры данных конструируются из элементов данных и могут содержать альтернативы, условные вхождения и итерации. Условное вхождение означает, что данный компонент может отсутствовать в структуре. Альтернатива означает, что в структуру может входить один из перечисленных элементов. Итерация означает вхождение любого числа элементов в указанном диапазоне. Для каждого элемента данных может указываться его тип (непрерывные или дискретные данные). Для непрерывных данных может указываться единица измерения (кг, см и т.п.), диапазон значений, точность представления и форма физического кодирования. Для дискретных данных может указываться таблица допустимых значений.

После построения законченной модели системы ее необходимо верифицировать (проверить на полноту и согласованность). В полной модели все ее объекты (подсистемы, процессы, потоки данных) должны быть подробно описаны и детализированы. Выявленные недетализированные объекты следует детализировать, вернувшись на предыдущие шаги разработки. В согласованной модели для всех потоков данных и накопителей данных должно выполняться правило сохранения информации: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть записаны.

Моделирование данных. Цель моделирования данных состоит в обеспечении разработчика АИС концептуальной схемой БД в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему БД.

Наиболее распространенным средством моделирования данных являются диаграммы «сущность—связь» (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных БД (см. гл. 6).

Нотация ERD была впервые введена П.Ченом *(P. Chen)* и получила дальнейшее развитие в работах

С.Баркера *(S. Barker)*.

Методология IDEF1. Метод IDEF1, разработанный Т. Рэмеем *(T.Ramey),* также основан на подходе П.Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия — методология IDEF1X. IDEF1X разработана с учетом таких требований, как простота изучения и возможность автоматизации. IDEFlX-диаграммы используются рядом распространенных CASE-средств (в частности, ERwin, Design/IDEF).

4.5. Методологии проектирования программного обеспечения

Современные методологии и реализующие их технологии поставляются в электронном виде вместе с CASE-средствами и включают библиотеки процессов, шаблонов, методов, моделей и других компонентов, предназначенных для построения ПО того класса систем, на который ориентирована методология. Электронные методологии включают также средства, которые должны обеспечивать их адаптацию для конкретных пользователей и развитие методологии по результатам выполнения конкретных проектов.

Процесс адаптации заключается в удалении ненужных процессов, действий ЖЦ и других компонентов методологии, в изменении неподходящих или в добавлении собственных процессов и действий, а также методов, моделей, стандартов и руководств. Настройка методологии может осуществляться также по следующим аспектам: этапы и операции ЖЦ, участники проекта, используемые модели ЖЦ, поддерживаемые концепции и др.

Электронные методологии и технологии (и поддерживающие их CASE-средства) составляют ядро комплекса согласованных инструментальных средств среды разработки АИС.

Методология DATARUN. Одной из наиболее распространенных в мире электронных методологий является методология DATARUN. В соответствии с методологией DATARUN ЖЦПО разбивается на стадии, которые связываются с результатами выполнения основных процессов, определяемых стандартом *ISO* 12207. Каждую стадию, помимо ее результатов, должен завершать план работ на следующую стадию.

Стадия формирования требований и планирования включает в себя действия по определению начальных оценок объема и стоимости проекта. Должны быть сформулированы требования и экономическое обоснование для разработки АИС, функциональные модели (модели бизнес-процессов организации) и исходная концептуальная модель данных, которые дают основу для оценки технической реализуемости проекта. Основными результатами этой стадии должны быть модели деятельности организации (исходные модели процессов и данных организации), требования к системе, включая требования по сопряжению с существующими АИС, исходный бизнес-план.

Стадия концептуального проектирования начинается с детального анализа первичных данных и уточнения концептуальной модели данных, после чего проектируется архитектура системы. Архитектура включает в себя разделение концептуальной модели на обозримые подмодели. Оценивается возможность использования существующих АИС и выбирается соответствующий метод их преобразования. После построения проекта уточняется исходный бизнес-план. Выходными компонентами этой стадии являются концептуальная модель данных, модель архитектуры системы и уточненный бизнес-план.

На *стадии спецификации приложений* продолжается процесс создания и детализации проекта. Концептуальная модель данных преобразуется в реляционную модель данных. Определяется структура приложения, необходимые интерфейсы приложения в виде экранов, отчетов и пакетных процессов вместе с логикой их вызова. Модель данных уточняется бизнес-правилами и методами для каждой таблицы. В конце этой стадии принимается окончательное решение о способе реализации приложений. По результатам стадии должен быть построен проект АИС, включающий модели архитектуры АИС, данных, функций, интерфейсов (с внешними системами и с пользователями), требований к разрабатываемым приложениям (модели данных, интерфейсов и функций), к доработкам существующих АИС, к интеграции приложений, а также сформирован окончательный план создания АИС.

На *стадии разработки, интеграции и тестирования* должна быть создана тестовая БД, частные и комплексные тесты. Проводится разработка, прототипирование и тестирование БД и приложений в соответствии с проектом. Отлаживаются интерфейсы с существующими системами. Описывается

конфигурация текущей версии ПО. На основе результатов тестирования проводится оптимизация БД и приложений. Приложения интегрируются в систему, проводится тестирование приложений в составе системы и испытания системы. Основными результатами стадии являются готовые приложения, проверенные в составе системы на комплексных тестах, текущее описание конфигурации ПО, скорректированная по результатам испытаний версия системы и эксплуатационная документация на систему.

Стадия внедрения включает в себя действия по установке и внедрению БД и приложений. Основными результатами стадии должны быть готовая к эксплуатации и перенесенная на программно-аппаратную платформу заказчика версия системы, документация сопровождения и акт приемочных испытаний по результатам опытной эксплуатации.

Стадии сопровождения и развития включают процессы и операции, связанные с регистрацией, диагностикой и локализацией ошибок, внесением изменений и тестированием, проведением доработок, тиражированием и распространением новых версий ПО в места его эксплуатации, переносом приложений на новую платформу и масштабированием системы. Стадия развития фактически является повторной итерацией стадии разработки.

Методология DATARUN опирается на две модели, или два представления: модель организации и модель АИС.

Методология DATARUN базируется на системном подходе к описанию деятельности организации. Построение моделей начинается с описания процессов, из которых затем извлекаются первичные данные (стабильное подмножество данных, которые организация должна использовать для своей деятельности). Первичные данные описывают продукты или услуги организации, выполняемые операции (транзакции) и потребляемые ресурсы. К первичным относятся данные, которые описывают внешние и внутренние сущности, такие, как служащие, клиенты или агентства, а также данные, полученные в результате принятия решений, как, например, графики работ, цены на продукты.

Основной принцип DATARUN заключается в том, что первичные данные, если они должным образом организованы в модель данных, становятся основой для проектирования архитектуры АИС. Архитектура АИС будет более стабильной, если она основана на первичных данных, тесно связанных с основными деловыми операциями, определяющими природу бизнеса, а не на традиционной функциональной модели.

Любая АИС представляет собой набор модулей, исполняемых процессорами и взаимодействующих с базами данных. БД и процессоры могут располагаться централизованно или быть распределенными. События в системе могут инициироваться внешними сущностями. Все транзакции осуществляются через объекты или модули интерфейса, которые взаимодействуют с одной или более БД.

Подход DATARUN преследует две цели:

- определить стабильную структуру, на основе которой будет строиться АИС. Такой структурой является модель данных, полученная из первичных данных, представляющих фундаментальные процессы организации;
- спроектировать АИС на основании модели данных.

Объекты, формируемые на основании модели данных, являются объектами БД, обычно размещаемыми на серверах в среде «клиент—сервер». Объекты интерфейса, определенные в архитектуре компьютерной системы, обычно размещаются на клиентской части. Модель данных, являющаяся основой для спецификации совместно используемых объектов БД и различных объектов интерфейса, обеспечивает сопровождаемость АИС.

В процессе разработки АИС создается ряд моделей:

- BPM (Business Process Model) — модуль построения модели процессов (бизнес-процессов);
- PDS (Primary Data Structure) — структура первичных данных;
- CDM (Conceptual Data Model) — концептуальная модель данных;
- SPM (System Process Model) — модель процессов системы;
- ISA (Information System Architecture) — архитектура информационной системы;
- ADM (Application Data Model) — модель данных приложения;
- IPM (Interface Presentation Model) — модель представления интерфейса;
- ISM (Interface Specification Model) — модель спецификации интерфейса.

CASE-средство Silverrun обеспечивает автоматизацию проведения проектных работ в соответствии с методологией DATARUN и обеспечивает создание этих моделей. Предоставляемая этими средствами среда проектирования дает возможность руководителю проекта контролировать проведение и

отслеживать выполнение работ, вовремя замечать отклонения от графика. Каждый участник проекта, подключившись к этой среде, может выяснить содержание и сроки выполнения порученной ему работы, детально изучить технику ее выполнения в гипертексте по технологиям и вызвать инструмент (модуль Silverrun) для реального выполнения работы.

Информационная система создается последовательным построением ряда моделей, начиная с модели бизнес-процессов и заканчивая моделью программы, автоматизирующей эти процессы.

Создаваемая АИС должна основываться на функциях, выполняемых организацией. Поэтому первая создаваемая модель — это модель бизнес-процессов, построение которой осуществляется в модуле Silverrun BPM. Для этой модели используется специальная нотация BPM. В процессе анализа и спецификации бизнес-функций выявляются основные информационные объекты, которые документируются как структуры данных, связанные с потоками и хранилищами модели. Источниками для создания структур являются используемые в организации документы, должностные инструкции, описания производственных операций. Эти данные вводятся в том виде, как они существуют в деятельности организации. Нормализация и удаление избыточности производится позже, при построении концептуальной модели данных в модуле Silverrun ERX. После создания модели бизнес-процессов информация сохраняется в репозитории проекта.

В процессе обследования работы организации выявляются и документируются структуры первичных данных. Эти структуры заносятся в репозитории модуля BPM при описании циркулирующих в организации документов, сообщений, данных. В модели бизнес-процессов первичные структуры данных связаны с потоками и хранилищами информации.

На основе структур первичных данных в модуле Silverrun ERX создается концептуальная модель данных (ER-модель). От структур первичных данных концептуальная модель отличается удалением избыточности, стандартизацией наименований понятий и нормализацией. Эти операции в модуле ERX выполняются при помощи встроенной экспертной системы. Цель концептуальной модели данных — описать используемую информацию без деталей возможной реализации в БД, но в хорошо структурированном нормализованном виде.

На основе модели бизнес-процессов и концептуальной модели данных проектируется архитектура АИС. Определяются входящие в систему приложения, для каждого приложения специфицируются используемые данные и реализуемые функции. Архитектура АИС создается в модуле Silverrun BPM с использованием специальной нотации ISA. Основное содержание этой модели — структурные компоненты системы и навигация между ними. Концептуальная модель данных разбивается на части, соответствующие входящим в состав системы приложениям.

Перед разработкой приложений должна быть спроектирована структура корпоративной базы данных. DATARUN предполагает использование БД, основанной на реляционной модели. Концептуальная модель данных после нормализации переносится в модуль реляционного моделирования Silverrun RDM (см. далее) с помощью специального моста ERX—RDM. Преобразование модели из формата ERX в формат RDM происходит автоматически без вмешательства пользователя. После преобразования форматов получается модель реляционной базы данных. Эта модель детализируется в модуле Silverrun RDM определением физической реализации (типов данных СУБД, ключей, индексов, триггеров, ограничений целостности). Правила обработки данных можно задавать как непосредственно на языке программирования СУБД, так и в декларативной форме, не привязанной к реализации. Мосты Silverrun к реляционным СУБД переводят эти декларативные правила на язык требуемой системы, что снижает трудоемкость программирования процедур сервера базы данных, а также позволяет из одной спецификации генерировать приложения для разных СУБД.

С помощью модели системных процессов детально документируется поведение каждого приложения. В модуле BPM создается модель системных процессов, определяющая, каким образом реализуются бизнес-процессы. Эта модель создается отдельно для каждого приложения и тесно связана с моделью данных приложения.

Приложение состоит из интерфейсных объектов (экранных форм, отчетов, процедур обработки данных). Каждый интерфейс системы (экранная форма, отчет, процедура обработки данных) имеет дело с подмножеством базы данных. В модели данных приложения (созданной в модуле RDM) создается подсхема базы данных для каждого интерфейса этого приложения. Уточняются также правила обработки данных, специфичные для каждого интерфейса.

Модель представления интерфейса — это описание внешнего вида интерфейса с точки зрения конечного пользователя системы. Это может быть документ, показывающий внешний вид экрана или

структуру отчета, или экран (отчет), созданный с помощью одного из средств визуальной разработки приложений — так называемых *языков четвертого поколения* (4GL — Fourth Generation Languages). Так как большинство языков 4GL позволяют быстро создавать работающие прототипы приложений, пользователь имеет возможность увидеть работающий прототип системы на ранних стадиях проектирования.

После создания подсхем реляционной модели для приложений проектируется детальная структура каждого приложения в виде схемы навигации экранов, отчетов, процедур пакетной обработки. На данном шаге эта структура детализируется до указания конкретных столбцов и таблиц базы данных, правил их обработки, вида экранных форм и отчетов. Полученная модель детально документирует приложение и непосредственно используется для программирования специфицированных интерфейсов.

Далее, с помощью средств разработки приложений происходит физическое создание системы: приложения программируются и интегрируются в ИС.

Характеристика современных CASE-средств. Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО [14].

Наиболее трудоемкими этапами разработки ИС являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

В разряд CASE-средств попадают как относительно дешевые системы для персональных компьютеров с весьма ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов ЖЦПО и обладающее следующими основными характерными особенностями:

- мощные графические средства для описания и документирования АИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности;
- интеграция отдельных компонентов CASE-средств, обеспечивающая управляемость процессом разработки АИС;
- использование специальным образом организованного хранилища проектных метаданных (репозитория).

Интегрированное CASE-средство (или комплекс средств, поддерживающих полный ЖЦПО) содержит следующие компоненты:

- репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;
- графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели ИС;
- средства разработки приложений, включая языки 4GL и генераторы кодов;
- средства конфигурационного управления;
- средства документирования;
- средства тестирования;
- средства управления проектом;
- средства реинжиниринга.

Все современные CASE-средства могут быть классифицированы в основном по типам и категориям. Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы ЖЦ. Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает отдельные локальные средства, решающие небольшие автономные задачи *(tools),*

набор частично интегрированных средств, охватывающих большинство этапов ЖЦ ИС *(toolkit),* и полностью интегрированные средства, поддерживающие весь ЖЦ ИС и связанные общим репозиторием. Помимо этого, CASE-средства можно классифицировать:

- по применяемым методологиям и моделям систем и БД;
- степени интегрированности с СУБД;
- доступным платформам.

Классификация по типам в основном совпадает с компонентным составом CASE-средств и включает следующие основные типы (после названия средства в скобках указана фирма-разработчик):

- средства анализа (Upper CASE), предназначенные для построения и анализа моделей предметной области (Design/IDEF *(Meta Software),* BPwin *(Logic Works));*
- средства анализа и проектирования (Middle CASE), поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций (Vantage Team Builder *(Cayenne),* Designer/2000 *(ORACLE),* Silverran *(CSA),* PRO-IV *(McDonnell Douglas),* CASE.Аналитик (МакроПроджект)). Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;
- средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin *(Logic Works),* S-Designer *(SDP)* и DataBase Designer *(ORACLE).* Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;
- средства разработки приложений. К ним относятся средства 4GL (Uniface *(Compuware),* JAM *(JYACC),* PowerBuilder *(Sybase),* Developer/2000 *(ORACLE),* New Era *(Informix),* SQL Windows *(Gupta),* Delphi *(Borland)* и др.) и генераторы кодов, входящие в состав Vantage Team Builder, PRO-IV и частично в Silverrun;
- средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций. Средства анализа схем БД и формирования ERD входят в состав Vantage Team Builder, PRO-IV, Silverrun, Designer/ 2000, ERwin и S-Designer. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке C++ (Rational Rose *(Rational Software),* Object Team *(Cayenne)).*

Вспомогательные типы включают:

- средства планирования и управления проектом (SE Companion, Microsoft Project и др.);
- средства конфигурационного управления (PVCS *(Intersolv));*
- средства тестирования (Quality Works *(Segue Software));*
- средства документирования (SoDA *(Rational Software Corporation)).*

На сегодняшний день российский рынок ПО располагает следующими наиболее развитыми CASE-средствами: Silverrun; Designer/2000; Vantage Team Builder (*Westmount I-CASE);* S-Designer; ERwin + BPwin; CASE.Аналитик.

Кроме того, на рынке постоянно появляются как новые для отечественных пользователей системы (например, CASE/4/0, PRO-IV, System Architect, Visible Analyst Workbench, EasyCASE), так и новые версии и модификации перечисленных систем.

Охарактеризуем основные возможности CASE-средств на примере имеющей широкое распространение системы Silverrun.

CASE-средство Silverrun американской фирмы *Computer Systems Advisers, Inc. (CSA)* используется для анализа и проектирования АИС бизнес-класса и ориентировано в большей степени на спиральную модель ЖЦ. Оно применимо для поддержки любой методологии, основанной на раздельном построении функциональной и информационной моделей (диаграмм потоков данных и диаграмм «сущность—связь»).

Настройка на конкретную методологию обеспечивается выбором требуемой графической нотации моделей и набора правил проверки проектных спецификаций. В системе имеются готовые настройки для наиболее распространенных методологий: DATARUN (основная методология, поддерживаемая Silverrun), Gane/Sarson, Yourdon/DeMarco, Merise, Ward/Mellor, Information Engineering. Для каждого понятия, введенного в проекте, имеется возможность добавления собственных описаний. Архитектура Silverrun позволяет наращивать среду разработки по мере необходимости.

Silverrun имеет модульную структуру и состоит из четырех модулей, каждый из которых является самостоятельным продуктом и может приобретаться и использоваться без связи с остальными модулями.

Модуль построения моделей бизнес-процессов в форме диаграмм потоков данных (BPM — Business Process Modeler) позволяет моделировать функционирование обследуемой организации или создаваемой АИС. В модуле BPM обеспечена возможность работы с моделями большой сложности: автоматическая перенумерация, работа с деревом процессов (включая визуальное перетаскивание ветвей), отсоединение и присоединение частей модели для коллективной разработки. Диаграммы могут изображаться в нескольких предопределенных нотациях, включая Yourdon/DeMarco и Gane/Sarson. Имеется также возможность создавать собственные нотации, в том числе добавлять в число изображаемых на схеме дескрипторов определенные пользователем поля.

Модуль концептуального моделирования данных (ERX — Entity-Relationship Expert) обеспечивает построение моделей данных «сущность—связь», не привязанных к конкретной реализации. Этот модуль имеет встроенную экспертную систему (ЭС), позволяющую создать корректную нормализованную модель данных посредством ответов на содержательные вопросы о взаимосвязи данных. Возможно автоматическое построение модели данных из описаний структур данных. Анализ функциональных зависимостей атрибутов дает возможность проверить соответствие модели требованиям третьей нормальной формы и обеспечить их выполнение. Проверенная модель передается в модуль RDM.

Модуль реляционного моделирования (RDM — Relational Data Modeler) позволяет создавать детализированные модели «сущность—связь», предназначенные для реализации в реляционной базе данных. В этом модуле документируются все конструкции, связанные с построением БД: индексы, триггеры, хранимые процедуры и т.д. Гибкая изменяемая нотация и расширяемость репозитория позволяют работать по любой методологии. Возможность создавать подсхемы соответствует подходу ANSI SPARC к представлению схемы БД. На языке подсхем моделируются как узлы распределенной обработки, так и пользовательские представления. Этот модуль обеспечивает проектирование и полное документирование реляционных БД.

Менеджер репозитория рабочей группы (WRM — Workgroup Repository Manager) применяется как словарь данных для хранения общей для всех моделей информации, а также обеспечивает интеграцию модулей Silverrun в единую среду проектирования.

Платой за высокую гибкость и разнообразие изобразительных средств построения моделей является такой недостаток Silverrun, как отсутствие жесткого взаимного контроля между компонентами различных моделей (например, возможности автоматического распространения изменений между DFD различных уровней декомпозиции). Следует, однако, отметить, что этот недостаток может иметь существенное значение только в случае использования каскадной модели ЖЦПО.

Для автоматической генерации схем баз данных у Silverrun существуют мосты к наиболее распространенным СУБД: Oracle, Informix, DB2, Ingres, Progress, SQL Server, SQLBase, Sybase. Для передачи данных в средства разработки приложений имеются мосты к языкам 4GL: JAM, PowerBuilder, SQL Windows, Uniface, NewEra, Delphi. Все мосты позволяют загрузить в Silverrun RDM информацию из каталогов соответствующих СУБД или языков 4GL. Это позволяет документировать, перепроектировать или переносить на новые платформы уже находящиеся в эксплуатации базы данных и прикладные системы. При использовании моста Silverrun расширяет свой внутренний репозиторий специфичными для целевой системы атрибутами. После определения значений этих атрибутов генератор приложений переносит их во внутренний каталог среды разработки или использует при генерации кода на языке SQL. Таким образом, можно полностью определить ядро БД с использованием всех возможностей конкретной СУБД: триггеров, хранимых процедур, ограничений ссылочной целостности. При создании приложения на языке 4GL данные, перенесенные из репозитория Silverrun, используются либо для автоматической генерации интерфейсных объектов, либо для быстрого их создания вручную.

Для обмена данными с другими средствами автоматизации проектирования, создания специализированных процедур анализа и проверки проектных спецификаций, составления специализированных отчетов в соответствии с различными стандартами в системе Silverrun имеется три способа выдачи проектной информации во внешние файлы:

- система отчетов. Можно, определив содержимое отчета по репозиторию, выдать отчет в текстовый файл. Этот файл можно затем загрузить в текстовый редактор или включить в другой

отчет;

- система экспорта/импорта. Для более полного контроля над структурой файлов в системе экспорта/импорта имеется возможность определять не только содержимое экспортного файла, но и разделители записей, полей в записях, маркеры начала и конца текстовых полей. Файлы с указанной структурой можно не только формировать, но и загружать в репозиторий. Это дает возможность обмениваться данными с различными системами: другими CASE-средствами, СУБД, текстовыми редакторами и электронными таблицами;
- хранение репозитория во внешних файлах через ODBC-драйверы. Для доступа к данным репозитория из наиболее распространенных систем управления БД обеспечена возможность хранить всю проектную информацию непосредственно в формате этих СУБД.

Групповая работа поддерживается в системе Silverrun двумя способами:

- в стандартной однопользовательской версии имеется механизм контролируемого разделения и слияния моделей. Разделив модель на части, можно раздать их нескольким разработчикам. После детальной доработки модели объединяются в единые спецификации;
- сетевая версия Silverrun позволяет осуществлять одновременную групповую работу с моделями, хранящимися в сетевом репозитории на базе СУБД Oracle, Sybase или Informix. При этом несколько разработчиков могут работать с одной и той же моделью, так как блокировка объектов происходит на уровне отдельных элементов модели.

Имеются реализации Silverrun трех платформ — MS Windows, Macintosh и OS/2 Presentation Manager — с возможностью обмена проектными данными между ними.

Помимо системы Silverrun, укажем назначение и других популярных CASE-средств и их групп.

Vantage Team Builder представляет собой интегрированный программный продукт, ориентированный на реализацию каскадной модели ЖЦПО и поддержку полного ЖЦПО.

Uniface 6.1 — продукт фирмы *Compuware* (США) — представляет собой среду разработки крупномасштабных приложений в архитектуре «клиент—сервер».

CASE-средство Designer/2000 2.0 фирмы Oracle является интегрированным CASE-средством, обеспечивающим в совокупности со средствами разработки приложений Developer/2000 поддержку полного ЖЦПО для систем, использующих СУБД Oracle.

Пакет CASE/4/0 (microTOOL GmbH), включающий структурные средства системного анализа, проектирования и программирования, обеспечивает поддержку всего ЖЦ разработки (вплоть до сопровождения) на основе сетевого репозитория, контролирующего целостность проекта и поддерживающего согласованную работу всех его участников (системных аналитиков, проектировщиков, программистов).

Локальные средства. Пакет ERWin (Logic Works) используется при моделировании и создании баз данных произвольной сложности на основе диаграмм «сущность—связь». В настоящее время ERWin является наиболее популярным пакетом моделирования данных благодаря поддержке широкого спектра СУБД самых различных классов — SQL-серверов (Oracle, Informix, Sybase SQL Server, MS SQL Server, Progress, DB2, SQLBase, Ingress, Rdb и др.) и «настольных» СУБД типа xBase (Clipper, dBASE, FoxPro, MS Access, Paradox и др.).

BPWin — средство функционального моделирования, реализующее методологию IDEFO. Модель в BPwin представляет собой совокупность SADT-диаграмм, каждая из которых описывает отдельный процесс, разбивая его на шаги и подпроцессы.

S-Designer 4.2 (Sybase/Powersoft) представляет собой CASE-средство для проектирования реляционных баз данных. По своим функциональным возможностям и стоимости он близок к CASE-средству ERWin, отличаясь внешне используемой на диаграммах нотацией. S-Designor реализует стандартную методологию моделирования данных и генерирует описание БД для таких СУБД, как Oracle, Informix, Ingres, Sybase, DB/2, Microsoft SQL Server и др.

CASE. Аналитик 1.1 (Эйтекс) является практически единственным в настоящее время конкурентоспособным отечественным CASE-средством функционального моделирования и реализует построение диаграмм потоков данных в соответствии с описанной ранее методологией.

Объектно-ориентированные CASE-средства. Rational Rose — CASE-средство фирмы *Rational Software Corporation* (США) — предназначено для автоматизации этапов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации. Rational Rose использует синтез-методологию объектно-ориентированного анализа и проектирования, основанную на подходах трех ведущих специалистов в данной области: М. Буча, К. Рамбо и Т. Джекобсона.

Разработанная ими универсальная нотация для моделирования объектов (язык UML — Unified Modeling Language) является в настоящее время и, очевидно, останется в будущем общепринятым стандартом в области объектно-ориентированного анализа и проектирования. Конкретный вариант Rational Rose определяется языком, на котором генерируются коды программ (C++, Smalltalk, PowerBuilder, Ada, SQLWindows и ObjectPro). Основной вариант — Rational Rose/C++ — позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций, а также генерировать программные коды на C++. Кроме того, Rational Rose содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонентов в новых проектах.

Средства конфигурационного управления. Цель конфигурационного управления — обеспечить управляемость и контролируемость процессов разработки и сопровождения ПО. Для этого необходима точная и достоверная информация о состоянии ПО и его компонентов в каждый момент времени, а также о всех предполагаемых и выполненных изменениях.

Для решения задач КУ применяются методы и средства, обеспечивающие идентификацию состояния компонентов, учет номенклатуры всех компонентов и модификаций системы в целом, контроль за вносимыми изменениями в компоненты, структуру системы и ее функции, а также координированное управление развитием функций и улучшением характеристик системы.

Наиболее распространенным средством КУ является PVCS фирмы *Intersolv* (США), включающее ряд самостоятельных продуктов: PVCS Version Manager, PVCS Tracker, PVCS Configuration Builder и PVCS Notify.

Средства документирования. Для создания документации в процессе разработки АИС используются разнообразные средства формирования отчетов, а также компоненты издательских систем. Обычно средства документирования встроены в конкретные CASE-средства. Исключением являются некоторые пакеты, предоставляющие дополнительный сервис при документировании. Из них наиболее активно используется SoDA (Software Document Automation).

Продукт SoDA предназначен для автоматизации разработки проектной документации на всех фазах ЖЦПО. Он позволяет автоматически извлекать разнообразную информацию, получаемую на разных стадиях разработки проекта, и включать ее в выходные документы. При этом контролируется соответствие документации проекту, взаимосвязь документов, обеспечивается их своевременное обновление. Результирующая документация автоматически формируется из множества источников, число которых не ограничено.

Пакет включает в себя графический редактор для подготовки шаблонов документов. Он позволяет задавать необходимый стиль, фон, шрифт, определять расположение заголовков, резервировать места, где будет размещаться извлекаемая из разнообразных источников информация. Изменения автоматически вносятся только в те части документации, на которые они повлияли в программе. Это сокращает время подготовки документации за счет отказа от перегенерации всей документации.

SoDA реализована на базе издательской системы FrameBuilder и предоставляет полный набор средств по редактированию и верстке выпускаемой документации.

Итоговым результатом работы системы SoDA является готовый документ (или книга). Документ может храниться в файле формата SoDA (FrameBuilder), который получается в результате генерации документа. Вывод на печать этого документа (или его части) возможен из системы SoDA.

Среда функционирования SoDA — ОС типа UNIX на рабочих станциях Sun SPARCstation, IBM RISC System/6000 или Hewlett Packard HP 9000 700/800.

Средства тестирования. Под *тестированием* понимается процесс исполнения программы с целью обнаружения ошибок. *Регрессионное тестирование* — это тестирование, проводимое после усовершенствования функций программы или внесения в нее изменений.

Одно из наиболее развитых средств тестирования QA (новое название — Quality Works) представляет собой интегрированную, многоплатформенную среду для разработки автоматизированных тестов любого уровня, включая тесты регрессии для приложений с графическим интерфейсом пользователя.

QA позволяет начинать тестирование на любой фазе ЖЦ, планировать и управлять процессом тестирования, отображать изменения в приложении и повторно использовать тесты для более чем 25 различных платформ.

В заключение приведем пример комплекса CASE-средств, обеспечивающего поддержку полного ЖЦПО. Нецелесообразно сравнивать отдельно взятые CASE-средства, поскольку ни одно из них не решает в целом все проблемы создания и сопровождения ПО. Это подтверждается также полным

набором критериев оценки и выбора, которые затрагивают все этапы ЖЦПО. Сравниваться могут комплексы методологически и технологически согласованных инструментальных средств, поддерживающие полный ЖЦПО и обеспеченные необходимой технической и методической поддержкой со стороны фирм-поставщиков (отметим, что рациональное комплексирование инструментальных средств разработки ПО ИС является важнейшим условием обеспечения качества этого ПО, причем это замечание справедливо для всех предметных областей). На сегодняшний день наиболее развитым из всех поставляемых в Россию комплексов такого рода является комплекс технологий и инструментальных средств создания ИС, основанный на методологии и технологии DATARUN. В состав комплекса входят следующие инструментальные средства:

- CASE-средство Silverrun;
- средство разработки приложений JAM;
- мост Silverran RDM ↔ JAM;
- комплекс средств тестирования QA;
- менеджер транзакций Tuxedo;
- комплекс средств планирования и управления проектом SE Companion;
- комплекс средств конфигурационного управления PVCS;
- объектно-ориентированное CASE-средство Rational Rose;
- средство документирования SoDA.

РАЗДЕЛ II. БАЗЫ ДАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Глава 5. ПРИНЦИПЫ ПОСТРОЕНИЯ И ЭТАПЫ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ

5.1. Основные понятия и определения

Автоматизированные информационно-справочные системы (АИСС) в настоящее время получили весьма широкое распространение, что связано прежде всего со сравнительной простотой их создания и исключительно высоким эффектом от внедрения. Методологической основой информационных технологий, реализуемых в АИСС, являются концепции централизованной (в рамках разработки баз и банков данных) и распределенной (в рамках создания информационных сетей) обработки информации.

В науке одним из наиболее сложных для строгого определения является понятие «информация». Согласно кибернетическому подходу, *«информация* — первоначально сообщение данных, сведений, осведомление и т.п. Кибернетика вывела понятие «информация» за пределы человеческой речи и других форм коммуникации между людьми, связала его с целенаправленными системами любой природы. Информация выступает в трех формах:

- биологической (биотоки; связи в генетических механизмах);
- машинной (сигналы в электрических цепях);
- социальной (движение знаний в общественных системах)» [15].

Иными словами, «информация — связь в любых целенаправленных системах, определяющая их целостность, устойчивость, уровень функционирования» [49]. Содержание и особенности информации раскрываются указанием действий, в которых она участвует:

- хранение (на некотором носителе информации);
- преобразование (в соответствии с некоторым алгоритмом);
- передача (с помощью передатчика и приемника по некоторой линии связи).

В соответствии с этим же подходом «данные — факты и идеи, представленные в формализованном виде, позволяющем передавать или обрабатывать их при помощи некоторого процесса и соответствующих технических устройств» [15].

В источниках [49, 53] понятия «информация» и «данные» определены несколько иначе.

«Информация: 1) совокупность знаний о фактических данных и связях между ними; 2) в вычислительной технике — содержание, присваиваемое данным посредством соглашений, распространяющихся на эти данные; данные, подлежащие вводу в ЭВМ, хранимые в ее памяти, обрабатываемые на ЭВМ и выдаваемые пользователям».

«Данные — информация, представленная в виде, пригодном для обработки автоматическими средствами при возможном участии человека» [53].

Как легко заметить, приведенные определения вынужденно используют такие сложно определяемые понятия, как «факты», «идеи» и особенно «знания».

В дальнейшем под *информацией* будем понимать любые сведения о процессах и явлениях, которые в той или иной форме передаются между объектами материального мира (людьми, животными, растениями, автоматами и др.).

Если рассмотреть некоторый объект материального мира, информация о котором представляет интерес, и наблюдателя (в роли которого и выступают АИС), способного фиксировать эту информацию в определенной, понятной другим форме, то говорят, что в памяти (сознании) наблюдателя находятся данные, описывающие состояние объекта. Таким образом, *данными* будем называть формализованную информацию, пригодную для последующей обработки, хранения и передачи средствами автоматизации профессиональной деятельности.

Информацию в ЭВМ можно хранить в виде различных данных (числовых, текстовых, визуальных и т.п.). Более того, для описания одной и той же информации можно предложить различные варианты их состава и структуры. Иными словами, правомерно говорить о моделировании в АИС информации о некотором множестве объектов материального мира совокупностью взаимосвязанных данных.

Информационное обеспечение *(information support)* АИС — совокупность единой системы классификации и кодирования информации; унифицированных систем документации и используемых массивов информации [53, 54]. В дальнейшем нас будет интересовать именно последний аспект данного определения.

В этой связи в качестве главных задач создания информационного обеспечения АИС можно выделить, во-первых, определение состава и структуры данных, достаточно «хорошо» описывающих требуемую информацию, во-вторых, обоснование способов хранения и переработки данных с использованием ЭВМ.

Процесс создания информационного обеспечения включает несколько этапов, рассмотрению которых посвящен подразд. 5.2. В данном подразделе остановимся на понятиях и определениях, связанных с технологией банков данных.

Прежде чем определить понятие «банк данных», необходимо остановиться на другом ключевом понятии — «предметная область».

Под *предметной областью* будем понимать информацию об объектах, процессах и явлениях окружающего мира, которая с точки зрения потенциальных пользователей должна храниться и обрабатываться в информационной системе. В этом определении особое внимание следует уделить важности роли потенциальных потребителей информационных ресурсов АИС. Именно этот аспект обусловливает и структуру, и основные задачи, и вообще целесообразность создания того или иного банка.

Банк данных — ИС, включающая в свой состав комплекс специальных методов и средств для поддержания динамической информационной модели предметной области с целью обеспечения информационных потребностей пользователей [15, 39]. Очевидно, что банк данных может рассматриваться как специальная обеспечивающая подсистема в составе старшей по иерархии АИС.

Поддержание динамической модели предметной области предусматривает не только хранение информации о ней и своевременное внесение изменений в соответствии с реальным состоянием объектов, но и обеспечение возможности учета изменений состава этих объектов (в том числе появление новых) и связей между ними (т.е. изменений самой структуры хранимой информации).

Обеспечение информационных потребностей (запросов) пользователей имеет два аспекта [45]:

- определение границ конкретной предметной области и разработка описания соответствующей информационной модели;
- разработка банка данных, ориентированного на эффективное обслуживание запросов различных категорий пользователей.

С точки зрения целевой направленности профессиональной деятельности принято выделять пять основных категорий пользователей [45]: аналитики, системные программисты, прикладные программисты, администраторы, конечные пользователи.

Различают пользователей постоянных и разовых; пользователей-людей и пользователей-задач; пользователей с различным уровнем компетентности (приоритетом) и др., причем каждый класс пользователей предъявляет собственные специфические требования к своему обслуживанию (прежде всего с точки зрения организации диалога «запрос —ответ»). Так, например, постоянные пользователи, как правило, обращаются в банк данных с фиксированными по форме (типовыми) запросами;

пользователи-задачи должны иметь возможность получать информацию из банка данных в согласованной форме в указанные области памяти; пользователи с низким приоритетом могут получать ограниченную часть информации и т.д. Наличие столь разнообразного состава потребителей информации потребовало включения в банк данных специального элемента — *словаря данных,* о чем будет сказано ниже.

Уровень сложности и важности задач информационного обеспечения АИС в рамках рассматриваемой технологии определяет ряд основных требований к банку данных [53]:

- адекватность информации состоянию предметной области;
- быстродействие и производительность;
- простота и удобство использования;
- массовость использования;
- защита информации;
- возможность расширения круга решаемых задач.

(Отметим, что все названные требования можно предъявить и к любому финансовому банку.)

По сравнению с традиционным обеспечением монопольными файлами каждого приложения централизованное управление данными в банке данных имеет ряд важных преимуществ:

- сокращение избыточности хранимых данных;
- устранение противоречивости хранимых данных;
- многоаспектное использование данных (при однократном вводе);
- комплексная оптимизация (с точки зрения удовлетворения разнообразных, в том числе и противоречивых, требований «в целом»);
- обеспечение возможности стандартизации;
- обеспечение возможности санкционированного доступа к данным и др.

Все названные преимущества, по существу, связаны с такими основополагающими принципами концепции банка данных, как интеграция данных, централизация управления ими и обеспечение независимости прикладных программ обработки данных и самих данных.

Структура типового банка данных, удовлетворяющего предъявляемым требованиям, представлена на рис. 5.1.

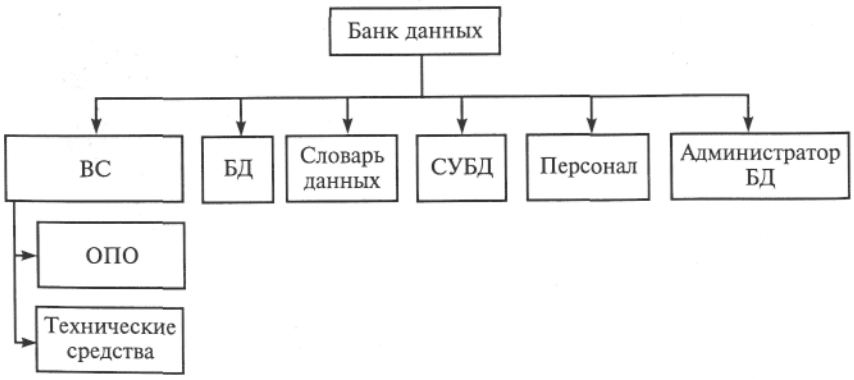


Рис. 5.1. Основные компоненты банка данных

Подробнее остановимся на составляющих банка данных, представляющих наибольший интерес.

База данных — совокупность специальным образом организованных (структурированных) данных и связей между ними. Иными словами, БД — это так называемое датологическое (от англ. *data* — данные) представление информации о предметной области. Если в состав банка данных входит одна БД, банк принято называть локальным; если БД несколько — интегрированным.

Словарь данных предназначен для хранения единообразной и централизованной информации обо всех ресурсах данных конкретного банка:

- об объектах, их свойствах и отношениях для данной ПО;
- данных, хранимых в БД (наименование, смысловое описание, структура, связи и т.п.);
- возможных значениях и форматах представления данных;
- источниках возникновения данных;
- кодах защиты и разграничении доступа пользователей к данным и т. п.

Система управления базами данных — специальный комплекс программ и языков, посредством которого организуется централизованное управление БД и обеспечивается доступ к ним.

В состав любой СУБД входят языки двух типов:

- язык описания данных (с его помощью описываются типы данных, их структура и связи);
- язык манипулирования данными (его часто называют «язык запросов к БД»), предназначенный для организации работы с данными в интересах всех типов пользователей.

Администратор БД — это лицо (группа лиц), реализующее управление БД. В этой связи сам банк данных можно рассматривать как автоматизированную систему управления БД. Функции администратора БД являются долгосрочными; он координирует все виды работ на этапах создания и применения банка данных. На стадии проектирования администратор БД выступает как идеолог и главный конструктор системы; на стадии эксплуатации он отвечает за нормальное функционирование банка данных, управляет режимом его работы и обеспечивает безопасность данных (последнее особенно важно при современном уровне развития средств коммуникации — см. гл. 3). Основные функции администратора БД [15, 54]:

- решать вопросы организации данных об объектах ПО и установления связей между этими данными с целью объединения информации о различных объектах; согласовывать представления пользователей;
- координировать все действия по проектированию, реализации и ведению БД; учитывать текущие и перспективные требования пользователей; следить, чтобы БД удовлетворяли актуальным потребностям;
- решать вопросы, связанные с расширением БД в связи с изменением границ ПО;
- разрабатывать и реализовывать меры по обеспечению защиты данных от некомпетентного их использования, от сбоев технических средств, по обеспечению секретности определенной части данных и разграничению доступа к ним;
- выполнять работы по ведению словаря данных; контролировать избыточность и противоречивость данных, их достоверность;
- следить за тем, чтобы банк данных отвечал заданным требованиям по производительности, т. е. чтобы обработка запросов выполнялась за приемлемое время;
- выполнять при необходимости изменения методов хранения данных, путей доступа к ним, связей между данными, их форматов; определять степень влияния изменений в данных на всю БД;
- координировать вопросы технического обеспечения системы аппаратными средствами, исходя из требований, предъявляемых БД к оборудованию;
- координировать работы системных программистов, разрабатывающих дополнительное программное обеспечение для улучшения эксплуатационных характеристик системы;
- координировать работы прикладных программистов, разрабатывающих новые прикладные программы, и выполнять их проверку и включение в состав ПО системы и т. п.

На рис. 5.2 представлен типовой состав группы администратора БД, отражающий основные направления деятельности специалистов.

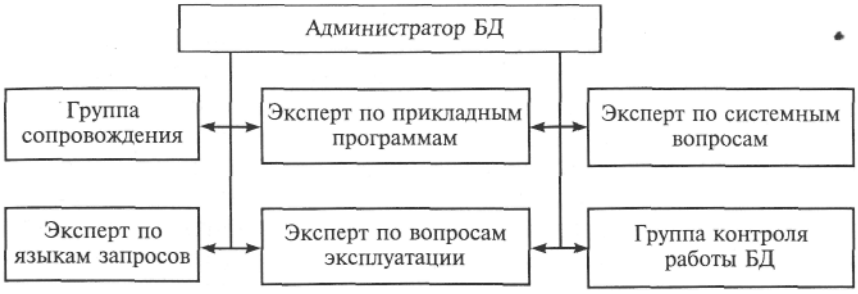


Рис. 5.2. Типовой состав группы администратора БД

5.2. Описательная модель предметной области

Процесс проектирования БД является весьма сложным. По сути, он заключается в определении перечня данных, хранимых на физических носителях (магнитных дисках и лентах), которые достаточно

полно отражают информационные потребности потенциальных пользователей в конкретной предметной области. Проектирование БД начинается с анализа предметной области и возможных запросов пользователей. В результате этого анализа определяется перечень данных и связей между ними, которые адекватно—с точки зрения будущих потребителей — отражают предметную область. Завершается проектирование БД определением форм и способов хранения необходимых данных на физическом уровне.

Весь процесс проектирования БД можно разбить на ряд взаимосвязанных этапов, каждый из которых обладает своими особенностями и методами проведения. На рис. 5.3 представлены типовые этапы.

Рис. 5.3. Этапы проектирования БД

На *этапе инфологического (информационно-логического) проектирования* осуществляется построение семантической модели, описывающей сведения из предметной области, которые могут заинтересовать пользователей БД. Семантическая модель *(semantic model)* — представление совокупности о предметной области понятий в виде графа, в вершинах которого расположены понятия, в терминальных вершинах — элементарные понятия, а дуги представляют отношения между понятиями.

Сначала из объективной реальности выделяется предметная область, т. е. очерчиваются ее границы. Логический анализ выделенной предметной области и потенциальных запросов пользователей завершается построением инфологической модели — перечня сведений об объектах предметной области, которые необходимо хранить в БД, и связях между ними.

Анализ информационных потребностей потенциальных пользователей имеет два аспекта: определение собственно сведений об объектах предметной области; анализ возможных запросов к БД и требований по оперативности их выполнения.

Анализ возможных запросов к БД позволяет уточнить связи между сведениями, которые необходимо хранить. Пусть, например, в БД по учебному процессу института хранятся сведения об учебных группах, читаемых курсах и кафедрах, а также связи «учебные группы—читаемые курсы» и «читаемые курсы —кафедры». Тогда запрос о том, проводит ли некоторая кафедра занятия в конкретной учебной группе, может быть выполнен только путем перебора всех читаемых в данной группе курсов.

Хранение большого числа связей усложняет БД и приводит к увеличению памяти ЭВМ, но часто существенно ускоряет поиск нужной информации. Поэтому разработчику БД (администратору БД) приходится принимать компромиссное решение, причем процесс определения перечня хранимых связей, как правило, имеет итерационный характер.

Этап датологического проектирования подразделяется на логическое (построение концептуальной модели данных) и физическое (построение физической модели) проектирование.

Главной задачей л о г и ч е с к о г о п р о е к т и р о в а н и я является представление выделенных на предыдущем этапе сведений в виде данных в форматах, поддерживаемых выбранной СУБД.

Задача ф и з и ч е с к о г о п р о е к т и р о в а н и я — выбор способа хранения данных на физических носителях и методов доступа к ним с использованием возможностей, предоставляемых СУБД.

Инфологическая модель «сущность—связь» *(entity-relationship model; ER-model)* П.Чена представляет собой описательную (неформальную) модель предметной области, семантически

определяющую в ней сущности и связи [44].

Относительная простота и наглядность описания предметной области позволяет использовать ее в процессе диалога с потенциальными пользователями с самого начала инфологического проектирования. Построение инфологической модели П.Чена, как и любой другой модели, является творческим процессом, поэтому единой методики ее создания нет. Однако при любом подходе к построению модели используют три основных конструктивных элемента: сущность, атрибут, связь.

Сущность — это собирательное понятие некоторого повторяющегося объекта, процесса или явления окружающего мира, о котором необходимо хранить информацию в системе. Сущность может определять как материальные (например, «студент», «грузовой автомобиль» и т.п.), так и нематериальные объекты (например, «экзамен», «проверка» и т.п.). Главной особенностью сущности является то, что вокруг нее сосредоточен сбор информации в конкретной предметной области. Тип сущности определяет набор однородных объектов, а экземпляр сущности — конкретный объект в наборе. Каждая сущность в модели П.Чена именуется. Для идентификации конкретного экземпляра сущности и его описания используется один или несколько атрибутов.

Атрибут — это поименованная характеристика сущности, которая принимает значения из некоторого множества значений [46]. Например, у сущности «студент» могут быть атрибуты «фамилия», «имя», «отчество», «дата рождения», «средний балл за время обучения» и т. п.

Связи в инфологической модели выступают в качестве средства, с помощью которого представляются отношения между сущностями, имеющими место в предметной области. При анализе связей между сущностями могут встречаться бинарные (между двумя сущностями) и в общем случае *n*-арные (между *n* сущностями) связи. Например, сущности «отец», «мать» и «ребенок» могут находиться в 3-арном отношении «семья» («является членом семьи»).

Связи должны быть поименованы; между двумя типами сущностей могут существовать несколько связей.

Наиболее распространены бинарные связи. Учитывая, что любую *n*-арную связь можно представить в виде нескольких бинарных, подробнее остановимся именно на таких связях между двумя типами сущностей, устанавливающими соответствие между множествами экземпляров сущностей.

Различают четыре типа связей:

- один к одному (1: 1);
- один ко многим (1: M);
- многие к одному (M: 1);
- многие ко многим (M: N).

Связь один к одному определяет такой тип связи между типами сущностей *А и Б,* при котором каждому экземпляру сущности *А* соответствует один и только один экземпляр сущности *В,* и наоборот. Таким образом, имея некоторый экземпляр сущности *А,* можно однозначно идентифицировать соответствующий ему экземпляр сущности *В,* а по экземпляру сущности *В* — экземпляр сущности *А.* Например, связь типа 1: 1 «имеет» может быть определена между сущностями «автомобиль» и «двигатель», так как на конкретном автомобиле может быть установлен только один двигатель и один двигатель, естественно, нельзя установить сразу на несколько автомобилей.

Связь один ко многим определяет такой тип связи между типами сущностей *А и В,* для которой одному экземпляру сущности *А* может соответствовать 0, 1 или несколько экземпляров сущности *В,* но каждому экземпляру сущности *В* соответствует один экземпляр сущности *А.* При этом однозначно идентифицировать можно только экземпляр сущности *А* по экземпляру сущности *В.* Примером связи типа 1 : М является связь «учится» между сущностями «учебная группа» и «студент». Для такой связи, зная конкретного студента, можно однозначно идентифицировать учебную группу, в которой он учится, или, зная учебную группу, можно определить всех обучающихся в ней студентов.

Связь многие к одному по сути эквивалентна связи один ко многим. Различие заключается лишь в том, с точки зрения какой сущности *(А* или *В)* данная связь рассматривается.

Связь многие ко многим определяет такой тип связи между типами сущностей *А* и *В,* при котором каждому экземпляру сущности *А* может соответствовать 0, 1 или несколько экземпляров сущности *В,* и наоборот. При такой связи, зная экземпляр одной сущности, можно указать все экземпляры другой сущности, относящиеся к исходному, т. е. идентификация сущностей неуникальна в обоих направлениях. В качестве примера такой связи можно рассмотреть связь «изучает» между сущностями «учебная дисциплина» и «учебная группа».

Реально все связи являются *двунаправленными,* т.е., зная экземпляр одной из сущностей, можно

идентифицировать (однозначно или многозначно) экземпляр (экземпляры) другой сущности. В некоторых случаях целесообразно рассматривать лишь однонаправленные связи между сущностями в целях экономии ресурсов ЭВМ. Возможность введения таких связей полностью определяется информационными потребностями пользователей. Различают простую и многозначную однонаправленные связи, которые являются аналогами связей типа 1: 1 и 1 : M с учетом направления идентификации. Так, для простой однонаправленной связи «староста» («является старостой») между сущностями «учебная группа» и «студент» можно, зная учебную группу, однозначно определить ее старосту, но, зная конкретного студента, нельзя сказать, является ли он старостой учебной группы. Примером многозначной однонаправленной связи служит связь между сущностями «пациент» и «болезнь», для которой можно для каждого пациента можно указать его болезни, но нельзя выявить всех обладателей конкретного заболевания.

Введение однонаправленных связей означает, что в результате анализа потенциальных запросов потребителей установлено, что потребности в информации, аналогичной приведенной в двух последних примерах, у пользователей не будет (и они не будут формулировать соответствующие запросы к БД).

Графически типы сущностей, атрибуты и связи принято изображать прямоугольниками, овалами и ромбами соответственно. На рис. 5.4 представлены примеры связей различных типов; на рис. 5.5 и 5.6 — фрагменты инфологических моделей «студенты» (без указания атрибутов) и «учебный процесс факультета».

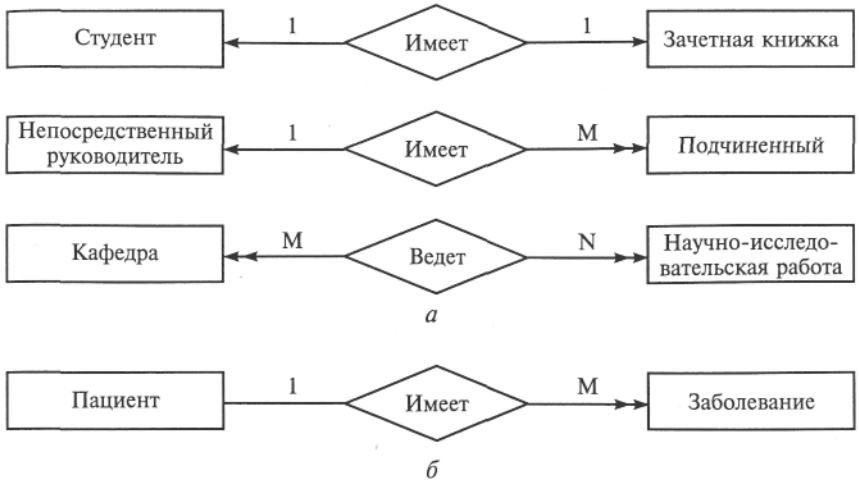


Рис. 5.4. Примеры связей между сущностями:
а — двунаправленные связи; *б* — однонаправленная связь

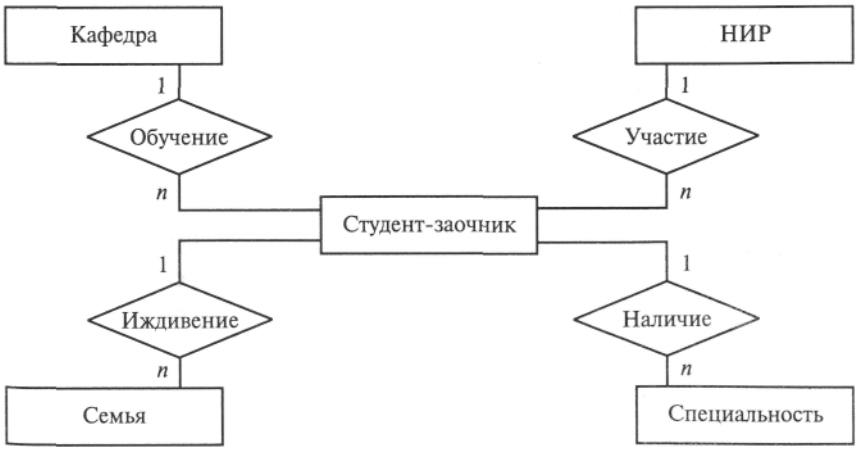


Рис. 5.5. Фрагмент ER-модели «Студенты»

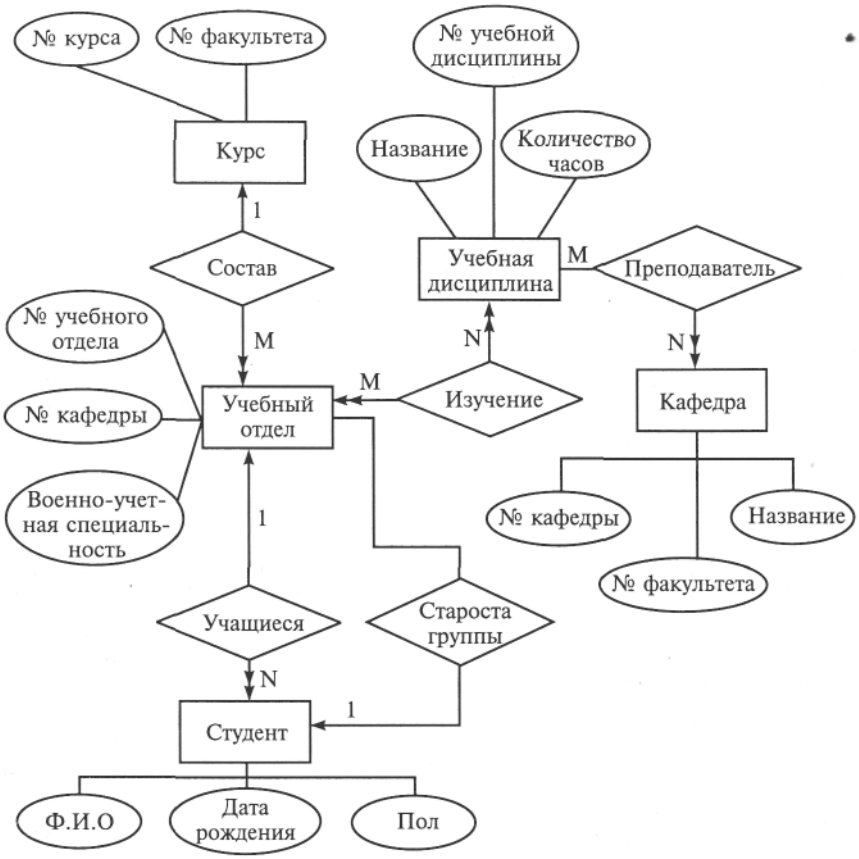


Рис. 5.6. Фрагмент ER-модели «Учебный процесс факультета»

Несмотря на то что построение инфологической модели есть процесс творческий, можно указать два основополагающих правила, которыми следует пользоваться всем проектировщикам БД [15, 44]:

- при построении модели должны использоваться только три типа конструктивных элементов: сущность, атрибут, связь;
- каждый компонент информации должен моделироваться только одним из приведенных выше конструктивных элементов для исключения избыточности и противоречивости описания.

Моделирование предметной области начинают с выбора сущностей, необходимых для ее описания. Каждая сущность должна соответствовать некоторому объекту (или группе объектов) предметной области, о котором в системе будет накапливаться информация. Существует проблема выбора конструктивного элемента для моделирования той или иной «порции» информации, что существенно затрудняет процесс построения модели. Так, информация о том, что некоторый студент входит в состав учебной группы, можно в модели представить:

- как связь: «входит в состав» для сущностей «студент» и «учебная группа»;
- как атрибут: «имеет в составе «студента» сущности «учебная группа»;
- как сущность: «состав учебной группы».

В этих случаях приходится рассматривать несколько вариантов и с учетом информационных потребностей пользователей разбивать предметную область на такие фрагменты, которые, с их точки зрения, представляют самостоятельный интерес.

При моделировании предметной области следует обращать внимание на существующий в ней документооборот. Именно документы, циркулирующие в предметной области, должны являться основой для формулирования сущностей. Это связано с двумя обстоятельствами. Во-первых, эти документы, как правило, достаточно полно отражают информацию, которую необходимо хранить в БД, причем в виде конкретных данных. Во-вторых, создаваемая ИС должна предоставлять пользователям привычную для них информацию в привычном виде, что в последующем существенно облегчит ввод БД в эксплуатацию.

При описании атрибутов сущности необходимо выбрать ряд атрибутов, позволяющих однозначно идентифицировать экземпляр сущности. Совокупность идентифицирующих атрибутов называют *ключом.*

Помимо идентифицирующих используются и описательные атрибуты, предназначенные для более полного определения сущностей. Число атрибутов (их тип) определяется единственным образом — на основе анализа возможных запросов пользователей. Существует ряд рекомендаций по работе с атрибутами [15, 44], например, по исключению повторяющихся групп атрибутов (рис. 5.7). Все они направлены на улучшение качества инфологической модели.

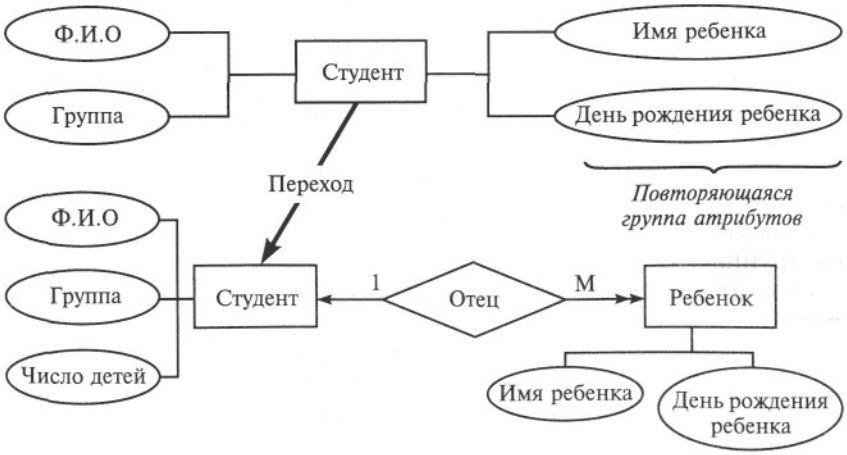


Рис. 5.7. Исключение повторяющейся группы атрибутов

При определении связей между сущностями следует избегать связей типа M : N, так как они приводят к существенным затратам ресурсов ЭВМ. Устранение таких связей предусматривает введение других (дополнительных) элементов — сущностей и связей. На рис. 5.8 приведен пример исключения связи «многие ко многим».

Рис. 5.8. Исключение связи типа M : N

В заключение приведем типовую последовательность работ (действий) по построению инфологической модели:

- выделение в предметной области сущностей;
- введение множества атрибутов для каждой сущности и выделение из них ключевых;
- исключение множества повторяющихся атрибутов (при необходимости);
- формирование связей между сущностями;
- исключение связей типа M: N (при необходимости);
- преобразование связей в однонаправленные (по возможности).

Помимо модели П.Чена, существуют и другие инфологические модели. Все они представляют собой описательные (неформальные) модели, использующие различные конструктивные элементы и соглашения по их использованию для представления в БД информации о предметной области. Иными словами, первый этап построения БД всегда связан с моделированием предметной области.

5.3. Концептуальные модели данных

В отличие от инфологической модели предметной области, описывающей по некоторым правилам

сведения об объектах материального мира и связи между ними, которые следует иметь в БД, *концептуальная модель* описывает хранимые в ЭВМ данные и связи. В силу этого каждая модель данных неразрывно связана с языком описания данных конкретной СУБД.

По существу, модель данных — это совокупность трех составляющих [15, 44]: типов структур данных, операций над данными, ограничений целостности.

Другими словами, модель данных представляет собой некоторое интеллектуальное средство проектировщика, позволяющее реализовать интерпретацию сведений о предметной области в виде формализованных данных в соответствии с определенными требованиями, т. е. средство абстракции, которое дает возможность увидеть «лес» (информационное содержание данных), а не отдельные «деревья» (конкретные значения данных).

Типы структур данных. Среди широкого множества определений, обозначающих типы структур данных, наиболее распространена терминология CODASYL (Conference of DAta SYstems Language) — международной ассоциации по языкам систем обработки данных, созданной в 1959 г.

В соответствии с этой терминологией используют пять типовых структур (в порядке усложнения):

- элемент данных;
- агрегат данных;
- запись;
- набор;
- база данных.

Дадим краткие определения этих структур [15, 44, 45].

Элемент данных — наименьшая поименованная единица данных, к которой СУБД может адресоваться непосредственно и с помощью которой выполняется построение всех остальных структур данных.

Агрегат данных — поименованная совокупность элементов данных, которую можно рассматривать как единое целое. Агрегат может быть простым или составным (если он включает в себя другие агрегаты).

Запись — поименованная совокупность элементов данных и (или) агрегатов. Таким образом, запись — это агрегат, не входящий в другие агрегаты. Запись может иметь сложную иерархическую структуру, поскольку допускает многократное применение агрегации.

Набор — поименованная совокупность записей, образующих двухуровневую иерархическую структуру. Каждый тип набора представляет собой связь между двумя типами записей. Набор определяется путем объявления одного типа записи «записью-владельцем», а других типов записей — «записями-членами». При этом каждый экземпляр набора должен содержать один экземпляр «записи-владельца» и любое количество «записей-членов». Если запись представляет в модели данных сущность, то набор — связь между сущностями. Например, если рассматривать связь «учится» между сущностями «учебная группа» и «студент», то первая из сущностей объявляется «записью-владельцем» (она в экземпляре набора одна), а вторая — «записью-членом» (их в экземпляре набора может быть несколько).

База данных — поименованная совокупность экземпляров записей различного типа, содержащая ссылки между записями, представленные экземплярами наборов.

Отметим, что структуры БД строятся на основании следующих основных композиционных правил [15, 44]:

- БД может содержать любое количество типов записей и типов наборов;
- между двумя типами записей может быть определено любое количество наборов;
- тип записи может быть владельцем и одновременно членом нескольких типов наборов.

Следование данным правилам позволяет моделировать данные о сколь угодно сложной предметной области с требуемым уровнем полноты и детализации.

Рассмотренные типы структур данных могут быть представлены в различной форме — графовой; табличной; в виде исходного текста языка описания данных конкретной СУБД.

Операции над данными. Операции, реализуемые СУБД, включают селекцию (поиск) данных и действия над ними. Селекция данных выполняется с помощью критерия, основанного на использовании или логической позиции данного (элемента, агрегата, записи) или значения данного, либо связей между данными [46]. Селекция на основе логической позиции данного базируется на упорядоченности данных в памяти системы. При этом критерии поиска могут формулироваться следующим образом:

- найти следующее данное (запись);
- найти предыдущее данное;
- найти *n*-е данное;
- найти первое (последнее) данное.

Этот тип селекции называют селекцией посредством текущей селекции, в качестве которой используется индикатор текущего состояния, автоматически поддерживаемый СУБД и, как правило, указывающий на некоторый экземпляр записи БД.

Критерий селекции по значениям данных формируется из простых или булевых условий отбора. Примерами простых условий поиска являются:

- ВОЕННО-УЧЕТНАЯ СПЕЦИАЛЬНОСТЬ = 200100;
- ВОЗРАСТ > 20;
- ДАТА < 19.04.2002 и т.п.

Булево условие отбора формируется путем объединения простых условий с применением логических операций, например:

- (ДАТА_РОЖДЕНИЯ < 28.12.1963) И (СТАЖ > 10);
- (УЧЕНОЕ_ЗВАНИЕ = ДОЦЕНТ) ИЛИ (УЧЕНОЕ ЗВАНИЕ = ПРОФЕССОР) и т.п.

Если модель данных, поддерживаемая некоторой СУБД, позволяет выполнить селекцию данных по связям, то можно найти данные, связанные с текущим значением какого-либо данного [46]. Например, если в модели данных реализована двунаправленная связь «учится» между сущностями «студент» и «учебная группа», можно выявить учебные группы, в которых учатся юноши (если в составе описания студента входит атрибут «пол»).

Как правило, большинство современных СУБД позволяют осуществлять различные комбинации описанных выше видов селекции данных.

Ограничения целостности. Эти логические ограничения на данные используются для обеспечения непротиворечивости данных некоторым заранее заданным условиям при выполнении операций над ними. По сути ограничения целостности — это набор правил, используемых при создании конкретной модели данных на базе выбранной СУБД.

Различают внутренние и явные ограничения.

Ограничения, обусловленные возможностями конкретной СУБД, называют *внутренними ограничениями целостности.* Эти ограничения касаются типов хранимых данных (например, «текстовый элемент данных может состоять не более чем из 256 символов» или «запись может содержать не более 100 полей») и допустимых типов связей (например, СУБД может поддерживать только так называемые функциональные связи, т.е. связи типа 1:1, 1: М или М: 1). Большинство существующих СУБД поддерживают прежде всего именно внутренние ограничения целостности [46], нарушения которых приводят к некорректности данных и достаточно легко контролируются.

Ограничения, обусловленные особенностями хранимых данных о конкретной ПО, называют *явными ограничениями целостности.* Эти ограничения также поддерживаются средствами выбранной СУБД, но они формируются обязательно с участием разработчика БД путем определения (программирования) специальных процедур, обеспечивающих непротиворечивость данных. Например, если элемент данных «зачетная книжка» в записи «студент» определен как ключ, он должен быть уникальным, т.е. в БД не должно быть двух записей с одинаковыми значениями ключа. Другой пример: пусть в той же записи предусмотрен элемент «военно-учетная специальность» и для него отведено шесть десятичных цифр. Тогда другие представления этого элемента данных в БД невозможны. С помощью явных ограничений целостности можно организовать как «простой» контроль вводимых данных (прежде всего на предмет принадлежности элементов данных фиксированному и заранее заданному множеству значений: например, элемент «ученое звание» не должен принимать значение «почетный доцент», если речь идет о российских ученых), так и более сложные процедуры (например, введение значения «профессор» элемента данных «ученое звание» в запись о преподавателе, имеющем возраст 25 лет, должно требовать, по крайней мере, дополнительного подтверждения).

Элементарная единица данных может быть реализована множеством способов, что, в частности, привело к многообразию известных моделей данных. Модель данных определяет правила, в соответствии с которыми структурируются данные. Обычно операции над данными соотносятся с их структурой.

Разнообразие существующих моделей данных соответствует разнообразию областей применения и

предпочтений пользователей.

В специальной литературе встречается описание довольно большого количества различных моделей данных. Хотя наибольшее распространение получили иерархическая, сетевая и, бесспорно, реляционная модели, вместе с ними следует упомянуть и некоторые другие.

Используя в качестве классификационного признака особенности логической организации данных, можно привести следующий перечень известных моделей:

- иерархическая модель данных;
- сетевая модель данных;
- реляционная модель данных;
- бинарная модель данных;
- семантическая сеть.

Рассмотрим основные особенности перечисленных моделей.

Иерархическая модель данных. Наиболее давно используемой (можно сказать классической) является модель данных, в основе которой лежит иерархическая структура типа дерева. Дерево — это орграф, в каждую вершину которого кроме первой (корневой), входит только одна дуга, а из любой вершины (кроме конечных) может исходить произвольное число дуг. В иерархической структуре подчиненный элемент данных всегда связан только с одним исходным.

На рис. 5.9 показан фрагмент объектной записи в иерархической модели данных. Часто используется также «упорядоченное дерево», в котором значим относительный порядок поддеревьев.

Рис. 5.9. Иерархическая модель данных

Достоинства такой модели несомненны: простота представления предметной области, наглядность, удобство анализа структур и простота их описания. К недостаткам следует отнести сложность добавления новых и удаления существующих типов записей, невозможность отображения отношений, отличающихся от иерархических, громоздкость описания и информационную избыточность.

Характерные примеры реализации иерархических структур — язык COBOL и СУБД семейства IMS (создана в рамках проекта высадки на Луну — «Аполлон») и System 2000 (S2K).

Сетевая модель данных. В системе БД, предложенных CODASYL, за основу была взята сетевая структура. Существенное влияние на разработку этой модели оказали более ранние сетевые системы — IDS и Ассоциативный ПЛ/1. Необходимость в процессе получения одного отчета обрабатывать несколько файлов обусловила целесообразность установления перекрестных ссылок между файлами, что в конце концов и привело к сетевым структурам [54].

Сетевая модель данных основана на представлении информации в виде орграфа, в котором в каждую вершину может входить произвольное число дуг. Вершинам графа сопоставлены типы записей, дугам — связи между ними. На рис. 5.10 представлен пример структуры сетевой модели данных.

Рис. 5.10. Сетевая модель данных

По сравнению с иерархическими сетевые модели обладают рядом существенных преимуществ: возможность отображения практически всего многообразия взаимоотношений объектов предметной области, непосредственный доступ к любой вершине сети (без указания других вершин), малая информационная избыточность. Вместе с тем в сетевой модели невозможно достичь полной независимости данных — с ростом объема информации сетевая структура становится весьма сложной для описания и анализа [54].

Известно, что применение на практике иерархических и сетевых моделей данных в некоторых случаях требует разработки и сопровождения значительного объема кода приложения, что иногда может вызвать перегрузку ИС [62].

Реляционная модель данных. В основе реляционной модели данных (см. подразд. 5.4) лежат не графические, а табличные методы и средства представления данных и манипулирования ими (рис. 5.11).

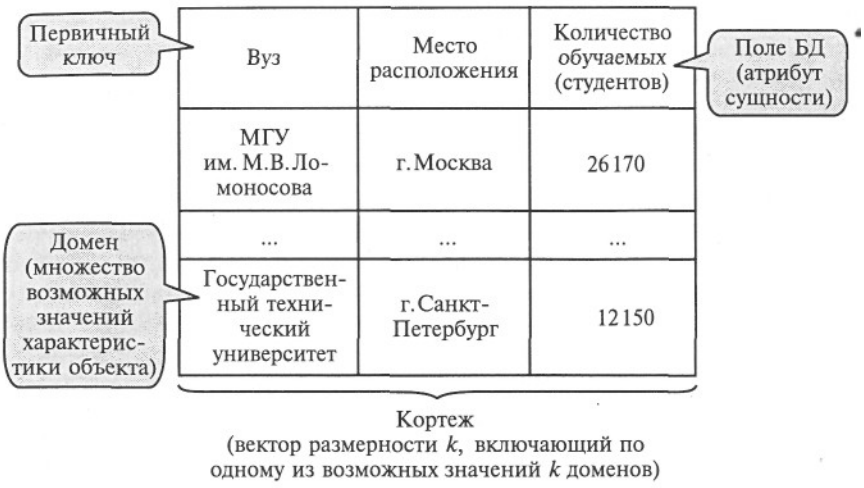


Рис. 5.11. Реляционная модель данных

В реляционной модели для отображения информации о предметной области используется таблица, называемая «отношением». Строка такой таблицы называется кортежем, столбец — атрибутом. Каждый атрибут может принимать некоторое подмножество значений из определенной области — домена [42].

Табличная организация БД позволяет реализовать ее важнейшее преимущество перед другими моделями данных, а именно возможность использования точных математических методов манипулирования данными, и прежде всего аппарата реляционной алгебры и исчисления отношений [54]. К другим достоинствам реляционной модели можно отнести наглядность, простоту изменения данных и организации разграничения доступа к ним.

Основным недостатком реляционной модели данных является информационная избыточность, что ведет к перерасходу ресурсов ВС (отметим, что существует ряд приемов, позволяющих в значительной степени избавиться от этого недостатка — см. гл. 6). Однако именно реляционная модель данных находит все более широкое применение в практике автоматизации информационного обеспечения профессиональной деятельности.

Подавляющее большинство СУБД, ориентированных на персональные ЭВМ, являются системами, построенными на основе реляционной модели данных — так называемыми «реляционными» СУБД.

Бинарная модель данных. Это графовая модель, в которой вершины являются представлениями простых однозначных атрибутов, а дуги — представлениями бинарных связей между атрибутами (рис. 5.12

Рис. 5.12. Бинарное отношение

Бинарная модель не получила особо широкого распространения, но в ряде случаев находит практическое применение.

Так, в области ИИ уже давно ведутся исследования с целью представления информации в виде бинарных отношений. Рассмотрим триаду (тройку) «объект — атрибут — значение» (более подробно об этом будет сказано в гл. 13). Триада «Кузнецов — возраст — 20» означает, что возраст некоего Кузнецова равен 20 годам. Эта же информация может быть выражена, например, бинарным отношением ВОЗРАСТ. Понятие бинарного отношения положено в основу таких моделей данных, как, например, Data Semantics и DIAM II.

Бинарные модели данных обладают возможностью представления связей любой сложности (и это их несомненное преимущество), но вместе с тем их ориентация на пользователя недостаточна [53].

Семантическая сеть. Семантические сети как модели данных были предложены исследователями, работавшими над различными проблемами ИИ (см. разд. IV). Так же, как в сетевой и бинарной моделях, базовые структуры семантической сети могут быть представлены графом, множество вершин и дуг которого образует сеть. Однако семантические сети предназначены для представления и систематизации знаний самого общего характера [53].

Таким образом, семантической сетью можно считать любую графовую модель (например, помеченный бинарный граф) при условии, что изначально четко определено, что обозначают вершины и дуги и как они используются.

Семантические сети являются богатыми источниками идей моделирования данных, чрезвычайно полезных в плане решения проблемы представления сложных ситуаций. Они могут быть использованы независимо или совместно с идеями, положенными в основу других моделей данных. Их интересной особенностью является то, что расстояние, измеренное на сети (семантическое расстояние или метрика), играет важную роль, определяя близость взаимосвязанных понятий. При этом предусмотрена возможность в явной форме подчеркнуть, что семантическое расстояние велико. Как показано на рис. 5.13, «СПЕЦИАЛЬНОСТЬ» соотносится с личностью «ПРЕПОДАВАТЕЛЬ», и в то же время «ПРЕПОДАВАТЕЛЮ» присущ «РОСТ». Взаимосвязь личности со специальностью очевидна, однако из этого не обязательно следует взаимосвязь «СПЕЦИАЛЬНОСТИ» и «РОСТА».

Рис. 5.13. Соотношение понятий в семантической сети

Следует сказать, что моделям данных типа семантической сети при всем присущем им богатстве возможностей при моделировании сложных ситуаций присуща усложненность и некоторая неэкономичность в концептуальном плане [53].

5.4. Реляционная модель данных

Как было отмечено в подразд. 5.3, в основе реляционной модели данных лежит их представление в виде таблиц, что в значительной степени облегчает работу проектировщика БД и — в последующем — пользователя в силу привычности и распространенности такого варианта использования информации.

Данная модель была предложена Э.Ф.Коддом *(E.F. Codd)* в начале 70-х гг. XX в., и вместе с иерархической и сетевой моделями составляет множество так называемых великих моделей. Можно сказать, что сегодня именно эта модель используется во всех наиболее распространенных СУБД.

Определение любой модели данных требует описания трех элементов:

- определение типов (структур) данных;
- определение операций над данными;
- определение ограничений целостности.

Сначала рассмотрим структуры данных и ограничения целостности, а затем более подробно остановимся на операциях реляционной алгебры.

Типы структур данных. Рассмотрение этого вопроса требует введения определений нескольких основных понятий.

Множество возможных значений некоторой характеристики объекта называется *доменом (domain):*

$$D_i = \{d_{i1},\ d_{i2},\ \ldots,\ d_{in}\}.$$

Например, в качестве домена можно рассматривать такие характеристики студента, как его фамилия, курс, рост и т.п.:

$$D_{\text{курс}} = \{1,\ 2,\ 3,\ 4,\ 5\};$$
$$D_{\text{фамилия}} = \{\text{Иванов, Петров, Сидоров, …}\}$$
$$D_{\text{рост}} = \{160, 161, 162, \ldots, 190\}.$$

Очевидно, что можно сопоставить понятия «атрибут» инфологической и «домен» реляционной моделей данных. Возможные значения характеристик объектов могут принимать числовые или текстовые значения, а их множества могут быть как конечными, так и бесконечными. Отметим, что в случае конечности домена можно организовать проверку явных ограничений целостности: в нашем примере домен $D_{\text{рост}}$ определяет, что все студенты должны иметь рост от 160 до 190 см, а номер курса не может превышать 5.

Вектор размерности *k,* включающий в себя по одному из возможных значений *k* доменов, называется *кортежем (tuple)*. Для приведенного выше примера кортежами являются

(1, Иванов, 172);
(3, Сидоров, 181);
(5, Уткин, 184).

Если в кортеж входят значения всех характеристик объекта предметной области (т. е. атрибутов сущности инфологической модели), ему можно сопоставить такую типовую структуру данных, как запись (объектная запись).

Декартовым произведением k доменов называется множество всех возможных значений кортежей

$$D = D_1 D_2 \ldots D_k.$$

Пусть для того же примера определены три домена:

$$D_1 = \{1,\ 4\};$$
$$D_2 = \{\text{Иванов, Петров}\};$$
$$D_3 = \{168,\ 181\}.$$

Тогда их декартовым произведением будет множество *D,* состоящее из восьми записей:

$$D = D_1 D_2 D_3 = \begin{Bmatrix} (1, \text{Иванов}, 168) \\ (1, \text{Иванов}, 181) \\ (1, \text{Петров}, 168) \\ (1, \text{Петров}, 181) \\ (4, \text{Иванов}, 168) \\ (4, \text{Иванов}, 181) \\ (4, \text{Петров}, 168) \\ (4, \text{Петров}, 181) \end{Bmatrix}.$$

При увеличении размерности любого из доменов увеличивается и размерность их декартова произведения. Так, если в первом домене определены три элемента $D_1 = \{1, 4, 5\}$, декартово произведение имеет вид

$$D = D_1 D_2 D_3 = \begin{Bmatrix} (1, \text{Иванов}, 168) \\ (1, \text{Иванов}, 181) \\ (1, \text{Петров}, 168) \\ (1, \text{Петров}, 181) \\ (4, \text{Иванов}, 168) \\ (4, \text{Иванов}, 181) \\ (4, \text{Петров}, 168) \\ (4, \text{Петров}, 181) \\ (5, \text{Иванов}, 168) \\ (5, \text{Иванов}, 181) \\ (5, \text{Петров}, 168) \\ (5, \text{Петров}, 181) \end{Bmatrix}.$$

Иными словами, декартово произведение — множество всех возможных комбинаций элементов исходных доменов.

Наконец, важнейшее определение: *отношением (relation) R,* определенным на множествах доменов $D_1, D_2, ..., D_k,$ называют подмножество их декартова произведения

$$R \subseteq D_1 D_2 \ldots D_k.$$

Элементами отношения являются кортежи. Отношение может моделировать множество однотипных объектов (сущностей), причем экземпляр сущности может интерпретироваться как кортеж. С помощью отношения можно моделировать и связи, в которых находятся объекты предметной области (сущности в ее инфологической модели). При этом кортеж такого отношения состоит из идентифицирующих атрибутов связываемых сущностей.

Таким образом, понятие «отношение» позволяет моделировать данные и связи между ними. В силу этого можно определить *реляционную базу данных* (РБД) как совокупность экземпляров конечных отношений.

Если учесть, что результат обработки любого запроса к РБД также можно интерпретировать как отношение (возможно, не содержащее ни одного кортежа), то возникает возможность построения ИС, основным инструментом которой будет алгебра отношений (реляционная алгебра). Любой запрос в такой системе может быть представлен в виде формулы, состоящей из отношений, объединенных операциями реляционной алгебры. Создав СУБД, обеспечивающую выполнение этих операций, можно разрабатывать И С, в которых любой запрос потребителя программируется формулой.

Ограничения целостности. Отношение может быть представлено таблицей, обладающей определенными свойствами (которые, по сути, и определяют внутренние ограничения целостности данных) [54]:

- каждая строка таблицы — кортеж;
- порядок строк может быть любым;

- повторение строк не допускается;
- порядок столбцов в отношении фиксирован.

Понятие «отношение» весьма схоже с понятием «файл данных». Поэтому в дальнейшем будем использовать следующую терминологию: отношение — файл; кортеж — запись; домен — поле. Идентификация конкретной записи файла осуществляется по ключу (набору полей, по значению которого можно однозначно идентифицировать запись). В файле можно определить несколько ключей. Один из них, включающий минимально возможное для идентификации записи число полей, называется *первичным ключом.*

Применительно к понятию «файл данных» внутренние ограничения целостности формулируются следующим образом:

- количество полей и их порядок в файле должны быть фиксированными (т. е. записи файла должны иметь одинаковые длину и формат);
- каждое поле должно моделировать элемент данных (неделимую единицу данных фиксированного формата, к которому СУБД может адресоваться непосредственно);
- в файле не должно быть повторяющихся записей.

СУБД, основанные на РБД, поддерживают и явные ограничения целостности. На практике они определяются зависимостями между атрибутами (см. разд. 5.2).

5.5. Операции реляционной алгебры

Операции реляционной алгебры лежат в основе языка манипулирования данными СУБД, основанных на РБД. Эти операции выполняются над файлами и в результатом их выполнения также является файл, который в общем случае может оказаться и пустым.

При описании операций реляционной алгебры будем использовать обозначения: ИФ (ИФ1; ИФ2) — имя исходного (первого исходного; второго исходного) файла; ФР — имя файла результата.

Некоторые операции накладывают на исходные файлы ограничения, которые в определенном смысле можно рассматривать как внутренние ограничения целостности.

Проектирование. Формальная запись:

ФР = *proj* [список имен полей] (ИФ).

Операция не накладывает ограничений на исходный файл. Операция предусматривает следующие действия:

- из ИФ исключаются все поля, имена которых отсутствуют в списке имен полей;
- из полученного файла удаляются повторяющиеся записи.

Пример. Пусть ИФ (КАДРЫ) содержит 4 поля:

Кадры

НОМЕР	ДОЛЖНОСТЬ	ФАМИЛИЯ	П/Я
01	Инженер	Петров	34 170
02	Инженер	Горин	11 280
03	Старший инженер	Сидоров	34 170
04	Начальник цеха	Фомин	27 220
05	Начальник цеха	Николаев	11 280

Требуется выполнить операцию

ФР = *proj* [П/Я] (КАДРЫ).

Тогда после выполнения операции получим результат

ФР

П/Я
34 170
11 280
27 220

Заметим, что с помощью приведенной операции можно выявить, в каких почтовых ящиках работают сотрудники, информация о которых содержится в данном файле.

Селекция (выбор). Формальная запись:

ФР = *sel* [условие] (ИФ).

Эта операция также не накладывает ограничений на ИФ. В ФР заносятся те записи из ИФ, которые удовлетворяют условию поиска. Условие представляет собой логическое выражение, связывающее значения полей ИФ.

Пример. Пусть для приведенного выше ИФ «КАДРЫ» требуется выявить сотрудников П/Я 34 170, имеющих должность «старший инженер». Для отработки такого запроса достаточно выполнить операцию:

ФР = *sel* [ДОЛЖНОСТЬ = «старший инженер» И П/Я =
= «34 170»] (КАДРЫ).

ФР

НОМЕР	ДОЛЖНОСТЬ	ФАМИЛИЯ	П/Я
03	Старший инженер	Сидоров	34 170

Отметим, что данная операция не изменяет структуру ИФ. Кроме того, при такой формальной записи операции предполагается, что СУБД поддерживает отработку сложных (составных) запросов, в противном случае пришлось бы составное условие поиска отрабатывать последовательно — сначала выявить сотрудников, имеющих должность «старший инженер», а затем из них выделить тех, кто работает на П/Я 34 170 (или наоборот). Иногда такой (последовательный) порядок поиска имеет определенные преимущества — прежде всего в тех случаях, когда на сложный запрос дан отрицательный ответ и непонятно, что послужило причиной этого (в нашем примере — или нет сотрудников должности «старший инженер», или никто из них не «работает» в указанном П/Я, или такого предприятия вообще «нет» в БД).

Соединение. Формальная запись:

ФР = ИФ1 $\triangleright\triangleleft$ ИФ2.
(список полей)

В реляционной алгебре определено несколько операций соединения. Мы рассмотрим так называемое *естественное соединение.*

Условием выполнения данной операции является наличие в соединяемых файлах одного или нескольких однотипных полей, по которым и осуществляется соединение (эти поля указываются в списке; если список пуст, соединение осуществляется по всем однотипным полям).

В ФР заносятся записи, являющиеся конкатенациями (от англ. *concatenate* — сцеплять, связывать) записей исходных файлов. Иными словами, в ФР попадают записи ИФ1 и ИФ2 с совпадающими значениями полей, по которым осуществляется соединение («сцепка»).

Пример 1. Пусть, помимо файла «КАДРЫ» имеется файл «ЦЕХ» в котором указаны порядковый НОМЕР сотрудника (как и в первом файле) и НОМЕР_ЦЕХА — номер цеха, в котором данный сотрудник работает.

ЦЕХ

НОМЕР	НОМЕР_ЦЕХА
01	Ц1
02	Ц2
03	Ц1
04	Ц2
05	Ц1

Тогда после выполнения операции

$$ФР = КАДРЫ >< ЦЕХ.$$

получим

ФР

НОМЕР	ДОЛЖНОСТЬ	ФАМИЛИЯ	П/Я	НОМЕР_ЦЕХА
01	Инженер	Петров	34 170	Ц1
02	Инженер	Горин	11 280	Ц2
03	Старший инженер	Сидоров	34 170	Ц1
04	Начальник цеха	Фомин	27 220	Ц2
05	Начальник цеха	Николаев	11 280	Ц1

Следует обратить внимание, что в формате команды не указаны поля соединения. Следовательно, оно осуществляется по единственному однотипному полю (НОМЕР).

Пример 2. Пусть требуется выяснить, в каком цехе п/я 34 170 работает старший инженер Сидоров.
Для этого требуется выполнить операции:

ФР1 = *sel* [ДОЛЖНОСТЬ = «старший инженер» И П/Я = «34 170» И ФАМИЛИЯ = «Сидоров»] (КАДРЫ);

ФР2 = ФР1 ▷◁ ЦЕХ;

ФР = *proj* [ДОЛЖНОСТЬ, ФАМИЛИЯ, П/Я, НОМЕР_ЦЕХА]ФР2 (ДОЛЖНОСТЬ).

В результате получим

ФР

ДОЛЖНОСТЬ	ФАМИЛИЯ	П/Я	НОМЕР_ЦЕХА
Старший инженер	Сидоров	34 170	Ц1

Объединение. Формальная запись:

$$ФР = ИФ1 \cap ИФ2.$$

Условием выполнения операции является однотипность (одинаковая структура) исходных файлов.
В файл результата заносятся неповторяющиеся записи исходных файлов.

Пример. Пусть в БД имеются два файла: УЧ_Д_КАФЕДРЫ_1 и УЧ_Д_КАФЕДРЫ_2, в которых содержатся данные о читаемых кафедрами № 1 и № 2 учебных дисциплинах:

УЧ_Д_КАФЕДРЫ_1

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
2011-12	99/ЭВ.3-02
5300-43	96/ЭИ.6-01
5140-11	98/ЭВ.4-03

УЧ_Д_КАФЕДРЫ_2

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
5110-15	97/ЭИ.5-02
5413-23	98/ЭВ.4-01
2010-19	01/ЭИ.1-03
5300-43	96/ЭИ.6-01

Тогда после выполнения операции объединения

$$ФР = УЧ\_Д\_КАФЕДРЫ1 \cap УЧ\_Д\_КАФЕДРЫ2$$

получим данные об учебных дисциплинах, читаемых обеими кафедрами:

ФР

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
5110-15	97/ЭИ.5-02
5413-23	98/ЭВ.4-01
2010-19	01/ЭИ.1-03
5300-43	96/ЭИ.6-01
2011-12	99/ЭВ.3-02
5140-11	98/ЭВ.4-03

Напомним, что последовательность записей в файлах БД роли не играет.

Разность (вычитание). Формальная запись:

$$ФР = ИФ1 - ИФ2.$$

Условием выполнения операции является однотипность (одинаковая структура) исходных файлов. В файл результата заносятся записи первого ИФ, которых нет во втором.

Пример. В условиях предыдущего примера выполним операцию

$$ФР = УЧ\_Д\_КАФЕДРЫ1 - УЧ\_Д\_КАФЕДРЫ2.$$

Получим данные об учебных дисциплинах, читаемых кафедрой № 1 без участия кафедры № 2.

ФР

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
2011-12	99/ЭВ.3-02
5140-11	98/ЭВ.4-03

Пересечение. Формальная запись:

$$ФР = ИФ1 \cup ИФ2.$$

Условием выполнения операции является однотипность (одинаковая структура) исходных файлов. В РФ заносятся записи, присутствующие в обоих ИФ.

Пример. Для уже известных файлов УЧ_Д_КАФЕДРЫ1 и УЧ_Д_КАФЕДРЫ2 выполним операцию пересечения

$$ФР = УЧ\_Д\_КАФЕДРЫ1 \cup УЧ\_Д\_КАФЕДРЫ2.$$

Получим данные о совместно читаемых обеими кафедрами дисциплинах:

ФР

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
5300-43	533

Деление. Формальная запись:

$$ФР = ИФ1 \div ИФ2.$$

Для выполнимости операции деления необходимо, чтобы в первом ИФ было больше полей, чем во втором, и для каждого поля второго ИФ существовало однотипное ему поле в первом ИФ.

В ФР, состоящий из полей первого ИФ, не входящих во второй, заносятся те записи, которые согласуются со всеми записями второго ИФ.

Пример. Пусть в БД хранятся два файла, содержащие данные об учебной литературе, выпущенной некоторой кафедрой.

АВТОРЫ

АВТОР	НАЗВАНИЕ	ГОД_ИЗДАНИЯ
Иванов	Теория вероятностей	1997
Петров	Методы оптимизации	1998
Петров	Теория вероятностей	1997
Иванов	Методы оптимизации	1998
Сидоров	Методы оптимизации	1998
Петров	Концепции естествознания	1996
Сидоров	Исследование операций	1999

ИЗДАНИЯ

НАЗВАНИЕ	ГОД_ИЗДАНИЯ
Теория вероятностей	1997
Методы оптимизации	1998

После выполнения операции деления первого файла на второй (а она возможна, так как в файле «АВТОРЫ» имеются все поля файла ИЗДАНИЯ) получим данные об авторах (соавторах), которые приняли участие в написании всех книг, информация о которых хранится во втором файле:

ФР

АВТОР
Иванов
Петров

Умножение. Формальная запись:

$$ФР = ИФ1 \times ИФ2.$$

Условием выполнения операции умножения является отсутствие в исходных файлах полей с одинаковыми именами.

В ФР, содержащий поля обоих ИФ, заносятся все возможные комбинации записей ИФ1 и ИФ2.

Пример. Пусть в БД хранятся данные об инженерах и старших инженерах (в файлах «СТАРШИЕ_ИНЖЕНЕРЫ» и «ИНЖЕНЕРЫ» соответственно).

СТАРШИЕ_ИНЖЕНЕРЫ

ДОЛЖНОСТЬ	ФАМИЛИЯ
Старший инженер Ц1	Иванов
Старший инженер Ц2	Петров

ИНЖЕНЕРЫ

ДОЛЖНОСТЬ	ФАМИЛИЯ
Инженер Ц1	Сидоров
Инженер Ц1	Леонидов
Инженер Ц3	Дмитриев

Требуется получить данные о возможных вариантах комплектования дежурных смен управления предприятием в составе одного старшего инженера и одного инженера.

Поскольку имена полей в ИФ1 и ИФ2 совпадают, необходимо в одном из них (например, в ИФ2) поля переименовать (например, вместо «ДОЛЖНОСТЬ» - «ДОЛЖНОСТЬ1»; вместо «ФАМИЛИЯ» - «ФАМИЛИЯ1»). Тогда после выполнения операции

$$ФР = СТАРШИЕ\_ИНЖЕНЕРЫ \times ИНЖЕНЕРЫ$$

получим:

ФР

ДОЛЖНОСТЬ	ФАМИЛИЯ	ДОЛЖНОСТЬ1	ФАМИЛИЯ1
Старший инженер Ц1	Иванов	Инженер Ц1	Сидоров
Старший инженер Ц1	Иванов	Инженер Ц1	Леонидов
Старший инженер Ц1	Иванов	Инженер Ц3	Дмитриев
Старший инженер Ц2	Петров	Инженер Ц1	Сидоров
Старший инженер Ц2	Петров	Инженер Ц1	Леонидов
Старший инженер Ц2	Петров	Инженер Ц3	Дмитриев

С помощью приведенных выше восьми операций реляционной алгебры можно найти ответ на любой запрос к БД, если, конечно, интересующие пользователя данные в ней хранятся. Типовые запросы могут быть запрограммированы заранее и отрабатываться как процедуры (транзакции). Обработка уникальных (нетиповых) запросов должна предусматривать оперативную разработку последовательности необходимых операций и последующую ее реализацию.

Глава 6. НОРМАЛИЗАЦИЯ ФАЙЛОВ БАЗЫ ДАННЫХ

6.1. Полная декомпозиция файла

Выше файлы рассматривались как своеобразные хранилища данных и связей между ними, причем было показано, что при соблюдении определенных правил эти файлы можно считать отношениями и применять к ним операции реляционной алгебры. Открытым пока остался вопрос о том, какие файлы хранить в БД и какие в них должны быть поля, чтобы иметь модель предметной области с определенными положительными свойствами.

Рассмотрим простой пример. Пусть имеется ИФ, в котором хранятся данные о сотрудниках, осуществлявших управление непрерывным производственным циклом предприятия в качестве оперативного дежурного (ОД) или его помощника (ПОД), и имеющих номера рабочих телефонов, указанные в поле «ТЕЛЕФОН»:

ИФ

ДЕЖУРСТВО	СОТРУДНИК	ТЕЛЕФОН
ПОД	Иванов	3-12
ОД	Сидоров	3-12
ОД	Фомин	8-44
ПОД	Семин	8-44

Найдем две проекции ИФ:

ПФ1 = *proj* [ДЕЖУРСТВО, СОТРУДНИК] (ИФ);
ПФ2 = *proj* [СОТРУДНИК, ТЕЛЕФОН] (ИФ).

ПФ1

ДЕЖУРСТВО	СОТРУДНИК
ПОД	Иванов
ОД	Сидоров
ОД	Фомин
ПОД	Семин

ПФ2

СОТРУДНИК	ТЕЛЕФОН
Иванов	3-12
Сидоров	3-12
Фомин	8-44
Семин	8-44

Нетрудно убедиться, что соединение этих двух проекций образует ИФ:

ПФ1 ▷◁ ПФ2 = ИФ.

Полной декомпозицией файла называется совокупность произвольного числа его проекций, соединение которых идентично ИФ.

Говоря о полной декомпозиции файла, следует иметь в виду два обстоятельства: во-первых, у одного и того же файла может быть несколько полных декомпозиций; во-вторых, не всякая совокупность проекций файла образует его полную декомпозицию.

Для последнего примера найдем другую проекцию ИФ:

ПФ3 = *proj* [ДЕЖУРСТВО, ТЕЛЕФОН] (ИФ).

ПФ3

ДЕЖУРСТВО	ТЕЛЕФОН
ПОД	3-12
ОД	3-12
ОД	8-44
ПОД	8-44

В результате соединенияПФ2 и ПФ3 получим файл результата

ПФ2 ▷◁ ПФ3 = ФР ≠ ИФ:

ФР

ДЕЖУРСТВО	ТЕЛЕФОН	СОТРУДНИК
ПОД	3-12	Иванов
ПОД	3-12	Сидоров
ОД	3-12	Иванов
ОД	3-12	Сидоров
ОД	8-44	*Фомин*
ОД	8-44	*Семин*
ПОД	8-44	*Фомин*
ПОД	8-44	*Семин*

В ФР курсивом выделены записи, которых не было в ИФ.

Методы анализа, позволяющие определить, образует ли данная совокупность проекций файла его полную декомпозицию, будут рассмотрены в подразд. 6.4.

Возможность нахождения полной декомпозиции файла ставит вопросы о том, в каком виде хранить данные в БД? дает ли декомпозиция файла какие-либо преимущества? в каких условиях эти преимущества проявляются? и т.п.

6.2. Проблема дублирования информации

В некоторых случаях замена ИФ его полной декомпозицией позволяет избежать дублирования информации.

Рассмотрим пример. Пусть в исходном файле (как и в примере в подразд. 6.1) хранятся данные о сотрудниках, дежуривших в составе оперативной группы управления предприятием («ДАТА» — дата дежурства; «ТЕЛЕФОН» — рабочий телефон сотрудника).

ИФ

ДАТА	СОТРУДНИК	ТЕЛЕФОН
11.01	Иванов	3-12
15.01	Сидоров	4-21
24.01	Сидоров	4-21
30.01	Сидоров	4-21
01.02	Иванов	3-12

Рассмотрим две проекции файла:

ПФ1 = *proj* [ДАТА, СОТРУДНИК] (ИФ);
ПФ2 = *proj* [СОТРУДНИК, ТЕЛЕФОН] (ИФ).

ПФ1

ДАТА	СОТРУДНИК
11.01	Иванов
15.01	Сидоров
24.01	Сидоров
30.01	Сидоров
01.02	Иванов

ПФ2

СОТРУДНИК	ТЕЛЕФОН
Иванов	3-12
Сидоров	4-21

Данные проекции образуют полную декомпозицию исходного файла. В ПФ2 номер рабочего телефона каждого сотрудника упоминается однократно, тогда как в ИФ — столько раз, сколько этот сотрудник заступал на дежурство. Очевидно, что для нашего примера разбиение ИФ на проекции позволяет избежать дублирования информации.

Устранение дублирования информации важно по двум причинам:

- устранив дублирование, можно добиться существенной экономии памяти;
- если некоторое значение поля повторяется несколько раз, то при корректировке данных необходимо менять содержимое всех этих полей, в противном случае нарушится целостность данных.

Для того чтобы найти критерий, позволяющий объективно судить о целесообразности использования полной декомпозиции файла с точки зрения исключения дублирования информации, воспользуемся понятием первичного ключа. Напомним, что первичным ключом называют минимальный набор полей файла, по значениям которых можно однозначно идентифицировать запись. Если значение первичного ключа не определено, то запись не может быть помещена в файл БД.

Можно показать, что для нашего примера проекции

ПФ1 = *proj* [ДАТА, СОТРУДНИК] (ИФ);
ПФ2 = *proj* [ДАТА, ТЕЛЕФОН] (ИФ)

образуют полную декомпозицию ИФ, однако они не исключают дублирования информации.

Причина этого заключается в том, что обе приведенные проекции содержат первичный ключ исходного файла (таковым в рассмотренном примере является поле «ДАТА», если, конечно, сотрудник не может одновременно находиться в двух и более оперативных группах).

Можно доказать, что дублирование информации неизбежно, если проекции, порождающие полную декомпозицию, содержат общий первичный ключ исходного файла.

Рассмотрим ИФ:

ИФ

FX	*FY*	*FZ*
x	*y*	*z*
x′	*y*	*z*

и две его проекции:

ПФ1

FX	*FY*
x	*y*
x′	*y*

ПФ2

FY	*FZ*
y	*z*

Для того чтобы вторая запись ИФ отличалась от первой (в противном случае имели бы в файле БД две одинаковые записи, что недопустимо), она формально должна быть представлена одним из семи вариантов:

(x, y, z); (x, y′, z); (x, y, z′); (x′, y′, z); (x, y′, z′); (x′, y′, z′); (x′, y, z′).

Пусть *FY*— первичный ключ. Для того чтобы дублирования информации не было, вторая запись ИФ должна быть или *(x′, y, z),* или *(x, y, z′),* но это противоречит тому, что *FY* — первичный ключ. Следовательно, для того, чтобы дублирования информации не было, необходимо исключить наличие первичного ключа ИФ в проекциях, образующих его полную декомпозицию.

Другими словами, если существует такая полная композиция файла, которая образована проекциями, не имеющими первичного ключа ИФ, то замена ИФ этой декомпозицией исключает дублирование

информации. Если же полная декомпозиция файла содержит проекции, имеющие общий первичный ключ ИФ, то замена его полной декомпозицией не исключает дублирования информации.

6.3. Проблема присоединенных записей

Рассмотрим использованный в подразд. 6.1 пример. Пусть в ИФ хранятся данные о сотрудниках, дежуривших в составе оперативной группы предприятия («ДАТА» — дата дежурства; «ТЕЛЕФОН» — рабочий телефон сотрудника).

ИФ

ДАТА	СОТРУДНИК	ТЕЛЕФОН
11.01	Иванов	3-12
15.01	Сидоров	4-21
24.01	Сидоров	4-21
30.01	Сидоров	4-21
01.02	Иванов	3-12

Рассмотрим две проекции файла:

ПФ1 = *proj* [ДАТА, СОТРУДНИК] (ИФ);
ПФ2 = *proj* [СОТРУДНИК, ТЕЛЕФОН] (ИФ).

ПФ1

ДАТА	СОТРУДНИК
11.01	Иванов
15.01	Сидоров
24.01	Сидоров
30.01	Сидоров
01.02	Иванов

ПФ2

СОТРУДНИК	ТЕЛЕФОН
Иванов	3-12
Сидоров	4-21

В ИФ поле «ДАТА» является ключом и не может быть пустым. Как поступить, если нужно запомнить данные о фамилии и номере рабочего телефона нового сотрудника, который еще не дежурил (например, о Смирнове с номером телефона 7-35)? Записать эти данные в ИФ нельзя (первичный ключ не может быть пустым), но можно поместить эти сведения в проекцию ПФ2. При этом ПФ2 формально перестает быть проекцией ИФ, хотя соединение ПФ1 и ПФ2 дает исходный файл (без сведений о Смирнове).

Записи, вносимые в отдельные проекции ИФ, называются *присоединенными.* Представление файла в виде его полной декомпозиции может позволить решить проблему присоединенных записей, но важно помнить, что соединение проекций ИФ может привести к их потере.

Целесообразность представления ИФ в виде полной декомпозиции с точки зрения решения проблемы присоединенных записей, как и проблемы дублирования информации, полностью определяется наличием или отсутствием в проекциях ИФ общего первичного ключа.

Пусть в ИФ БД хранятся данные о сотрудниках, исполняющих обязанности в дежурном расчете («НОМЕР_Р» — номер в составе дежурного расчета; «ТЕЛЕФОН» — номер рабочего телефона).

ИФ

НОМЕР_Р	СОТРУДНИК	ТЕЛЕФОН
1	Иванов	3-12
2	Сидоров	4-20
3	Фомин	8-61

Если считать, что один и тот же сотрудник не может исполнять обязанности нескольких номеров дежурного расчета, то в качестве первичного ключа можно использовать «НОМЕР_Р». Полную декомпозицию исходного файла составляют проекции:

ПФ1

НОМЕР_Р	СОТРУДНИК
1	Иванов
2	Сидоров
3	Фомин

ПФ2

НОМЕР_Р	ТЕЛЕФОН
1	3-12
2	4-20
3	8-61

В качестве присоединенных записей можно рассматривать либо добавление нового номера дежурного расчета и фамилии сотрудника, либо нового номера расчета и телефона без указания фамилии сотрудника, однако эту информацию можно внести и в ИФ путем формирования записей типа

НОМЕР_Р	СОТРУДНИК	ТЕЛЕФОН
4	Семин	

или

НОМЕР_Р	СОТРУДНИК	ТЕЛЕФОН
4		9-18

Таким образом, представление ИФ в виде проекций, содержащих общий первичный ключ ИФ, не дает преимуществ с точки зрения решения проблемы присоединенных записей.

Обобщая сказанное, можно сформулировать общее требование к файлу, представление которого в виде полной декомпозиции не имеет смысла.

Говорят, что файл находится в пятой нормальной форме (5 НФ), если у него или нет ни одной полной декомпозиции или нет ни одной полной декомпозиции, в которую входили бы проекции, не имеющие общего первичного ключа ИФ.

Если файл не находится в 5 НФ, имеется возможность избежать дублирования информации и потерю присоединенных записей, переходя от ИФ к такой его полной декомпозиции, которая образована проекциями, не содержащими первичный ключ. Если полученные таким образом файлы проекций не находятся в 5 НФ, то каждую из них можно заменить полной декомпозицией и т.д.

Процесс последовательного перехода к полным декомпозициям файлов БД называется *нормализацией файлов БД,* главная цель которой — исключение дублирования информации и потери присоединенных записей.

6.4. Функциональная зависимость полей файла

При обсуждении в предыдущих подразделах полной декомпозиции файла остался открытым вопрос о том, при каких же условиях некоторые проекции ИФ образуют его полную декомпозицию. Естественно, существует возможность взять конкретный файл, заполненный данными, и непосредственно проверить, образуют ли те или иные его проекции при соединении ИФ. Однако такой путь не является конструктивным, так как, во-первых, может оказаться достаточно много вариантов разбиения ИФ, и, во-вторых, что более важно, нет гарантии, что при добавлении записей в ИФ его

проекции будут по-прежнему составлять полную декомпозицию.

Очевидно, необходимо сформулировать критерий, позволяющий даже для незаполненного файла, исходя из возможных значений его полей, судить о возможности получения полной декомпозиции файла из тех или иных его проекций. Такой критерий строится на понятии функциональной зависимости полей файла [34].

Пусть *X* и *Y* — некоторые непересекающиеся совокупности полей файла. Говорят, что *Y* находится в функциональной зависимости от *X* тогда и только тогда, когда с каждым значением *X* связано не более одного значения *Y*.

Любые две записи файла, содержащие одинаковые значения *X,* должны содержать одинаковые значения *Y,* причем это ограничение действует не только на текущие значения записей файла, но и на все возможные значения, которые могут появиться в файле. Вместе с тем одинаковым значениям *Y* могут соответствовать различные значения *X*.

Рассмотрим уже знакомый пример. Пусть в ИФ имеются поля:

ДЕЖУРСТВО	ДОЛЖНОСТЬ	ФАМИЛИЯ	ТЕЛЕФОН

Поле «ТЕЛЕФОН» находится в функциональной зависимости от полей «ДОЛЖНОСТЬ» и «ФАМИЛИЯ» (считаем, что в данном файле не будет храниться информация о сотрудниках-однофамильцах, имеющих одинаковые должности). Понятно, что один и тот же номер рабочего телефона могут иметь несколько сотрудников, т.е. по значению поля «ТЕЛЕФОН» нельзя однозначно определить должность и фамилию сотрудника.

Пусть *X* состоит из нескольких полей. Говорят, что *Y* находится в полной функциональной зависимости от *X,* если *Y* функционально зависит от *X* и функционально не зависит от любого подмножества *X′,* не совпадающего с *X (X′* $\subset$ *X)* [54].

В условиях предыдущего примера поле «ТЕЛЕФОН» находится в полной функциональной зависимости от совокупности полей «ДОЛЖНОСТЬ» и «ФАМИЛИЯ», поскольку оно не зависит функционально ни от поля «ДОЛЖНОСТЬ», ни от поля «ФАМИЛИЯ» по отдельности.

Теперь можно сформулировать критерий (правило), по которому следует ИФ разбивать на проекции для получения его полной декомпозиции (это утверждение называют теоремой Хита).

Пусть имеются три непересекающиеся совокупности полей исходного файла: *H*, *J*, *K*. Если *K* функционально зависит от *J*, то проекции *proj [H*, *J]* (ИФ) и *proj [J, K]* (ИФ) образуют полную декомпозицию ИФ.

Докажем это утверждение. Введем вспомогательный файл

$$\text{ИФ1} = proj[H, J](\text{ИФ}) \triangleright\triangleleft proj[J, K](\text{ИФ}).$$

Покажем, что каждая запись ИФ присутствует в ИФ1, и наоборот.

1. Возьмем произвольную запись исходного файла: *(h, j, к).* Очевидно, что ее часть *(h, j)* принадлежит первой проекции, (*j*, *k*) — второй проекции ИФ. По определению операции соединения можно утверждать, что запись *(h, j, k)* должна присутствовать в файле ИФ1.

2. Возьмем произвольную запись вспомогательного файла *(h′, j′, k′).* Согласно определению файла ИФ1, можно записать: *proj [H*, *J]* (ИФ1) = *proj [H*, *J]* (ИФ). Следовательно, в файле ИФ должна находиться хотя бы одна запись типа (*h′*, *j′, k″),* где *k″* пока не определено. По аналогии можно записать: *proj [J, K]* (ИФ1) = *proj [J, K]* (ИФ). Следовательно, в файле ИФ должна находиться хотя бы одна запись типа *(h″, j′, k′),* где *h″* пока не определено.

Таким образом, в ИФ должны содержаться записи *(h′, j′, k″)* и *(h″*, *j′, k′).* Но поскольку *K* функционально зависит от *J*, можно заключить, что *k″* = *k′* и, следовательно, в ИФ имеется запись *(h′, j′, k′),* которую мы определили как произвольную запись ИФ1. Доказательство закончено.

Вернемся к примеру. Пусть в ИФ имеются поля:

ДЕЖУРСТВО	ДОЛЖНОСТЬ	ФАМИЛИЯ	ТЕЛЕФОН

Так как поле «ТЕЛЕФОН» находится в функциональной зависимости от полей «ДОЛЖНОСТЬ» и

«ФАМИЛИЯ», можно заключить, что полную декомпозицию ИФ следует искать в виде проекций:

ПФ1 = *proj* [ДЕЖУРСТВО, ДОЛЖНОСТЬ, ФАМИЛИЯ] (ИФ);
ПФ2 = *proj* [ДОЛЖНОСТЬ, ФАМИЛИЯ, ТЕЛЕФОН] (ИФ).

Для этого примера можно обозначить: H= [ДЕЖУРСТВО]; J = [ДОЛЖНОСТЬ, ФАМИЛИЯ]; K = [ТЕЛЕФОН].

Таким образом, с помощью понятия функциональной зависимости полей файла можно получать его полные декомпозиции без анализа хранящихся в файле данных еще на этапе проектирования БД, что весьма важно на практике.

6.5. Нормальные формы файла

Как было показано в подразд. 6.2 и 6.3, при некоторых условиях замена файла его полной декомпозицией позволяет исключить дублирование информации и решить проблему присоединенных записей. Таким условием является отсутствие в проекциях, образующих полную декомпозицию файла, общего первичного ключа ИФ. Теорема Хита создает основу для построения различных полных декомпозиций и поэтому может служить основным инструментом в процессе нормализации файлов БД.

При проведении нормализации на каждом шаге проверяется принадлежность файла некоторой нормальной форме. Если он принадлежит этой нормальной форме, проверяется, находится ли он в следующей, и далее до 5 НФ. Принадлежность файла некоторой форме задает необходимые, но недостаточные условия для нахождения в следующей по старшинству форме.

Еще раз подчеркнем основное достоинство механизма нормализации файлов с помощью исследования функциональной зависимости полей файла: возможность проведения этой операции на этапе проектирования БД.

Перечислим основные НФ файлов в соответствии с [54].

Первая нормальная форма (1 НФ). Файл находится в 1 НФ, если каждое его поле является *атомарным* (т.е. не содержит более одного значения), и ни одно из ключевых полей не является пустым. По существу принадлежность файла 1 НФ означает, что он соответствует понятию отношения и его ключевые поля заполнены.

Например, принципиально существует возможность хранить информацию о профессорско-преподавательском составе кафедры в следующем виде:

ДОЛЖНОСТЬ	ФАМИЛИЯ
Заведующий	Иванов
Профессор	Петров, Сидоров
Доцент	Фомин
Преподаватель	Семин, Савин, Федоров

Однако такой файл не находится в 1 НФ (так как поле «ФАМИЛИЯ» не является атомарным).

Вторая нормальная форма (2 НФ). Файл находится во 2 НФ, если он находится в 1 НФ и все его поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.

Третья нормальная форма (3 НФ). Файл находится в 3 НФ, если он находится во 2 НФ и ни одно из его неключевых полей не зависит функционально от любого другого неключевого поля.

Нормальная форма Бойса — Кодда (усиленная 3 НФ). Файл находится в НФ Бойса — Кодда, если любая функциональная зависимость между его полями сводится к полной функциональной зависимости от первичного ключа.

Можно показать [54], что рассмотренные НФ подчиняются правилу вложенности по возрастанию номеров, т.е. если файл находится в 5 НФ, он находится и в 3 НФ, 2 НФ, 1 НФ, и наоборот (рис. 6.1).

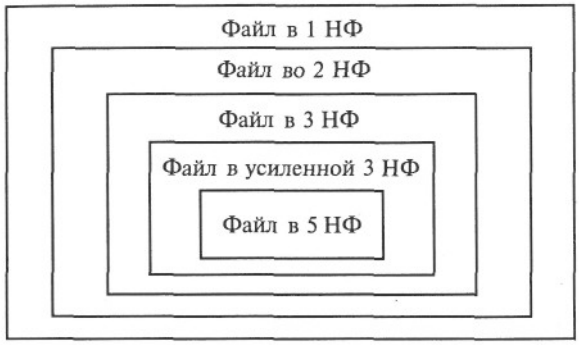


Рис. 6.1. Соотношение нормальных форм файлов

Помимо описанных выше нормальных форм, используется 4 НФ, основанная на понятии обобщенной функциональной зависимости [46]. На практике, приведя все файлы к нормальной форме Бойса — Кодда, можно с большой долей уверенности утверждать, что они находятся и в 5 НФ, т. е. что нормализация файлов БД завершена.

Отметим, что существующие СУБД (например, широко распространенная СУБД Access из пакета MS Office) содержат средства для автоматического выполнения операций нормализации (подобные мастеру по анализу таблиц), хотя качество этого анализа зачастую требуют последующего вмешательства разработчика БД [16, 59].

Необходимость нормализации файлов БД (кроме решения уже рассмотренных проблем исключения дублирования и потери присоединенных записей) определяется еще по крайней мере двумя обстоятельствами [43]: во-первых, разумным желанием группировать данные по их содержимому, что позволяет упростить многие процедуры в БД — от организации разграничения доступа до повышения оперативности поиска данных; во-вторых, стремлением разработать БД в виде множества унифицированных блоков, что может облегчить модернизацию отдельных частей базы, а также использовать таблицы одной БД в других.

Важным направлением совершенствования СУБД является их интеллектуализация, что подробнее будет рассмотрено в разд. IV.

Глава 7. СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ СЕТИ

7.1. Локальные вычислительные сети

Локальная вычислительная сеть (ЛВС) — это совокупность технических средств (компьютеров, кабелей, сетевых адаптеров и др.), работающих под управлением сетевой ОС и прикладного программного обеспечения [32].

Внутри одного кабинета, здания или в пределах небольшой территории ЛВС позволяет соединить между собой группу персональных компьютеров для совместного использования информационных ресурсов. Без ЛВС для обмена данными пришлось бы копировать файлы на дискеты или другие носители информации и передавать их другому пользователю. Такой метод переноса информации не позволяет нескольким пользователям одновременно работать с одними и теми же файлами данных. ЛВС предоставляет такую возможность, если используется многопользовательское прикладное программное обеспечение. Кроме простого разделения файлов, пользователи сети могут разделять принтер, накопитель для компакт-дисков (CD-ROM), модем, факсимильный аппарат, сканер или другое техническое устройство, совместимое с персональным компьютером и поддерживающее сетевой режим работы. Пользователи могут разделять ресурсы компьютеров и информацию, находясь на удалении друг от друга; они могут совместно работать над проектами и задачами, которые требуют тесной координации и взаимодействия. Кроме того, даже если компьютерная сеть выйдет из строя, возможно продолжение работы на вашем компьютере.

Существует два типа ЛВС (данная возможность поддерживается соответствующими операционными системами): *одноранговые сети* и *сети с выделенным сервером* (файл-сервером). Одноранговые сети не предусматривают выделение специальных компьютеров, организующих работу сети. Каждый пользователь, подключаясь к сети, выделяет в сеть какие-либо ресурсы (дисковое пространство, принтеры) и подключается к ресурсам, предоставленным в сеть другими пользователями. Такие сети

просты в установке, наладке; они существенно дешевле сетей с выделенным сервером. В свою очередь, сети с выделенным сервером, несмотря на сложность настройки и относительную дороговизну, позволяют осуществлять централизованное управление. В данном случае все компьютеры, кроме сервера, называются рабочими станциями.

Ниже перечислены семь задач [28], которые решаются с помощью персонального компьютера, работающего в составе ЛВС, и которые достаточно трудно решить с помощью отдельного компьютера.

1. *Разделение файлов.* ЛВС позволяет многим пользователям одновременно работать с одним файлом, хранящимся на центральном файл-сервере. Например, на предприятии или фирме несколько сотрудников могут одновременно использовать одни и те же руководящие документы.

2. *Передача файлов.* ЛВС позволяет быстро и надежно копировать файлы любого размера с одной машины на другую.

3. *Доступ к информации и файлам.* ЛВС позволяет запускать прикладные программы с любой из рабочих станций, где бы она ни была расположена.

4. *Разделение прикладных программ и БД.* ЛВС позволяет двум пользователям использовать одну и ту же копию программы. При этом, конечно, они не могут одновременно редактировать один и тот же документ или запись в БД.

5. *Одновременный ввод данных в прикладные программы.* Сетевые прикладные программы позволяют нескольким пользователям одновременно вводить данные, необходимые для работы этих программ. Например, вести записи в БД так, что они не будут мешать друг другу. Однако только специальные сетевые версии программ позволяют одновременный ввод информации. Обычные компьютерные программы позволяют работать с набором файлов только одному пользователю.

6. *Разделение принтера или другого технического устройства.* ЛВС позволяет нескольким пользователям на различных рабочих станциях совместно использовать один или несколько дорогостоящих лазерных принтеров или других устройств.

7. *Электронная почта.* Вы можете использовать ЛВС как почтовую службу и рассылать служебные записки, доклады, сообщения другим пользователям. В отличие от телефона электронная почта передаст ваше сообщение даже в том случае, если в данный момент абонент (группа абонентов) отсутствует на своем рабочем месте.

При помощи кабеля каждая рабочая станция соединяется с другими станциями и с сервером. В качестве кабеля используются «толстый» коаксиальный кабель («толстый» Ethernet), «тонкий» коаксиальный кабель («тонкий» Ethernet), витая пара, волоконно-оптический кабель. В последнее время все шире применяются ЛВС, в основе которых для связи между компьютерами используется инфракрасный сигнал. Очевидно, что компьютеры в этом случае должны находиться в пределах прямой видимости на небольшом расстоянии друг от друга (в пределах одного кабинета). «Толстый» кабель, в свою очередь, используется на участках большой протяженности при требованиях высокой пропускной способности. Волоконно-оптический кабель позволяет создавать протяженные участки без ретрансляторов при недостижимой с помощью других кабелей скорости и надежности. Однако стоимость кабельной сети на его основе высока, поэтому он не нашел пока широкого распространения в локальных сетях и применяется в более масштабных. В настоящее время локальные компьютерные сети в основном создаются на базе «тонкого» кабеля или витой пары.

Первоначально ЛВС создавались в своем большинстве по принципу «тонкого» Ethernet. В основе его — несколько компьютеров с сетевыми адаптерами, соединенные последовательно коаксиальным кабелем, причем все сетевые адаптеры выдают свой сигнал на него одновременно. Недостатки этого принципа выявились позже.

С ростом размеров ЛВС параллельная работа многих компьютеров на одну единую шину стала практически невозможной: стали значительными взаимные влияния персональных компьютеров друг на друга. Случайные выходы из строя коаксиального кабеля (например, внутренний обрыв жилы) надолго выводили всю сеть из строя. А определить место обрыва или возникновения программной неисправности, вызвавшей «зависание» сети, становилось практически невозможно.

Поэтому дальнейшее развитие ЛВС происходит на принципах структурирования. В этом случае каждая сеть складывается из набора взаимосвязанных участков — структур.

Каждая отдельная структура (отдельная рабочая группа) представляет собой несколько компьютеров с сетевыми адаптерами, каждый из которых соединен отдельным проводом — витой парой — с коммутатором (неким техническим распределительным устройством). При необходимости развития к ЛВС просто добавляют новую структуру (новую рабочую группу).

7.2. Всемирная информационная сеть Интернет

Сеть Интернет *(Internet)* из-за используемых высокоскоростных оптиковолоконных и (или) спутниковых каналов связи часто называют сетью супермагистралей, а из-за обилия информационных ресурсов и оперативного применения в сети новейших достижений компьютерных технологий — даже кибернетическим пространством. Организационно *Интернет* — это сеть, связывающая высокоскоростными каналами связи другие сети, например сети отдельных организаций, ЛВС и др.

По данным американского центра информации *NUA Internet Surveys* [62], во всем мире сейчас 148 млн пользователей Интернета. Из них в Канаде и США — 87 млн, в Европе — 33 млн. По оценкам *ROCIT (Russian Non-Profit Center for Internet Technologies)* и РосНИИРОС (Российский НИИ развития общественных сетей) число пользователей Интернета в России росло приблизительно следующим образом: январь 1997 г. — 300 тыс.; октябрь 1997 г. — 600 тыс.; июль 1998 г. — 1 млн. Удвоение траффика (объема пересылаемой сетью информации), по данным московских компаний — представителей сервиса Интернета (провайдеров), происходит каждые полгода, а количества пользователей — каждые 2 года. Недельная аудитория Интернета только в Москве составляет порядка 440 тыс. чел.

Для подключения к удаленным компьютерным сетям используются линии связи (телефонные или более совершенные). Процесс передачи данных по телефонным линиям происходит в форме электрических колебаний — аналога звукового сигнала, в то время как в компьютере информация хранится в виде кодов. Чтобы передать информацию от компьютера через телефонную линию, коды должны быть преобразованы в электрические колебания. Этот процесс носит название модуляции. Для того чтобы адресат смог прочитать на своем компьютере то, что ему отправлено, электрические колебания обратно преобразуются в машинные коды — демодулируются. Устройство, осуществляющее преобразование данных из цифровой формы, в которой они хранятся в компьютере, в аналоговую (электрические колебания), в которой они могут быть преданы по телефонной линии и обратно, называется *модем* (сокращенно от МОдулятор-ДЕМодулятор). Таким образом, отдельный компьютер при помощи специальной телекоммуникационной программы, управляющей модемом, связывается по телефонной линии с провайдером, а далее через провайдера по высокоскоростным каналам связи (оптиковолоконная или спутниковая связь) — с необходимым адресатом в Интернете.

Сеть Интернет появилась в США в результате исследования методов построения сетей, устойчивых к частичным повреждениям, получаемым, например, при бомбардировке их авиацией, и способных в таких условиях продолжать нормальное функционирование. В 60-х гг. XX в. исследователи начали экспериментировать с соединением множества различных типов компьютеров посредством телефонных линий, пользуясь фондами Агентства перспективных исследований *(ARPA)* Министерства обороны США. Созданная сеть получила название *ARPAnet (Advanced Research Projects Agency Network).* Она была предназначена для облегчения обмена информацией между военными ведомствами и их субподрядчиками по различным государственным проектам. Многие ведущие специалисты по компьютерам из промышленных организаций и академий получили доступ к данной сети благодаря *CSNET (Computer Science Network)* — проекту, созданному Национальным научным фондом *(NSF),* еще одним государственным агентством [34]. Вскоре все военные ведомства США были подключены к сети *ARPAnet,* что ознаменовало переход к ее практическому использованию. Агентство *ARPA* задалось целью проверить, можно ли соединить компьютеры, расположенные в разных местах на значительном расстоянии, при помощи новой технологии, получившей название «коммутация пакетов». Коммутация пакетов позволяла нескольким пользователям использовать один канал связи, посредством которого пакеты могли передаваться по сети к адресату, где восстанавливалось их исходное содержание. Прежде для работы в сети каждому компьютеру требовалась отдельная линия. Разработки, выполненные *NSF,* помогли создать высокоскоростную глобальную сеть, доступную для всех образовательных учреждений, государственных служащих, международных исследовательских организаций и т. п. Эта сеть позволила создать магистраль для передачи данных и подключить к ней множество компьютеров, совместно использующих один и тот же канал связи. Данные разбивались на пакеты, которые передавались на другую станцию. Каждому пакету присваивался компьютерный эквивалент места назначения (адрес) и временная метка, что позволяло передавать его в нужный пункт. Когда пакеты достигали адресата (пусть даже и по разным маршрутам), они собирались принимающим компьютером в связное сообщение.

Созданная на основе новой технологии сеть обеспечила независимую передачу данных между пунктами назначения и дала возможность компьютерам совместно использовать данные, а исследователям — обмениваться электронными сообщениями. Собственно изобретение электронной почты произвело революцию. До этого передача документов должна была осуществляться при помощи факсов, почтовых курьеров или государственной почты. Электронная почта, отправляемая через сеть, давала возможность отправлять подробные письма со скоростью и по ценам телефонного звонка.

По мере роста сети *ARPAnet* предприимчивые студенты разработали способ ее использования для проведения конференций в реальном времени. Сначала эти конференции имели научную тематику, но скоро они охватили практически все сферы интересов, поскольку люди оценили преимущества возможности общаться с сотнями или даже тысячами собеседников по всей стране, познакомившимися друг с другом при помощи электронной связи.

Сегодня эта сеть связывает компьютеры различных типов по всему миру при помощи протокола (стандарта передачи пакетов информации), получившего название *TCP/IP(Transmission Control Protocol / Internet Protocol).* В конце 70-х гг. XX в. были созданы каналы связи между *ARPAnet* и подобными ей сетями в других странах. Теперь мир был опутан компьютерной «паутиной» (общеизвестное сокращение WWW означает World Wide Web — всемирная паутина).

В 80-х гг. XX в. эта сеть сетей, получившая общее название «Интернет» (название произошло от англ. термина *internetworking* — межсетевое взаимодействие), стала расти с феноменальной скоростью. Тысячи исследовательских организаций и государственных ведомств, колледжей и университетов начали подключать свои компьютеры к этой всемирной сети.

Типовой адрес компьютера, используемого в Интернете, выглядит так:

www.name.ru

где www — информация об общепринятом соглашении использования адресатом протокола передачи и приема данных; name — (условное имя — название организации, предоставляющей или имеющей свой адрес в сети); ru — географическая привязка нахождения компьютера в мировой сети (ru — Россия, com и us — Америка и т.д.).

Подключение к Интернету возможно двумя способами. Первый, более простой — открыть у провайдера сервисов Интернета *(Internet Service Provider — ISP)* счет для получения доступа по телефонной линии. *ISP* может предоставить счет, обеспечивающий связь по протоколам *SLIP (Serial Line Internet Protocol)* или *PPP (Point-to-Point Protocol).*

Другой метод подключения к Интернету — подключение по выделенной линии. Этот способ более эффективен для больших организаций. Тип выделенной линии и скорость связи зависят от способа использования Интернетом в организации и требуемого для этого диапазона. Существует множество типов и скоростей выделенных линий: от линий со скоростью 56 Кбит/с до линий *ISDN* (цифровых сетей с интегрированными сервисами) и систем ретрансляции кадров, а также частичных или полных линий.

Вне зависимости от способа получения доступа к Интернету вам потребуется IP-адрес (IP — Internet Protocol) для учетной записи, обеспечивающей доступ к сети. Этот IP-адрес может выделяться провайдером либо *динамически* (это означает, что IP-адрес может меняться каждый раз при доступе к Интернету), либо *статически* (IP-адрес всегда остается одним и тем же).

Существует восемь основных путей использования сети Интернет.

1. Электронная почта.
2. Отправка и получение файлов с помощью FTP (File Transfer Protocol).
3. Чтение и посылка текстов в *Usenet.*
4. Поиск информации через GOPHER и WWW (World Wide Web).
5. Удаленное управление — запрос и запуск программ на удаленном компьютере.
6. Интернетпейджинг с помощью ICQ или аналогичных программ.
7. Chat-разговор с помощью сети IRC и электронной почты.
8. Видеоконференции и игровые формы работы через Интернет.

Перед более подробным раскрытием приведенных возможностей следует отметить, что развитие современных компьютерных технологий и разрабатываемого программного обеспечения имеет тенденцию сближения указанных возможностей и интеграции их в одном программном продукте. Множество программ, разрабатываемых для этих целей (таких, например, как Internet Explorer, Netscape

Communicator и др.), обеспечивающих отдельные функции Интернета, называются *«клиентами»*. Они удобны в использовании и предоставляют дружественный интерфейс для пользователей Интернета [38].

Электронная почта (e-mail). Отправка и получение писем остается пока наиболее популярным видом использования Интернета. В рамках Интернета для электронной почты существует система LISTSERV, позволяющая создавать группы пользователей с общей групповой адресацией. Таким образом, письмо, направленное на групповой адрес, будет получено всеми членами группы. Например, существует LISTSERV Netterain, объединяющий группу специалистов, обучающих пользованию Интернетом. Они объединились для того, чтобы обменяться идеями или задать вопросы своим коллегам, дать знать, что с ними можно связаться по электронной почте. В случае, если известно, что конкретное лицо или компания имеют адрес в Интернете, но сам адрес не известен, существуют способы узнать его с помощью системы NETFIND. Типовой адрес электронной почты выглядит так:

ignik@orc.ru

где ignik — определенное пользователем имя при организации электронного почтового ящика у провайдера; @ — разделительный знак в электронном адресе, называемый «собака»; orc — зарегистрированное имя сервера провайдера в сети Интернет.

Отправка и получение файлов с помощью FTP. При работе возможно использование FTP — одного из самых распространенных протоколов передачи файлов по Интернету. В начале это была терминальная программа с командной строкой, сейчас многие FTP-клиенты отличаются удобным интерфейсом и дополнительными возможностями, например возможностью «докачки» файла в случае обрыва связи.

Чтение и посылка текстов в Usenet. Usenet — это сеть информационных серверов в Интернете, часто называемой сетью новостей. В *Usenet* порядка 500 000 так называемых конференций (по сути, это каталог, куда стекаются со всего мира сообщения на определенную тему). Практически на любую тему в сети отведена своя собственная группа. Каждая отдельная группа сети *Usenet* называется *телеконференцией.* Серверы по всему миру, организованные в отдельную сеть, постоянно обмениваются между собой информацией, в результате происходит естественное обновление новостей. Среди множества телеконференцией *Usenet* имеются конференции, отражающие новости в науке (по отдельным ее областям) и в экономике. Многие телеконференции русифицированы.

Поиск информации в Интернете. Пользователь часто ищет информацию в Интернете либо чтобы узнать, имеется ли в мировых информационных ресурсах интересующий его материал по соответствующей теме и получить к данному материалу доступ, либо просто «осматривается» в информационном пространстве. Для работы в этих целях используются так называемые *«просмотрщики»*-браузеры (от англ. *browsing* — беспорядочное чтение) — специально ориентированные для этого программы-клиенты.

Значительная часть мировых информационных ресурсов представлена в Интернете. Можно потратить определенное количество времени, просто переходя с одного *сайта* (сервера Интернет) на другой и определяя, какая информация имеется в наличии. Эффект взрыва произвело в свое время появление таких средств управления поиском информации, как GOPHER и WWW. GOPHER использует систему меню, чтобы позволить пользователям осуществлять выбор информации. Наиболее развитое сегодня средство для поиска в Интернете — технология WWW. В последнее время часто под понятиями Интернет и Web подразумевают одно и то же. Технология Web позволяет свободно перемещаться среди информационных ресурсов (от одного документа на одном сервере Интернет к другому документу на другом сервере) по ключевым словам в документах, поддерживающих систему гипертекста и протокол HTTP. Переходы между документами (Web-страницами) осуществляются с использованием гипертекстовых связей, так называемых URL (Universal Resource Locators) — универсальных локаторов ресурсов. Большое число организаций и людей создают собственные элементы WWW, так называемые *Home Pages* (домашние Web-страницы), которые могут иметь гипертекстовые связи с информацией, находящейся на том же компьютере или на любом компьютере в сети Интернет. Для разработки Web-страниц используются специальные редакторы и язык разметки гипертекста HTML.

Интернет настолько велик и полон информационных ресурсов, что основной проблемой, с которой сталкиваются пользователи, является поиск нужных им данных. В дополнение к электронной почте, системам WWW и *Usenet* существуют и другие полезные инструменты, которые были созданы специально для помощи путешественникам по «информационному пространству». Наиболее полезным

из них является система FTP-клиент, позволяющая среди множества FTP-серверов (гигантских хранилищ архивов программ и документов) отыскивать необходимые архивы и программы.

Следует также сказать, что прогресс в компьютерных технологиях вызывает развитие специально ориентированных на поиск документов поисковых систем. Для этих целей в Интернете выделяются специальные серверы — мета-машины, обеспечивающие поиск информации по отдельным алгоритмам с явными и неявными критериями (по отдельным словам в документе или по их сочетаниям). Число таких машин постоянно растет. Наиболее известны из них мировые серверы Altavista (www.altavista.com), Yahoo (http://www.yahoo.com) и др. Среди русскоязычных наиболее популярны «Апорт!» (http://www.aport.ru), «Созвездие Интернет» (http://www.stars.ru), Яndex (http://www.yandex.ru) и др. Более того, поисковые серверы объединяются по определенным признакам (по темам и интересам) в свои сети и передают поисковые запросы друг через друга по всему миру.

Удаленное управление. Эта возможность очень полезна, когда при выполнении некоторой работы на отдельном компьютере требуются ресурсы больших систем. Существуют несколько различных типов удаленного управления. Некоторые из них работают на основе команд, подаваемых последовательно шаг за шагом (так называемых *ping* — пингов). Таким образом, исполнение запроса удаленного компьютера заключается в том, чтобы некоторая специфическая команда или их последовательность были выполнены на каком-то другом компьютере. Более развитые версии запросов сами выбирают систему и компьютер, которые будут к тому моменту в сети свободными. Существует также удаленный вызов процедуры, который позволяет программе запускать подпрограмму на другом компьютере и затем использовать результат ее работы.

Интернет-пейджинг с помощью ICQ. Система радиопейджинга давно известна и широко используется во всем мире. В 1998 г. разработчики израильской компании *Mirabilis* создали специальное программное обеспечение ICQ (*I seek you — я* тебя ищу) и перенесли возможности вызова абонентов (пейджинг) в сеть Интернет. Программа получила название «интернет-пейджера». У абонента интернет-пейджинга имеется уникальный ICQ-номер (UIN), по которому его можно вызвать. Стать абонентом сети достаточно просто — необходимо зарегистрироваться на сервере http://wwp.mirabilis.com или на русскоязычном сервере www.icq.ru. Серверы, поддерживающие ICQ, также часто объединяются в интернет-пейджинговые сети. Сервис позволяет вести переговоры в реальном времени вдвоем или многим пользователям одновременно, обмениваться сообщениями, пересылать файлы и т.д. Любой человек, не имеющий ICQ, может послать другому с ICQ сообщение на его ICQ-пейджер, либо, зайдя в сети на адрес http://wwp.mirabilis.com/******, где вместо звездочек его номер, либо послав e-mail на адрес *****@pager.mirabilis.com. Владелец ICQ имеет возможность заблокировать прием сообщений с www-пейджера или с email-express.

Возможность разговорить с многими людьми с помощью IRC. IRC (Internet relay chat) — это связка крупных сетей (Efhet, Dalnet, Undernet и др.), в каждой из которых сотни chat'ов и десятки тысяч пользователей. Официальный отсчет истории IRC ведется с 1988 г. Именно тогда один из финских студентов, некоторое время поговорив на многолинейных электронных досках объявлений (BBS), задался целью создать нечто похожее в Интернете, но более глобального масштаба. Тогда и появилась первая сеть IRC — Efhet. Сегодня серверы, поддерживающие IRC, объединены в единую сеть по всему миру.

Каналы в IRC *(channels)* можно сравнить с комнатами — вы «заходите» на канал, и после этого любая ваша фраза может быть услышана всеми, кто находится на том же канале — вне зависимости от того, что один ваш собеседник живет в Австралии, а другой — в Южной Африке. При необходимости вы можете общаться лично — ваше сообщение увидит только тот, кому вы его послали.

Видеоконференции и игровые формы работы через Интернет. Игровые формы работы (компьютерные игры в частности) занимают значительную часть времени пользователей. Именно компьютерные игры подталкивают разработчиков на создание и внедрение новых передовых компьютерных технологий и аппаратного обеспечения, а в некоторых сферах деятельности человека, например в экономике, игровые формы являются действенным способом апробации результатов — моделирования последствий принятых управленческих решений. На игровых принципах работы проводятся (моделируются) различные деловые игры. Играть можно против компьютера, против одного конкурента (человека) с помощью модема, против многих конкурентов с помощью локальных сетей или через Интернет. Для реализации специализированных форм работы необходима разработка специализированного программного обеспечения. Существует много серверов, которые предназначены исключительно для компьютерных игр, таких, как *Quake, Quake* II, *Team Fortress, Warcraft* III, *Starcraft*

и множество других. Имеются сведения, что сейчас разрабатываются и испытываются специализированные серверы, предназначенные исключительно для моделирования и обслуживания игровых форм работы в экономике. Для того чтобы качество игры было приемлемым, необходимо обеспечить стабильную и высокоскоростную связь с Интернетом.

Немалую пользу приносит организация между пользователями системы видеоконференций. Применение данного вида сервиса наиболее удачно для проведения оперативных совещаний и сбора докладов от подчиненных, находящихся на значительном расстоянии друг от друга. Работа в данном режиме позволяет пользователям видеть друг друга и даже демонстрировать друг другу различные предметы, образцы, документы. Весьма перспективным оказалось применение видеоконференций в медицине — при проведении консультаций, консилиумов и даже операций. Совершенно очевидна перспектива их использования и в экономике.

Объединение системы видеоконференций с одновременным проведением игровых форм работы предоставляет для пользователей новые возможности моделирования, обмена и уточнения информации.

7.3. Корпоративная сеть Интранет

Перечень услуг Интернета достаточно широк и разнообразен [43]. Желания разработчиков перенести данные возможности на большие внутриведомственные сети, а также обеспечить соответствующий режим защиты внутриведомственной информации вызвали необходимость создания новой сети, получившей название Интранет *(Intranet)*. Практика создания подобных сетей раскрыла главную их особенность для пользователей: объединить возможность работы пользователей над общими проектами с высоким уровнем сервиса Интернет. Итак, *Интранет* — это тот же Интернет, но организованная и работающая в рамках отдельной организации (корпорации). Именно поэтому Интранет и называют современной корпоративной сетью. Существуют различные типы сервисов (услуг), которые могут обеспечиваться в Интранете. Рассмотрим данные виды сервиса более подробнее, тем более что некоторые из них по своим возможностям шире, чем услуги Интернета [46, 47].

Почтовые сервисы. Существующая сетевая среда в рамках корпорации, как правило, уже располагает средствами пересылки сообщений между отдельными компьютерами и рабочими группами внутри организации, например, используя возможности ЛВС. Если это не так, то Интранет предоставляет возможность организовать для пользователей корпорации функционирование этого сервиса. Для этого необходимо выбрать почтовый пакет, поддерживающий электронную почту на основе Интернет-навигатора (клиента). При выборе почтового пакета может возникнуть необходимость в установке дополнительного протокола SMTP для обмена сообщениями между внутренней сетью организации и адресатами Интернета. Для поддержания этого сервиса потребуется выделенный либо совместно используемый сервер организации.

Файловые сервисы. При наличии в сети корпорации выделенного файл-сервера появляется возможность организации файлового сервиса (приема-передачи файлов между пользователями Интранета). Однако для работы Интранета на уровне возможностей Интернет потребуется установка на серверы корпорации дополнительного ПО для предоставления доступа к файлам на базе протокола FTP. Это может потребовать инсталляции на существующие файловые серверы и (или) рабочие станции протокола TCP/IP и набора необходимых программ.

Web-сервисы. World Wide Web — новейший вид сервиса, обеспечиваемый в рамках Интранета. Ради возможности использования данного сервиса внутри большой организации и началось развитие Интранет-сетей. Web-сервис может обеспечиваться в корпорации или отдельным Web-сервером или уже существующим файловым сервером. Для эффективной работы необходимо ПО Web-сервера. Также может возникнуть необходимость в расширении памяти или дискового пространства сервера. Кроме того, следует учитывать, что на работу и производительность сервера могут оказывать влияние другие программы или загружаемые модули рабочих станций сети. Любой аварийно завершающийся процесс остановит работу как файловых серверов, так и Интранета в целом.

Аудиосервис. Одним из преимуществ Интранета является надежность и доступность сетевого диапазона. Это дает возможность предоставления Интранет-услуг, которые ранее были бы недоступны в Интернете. Одной из таких услуг является передача аудиоданных по сети. В число аудиоуслуг может входить передача музыки, клиентских копий рекламных сообщений и даже выдержек из корпоративных заявлений или речей. Создав в Интранет-сети Web-страницу новостей, можно разместить на ней аудиозапись сообщений руководителя (начальника), ставящего и уточняющего задачу подчиненным,

или другую информацию.

Аудиосервис определяет выбор оборудования и операционной системы, с которой он будет работать. Поэтому, если использование аудиосервиса является важной качественной характеристикой Интранета, может возникнуть необходимость в пересмотре выбранного Интранет-сервера или включения в сеть дополнительного, специально выделенного сервера для установки на нем аудиосервиса.

Видеосервис. Для организации в Интранете качественного видеосервиса необходимо выделение специального видеосервера. Видеосервис не ограничен диапазоном Интранета. Видеосерверы могут обеспечивать несколько видеопотоков, что позволит одновременно отображать несколько видеоклипов на одной Web-странице. Видеоклипы могут содержать сведения об отдельных аспектах деятельности организации, новых достижениях, техническую информацию или предназначаться для обучения персонала и т.д.

Видеосерверы требуют большей мощности, чем традиционные Web-серверы, и должны устанавливаться на выделенной машине. Объем дискового пространства видеосервера тоже является важным фактором, поскольку видеоклипы сами по себе являются файлами большого объема. Как и аудиосервер, видеосервер определяет аппаратную платформу и операционную систему, с которой он будет работать. Если видеосервис также является важной частью Интранета, как и в случае аудиосервера, то может возникнуть необходимость добавления к сети специально выделенного видеосервера.

Еще одна особенность Интранета состоит в том, что данная сеть позволяет объединять компьютеры, изготовленные на различных аппаратных платформах и работающие на различных ОС. Последнее обстоятельство в сочетании с перечисленными достоинствами Интранет-сетей определяют их перспективность для массового использования в различных (в том числе экономических) организациях.

7.4. Сети электронных досок объявлений

Электронная доска объявлений (Bulletin Board System — BBS) физически представляет собой достаточно мощный компьютер со специальным ПО, позволяющим удаленному пользователю дистанционно обращаться к системе и во время связи (в режиме online) знакомиться с электронными объявлениями. Однако сегодня BBS — это уже не простая система обмена сообщениями, как это было в 80-х гг. прошлого века, когда возник этот английский термин. Современная BBS является мощным телекоммуникационным узлом, способным предоставить своим пользователям широкий спектр услуг, в котором сами по себе электронные объявления зачастую играют второстепенную роль [47]. Еще большие возможности открываются у пользователей при объединении BBS в единую сеть по тематическому, организационному, территориальному или иным признакам.

Предоставляемая пользователю информация на электронных досках объявлений строго структуризирована. Используемое на BBS ПО позволяет осуществлять оперативный поиск объявлений по ключевым словам, фразам, темам сообщений или их комбинации.

Как правило, узел BBS содержит большое количество полезных программных продуктов самой разной направленности, логически разбитых по тематике. При работе в системе в режиме online возможно ознакомление со списком предлагаемых файлов. Пользователь BBS в соответствии с установленным для него уровнем доступа на станцию может «перекачать» *(download)* на свой компьютер заинтересовавшую его информацию: от отдельных сообщений до необходимых пользователю файлов и программ, или «закачать» *(upload)* некоторую информацию. Помимо этого, на BBS доступны территории личной и публичной переписки между пользователями данной станции. Таким образом, можно размещать общие сообщения, рекламу, объявления о розыске ПО, анонимные послания и другую информацию.

За нарушение устанавливаемых на BBS правил по воле системного оператора — отвечающего за работу станции человека — можно лишиться дальнейшего доступа к данной BBS. Каждый зарегистрированный на BBS пользователь получает строго ограниченный системным оператором суточный период времени для реализации своих намерений. Этого иногда бывает недостаточно даже для того, чтобы принять список доступных на данной BBS файлов (так называемый *Filelist)*. При удовлетворении пользователем определенных потребностей или за другие заслуги системный оператор может повысить уровень доступа (Access Level) пользователя к данной BBS.

Существует множество классификаций узлов BBS. Они бывают любительскими или профессиональными, строго ориентированными на определенную тему или совокупность тем,

коммерческими и бесплатными, 24-часовыми и с ограниченным временем работы (как правило, работающие ночью; днем же — это обычный голосовой телефон), однолинейные и многолинейные и т.д.

К профессиональным станциям относятся крупные сетевые серверы или целые сети BBS, подобные Elvis, Izhma, Kiae, Simte, Chci и другие, а также небольшие узкоспециализированные станции или сети. Особенностью сетей BBS является то, что каждый узел в сети под общим ее названием имеет свой порядковый номер и, как правило, некоторую часть единого сервиса, характерного для всей сети в целом. Отдельные узлы сети могут иметь шлюзы для выхода на другие сети или отдельные узлы BBS. Главные отличия организованных таким образом сетей заключается в предоставлении доступа по использованию информационных ресурсов за абонентскую плату, 24-часовой график работы, большой выбор предлагаемого ПО, совместимость данных BBS с внутренними ЛВС организаций, другой сервис. С развитием компьютерных технологий и проникновением Интернета во все сферы общества подобный сервис появляется и на серверах всемирной информационной сети.

7.5. Компьютерные сети на основе FTN-технологий

Наряду с достаточно широким распространением Интернета и Интранета большой популярностью пользуются сети, в основе которых при использовании информационных ресурсов лежит непосредственный доступ к информации по коммутируемым — чаще всего телефонным — каналам связи. Больше того, компьютерные сети на основе так называемых FTN-технологий были своего рода родителями идей развития современных сетей Интернет и Интранет. В свое время широкое распространение персональных компьютеров и быстрое внедрение новых недорогих средств связи (модемов) сделало возможной передачу данных по телефонным линиям напрямую от одного компьютера к другому без промежуточных звеньев в виде больших машин или дополнительных технических устройств; при этом удаленность отправителя от адресата имела малое (или вообще не имела) значение. Каждый пользователь получил возможность предоставлять другим информационные услуги. Так создавались компьютерные сети передачи данных с добровольным распределением обязанностей по обмену информацией. Первая такая сеть появилась всего через три года после выхода на рынок первых IBM PC. Это была сеть *FIDOnet,* задуманная именно для объединения персональных компьютеров, используемых в качестве независимых телекоммуникационных систем.

Неформальный дух сети проявился уже в ее названии: создатель сети Том Дженнингс назвал сеть в честь своей собаки *Fido,* изображение которой стало символом *FIDOnet.* С самого начала сеть носила и носит любительский и некоммерческий характер. Участники сети тратят свои собственные деньги и время, чтобы она работала в интересах всех ее пользователей.

Технология *FIDOnet* оказалась столь популярной, что на ее основе по всему миру созданы и функционируют около тысячи любительских и коммерческих телекоммуникационных сетей, совместимых с *FIDOnet* по ПО, многие из них имеют шлюзы (ворота, каналы доступа) в *FIDOnet.* Около десятка подобных сетей ориентированы на экономические, научно-исследовательские и учебные цели. В сети *FIDOnet* также существует большое количество шлюзов с сетью Интернет. Еще на самом начальном этапе развития в структуру адресов *FIDOnet* была заложена иерархичность и многоуровневость, что позволило в дальнейшем разработать принципы децентрализованного управления и поддержки развития сети.

С момента возникновения *FIDOnet* ее технологические стандарты разрабатывались самими членами сети. Вначале это были просто дополнительные возможности, вводимые создателями первых программ для *FIDOnet,* однако со временем рост сети вызвал необходимость более жесткой стандартизации. С другой стороны, постоянно росло количество предлагаемых членами *FIDOnet* изменений и добавлений к технологии *FIDOnet.* Для решения возникших проблем был создан Комитет по стандартам технологии *FIDOnet (FIDOnet Technology Standards Comittee, FTSC,* или *FIDOnet Technology Network, FTN),* который за время своего существования разработал на основе многочисленных предложений членов сети несколько десятков стандартов различных компонентов технологии *FIDOnet.* Разработка новых стандартов продолжается и в настоящее время.

Изначально *FIDOnet* предназначалась для обмена личной электронной почтой между узлами, по сути, между операторами узлов. Вскоре была разработана технология эхоконференций (аналог телеконференции в сети Интернет). Данная технология позволила впервые объединить почтовые ящики разрозненных BBS или их сетей и создать общую систему электронного обмена информацией.

Технология эхоконференций дала мощный толчок развитию как *FIDOnet,* так и самих BBS — разработчики программного обеспечения BBS и почтовых программ *FIDOnet* стали обеспечивать в своих продуктах возможность интеграции BBS и узлов *FIDOnet* на одном компьютере, и *FIDOnet* стала похожа на «сеть BBS»: на большей части узлов *FIDOnet* были развернуты BBS, а большинство BBS стремились получить и получали адрес в сети *FIDOnet.* В настоящее время порядка 80 % узлов *FIDOnet* предоставляют доступ к своим ресурсам не только другим узлам сети в автоматическом режиме, но и пользователям BBS в интерактивном режиме. Однако *FIDOnet* была и остается именно сетью для автоматического обмена данными, и большинство крупных узлов *FIDOnet,* через которые проходят основные маршруты распространения почты, не поддерживают входящие звонки пользователей BBS.

Первое, что необходимо для того, чтобы достаточное количество телекоммуникационных узлов, объединенных в сеть, могли обмениваться информацией, — это наличие в сети определенной структуры. В *FIDOnet* структура определяется в первую очередь сетевым адресом узла. Адрес узла в *FIDOnet* (и любой FTN-совместимой сети) имеет числовую форму и строится по схеме: зона:-сеть_или_регион/узел .пойнт.

Узел (Node) является наименьшей структурной единицей *FIDOnet;* в то же время это основная единица *FIDOnet.*

Пойнт (Point) — пользователь FTN-сети, прикрепленный к узлу.

Сеть (Network) — это объединение узлов некой локальной географической области, обычно определяемое областью с удобной (т.е. бесплатной) телефонной связью между узлами сети.

Регион (Region) — это определенная достаточно крупная географическая область, включающая узлы, которые могут быть объединены либо не объединены в сети; типичный регион содержит множество узлов, объединенных в сети, и несколько независимых узлов, не являющихся частью какой-либо сети. В адрес сети, как правило, входит как составная часть адрес региона, которому принадлежит эта сеть.

Зона (Zone) — это наиболее крупная структурная единица *FIDOnet,* большая географическая область, включающая множество регионов и охватывающая одну или несколько стран и(или) континентов. *FIDOnet* насчитывает шесть зон: 1 — Северная Америка; 2 — Европа и территория бывшего СССР; 3 — Австралия и Океания; 4 — Южная Америка; 5 — Африка; 6 — Азия.

Таким образом, сетевая принадлежность конкретного узла, например 2 : 5020/113, определяется как узел 113 сети 5020 региона 50 зоны 2 *FIDOnet.* Географическое местоположение узла можно также определить из сетевого адреса: 2 — Европа, 50 — Россия, 20 — Москва.

Структура типовой сети *FIDOnet* представлена на рис. 7.1.

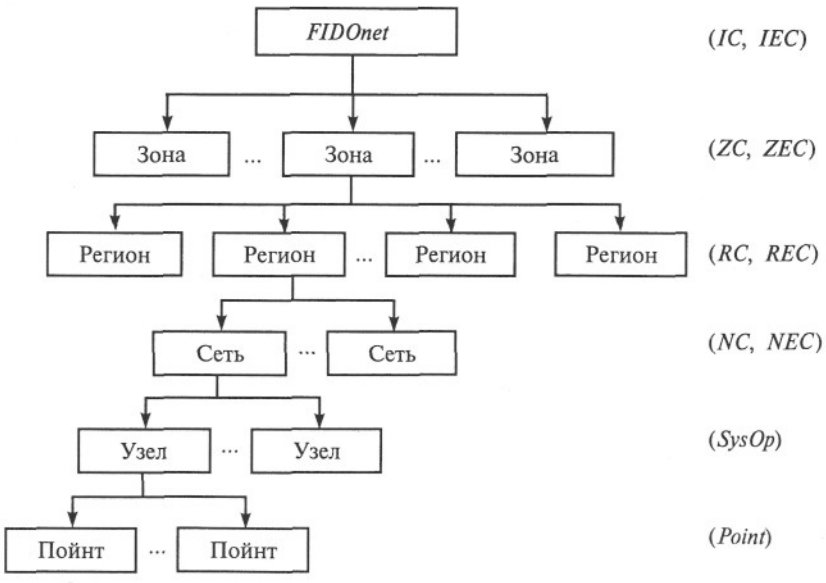


Рис. 7.1. Структура типовой FTN-совместимой сети:

IC — интернациональный координатор; *IEC* — интернациональный координатор эхоконференций; *ZC* — зональный координатор; *ZEC* — зональный координатор эхоконференций; *NC* — сетевой координатор; *NEC* — сетевой координатор эхоконференций; *SysOp* — системный оператор; *Point* — пользователь

По мере развития сети и самих модемов возникли и развились файловые конференции *(File-Echo*

Conference), где в качестве элементарных единиц выступали не сообщения, а файлы. Тем самым член сети, разработавший, по его мнению, гениальную программу, мог разослать ее посредством файловой конференции всем на нее подписанным пользователям. Правда, ежедневный поток (*Traffic)* в таких конференциях составляет от одного до нескольких мегабайтов в день, но существующие на данный момент мощности модемов позволяют поддерживать их без особых на то усилий. Число же официально существующих на сегодняшний день файловых конференций в *FIDOnet* превышает 2000. Для каждой из них разрабатываются определенные правила пользования конференцией, а пользователи сети голосованием выбирают модератора — ответственного за порядок в файловой конференции (эхоконференции). Основополагающим принципом *FIDOnet* является обеспечение возможности передачи данных напрямую от любого узла *FIDOnet* к любому другому узлу. Это обеспечивается распространением среди всех узлов сети списка-справочника узлов, или *нодлиста (Nodelist).* Нодлист представляет собой структурированное текущее описание узлов *FIDOnet* и, по сути дела, определяет саму сеть. Актуальность нодлиста поддерживается выпуском еженедельных файлов изменений и добавлений с рассылкой их по сети.

С расширением *FIDOnet* и ростом ее популярности появилось достаточно большое количество людей, стремящихся к общению в *FIDOnet,* желающих отправлять и принимать почту и файлы в автоматическом режиме, а не через BBS, но не имеющих возможности поддерживать узел *FIDOnet.* Согласно первоначальным стандартам *FIDOnet,* для таких пользователей на узлах, к которым они подключались, образовывались *«псевдосети» (fakenets)* с произвольным номером сети; при отправке писем этих пользователей с узла *FIDOnet* в них подставлялся реальный *FIDOnet-адрес* узла-отправителя. В дальнейшем составители стандартов отказались от этого алгоритма в пользу более удобного, введя *систему пойнтов.* Пойнт, посылающий почту через определенный узел, пользуется адресом узла, к которому через точку добавлен номер пойнта, например 2:5020/113.1.

Пойнт не обязан соблюдать технические процедуры, установленные для узла *FIDOnet.* Фактически пойнт представляет собой пользователя BBS, наделенного сетевым адресом и использующего *FIDOnet* - совместимое программное обеспечение для работы с почтой. В *FIDOnet* ведутся и распространяются списки пойнтов отдельных сетей в формате, аналогичном нодлисту.

Появление *FIDOnet* в России весной 1990 г. было вполне в духе сети — первой *FIDOnet*-совместимой почтовой системой на территории России был пойнт одного из польских узлов, расположенный в Новосибирске. Благодаря тому что в структуре адресов *FIDOnet* заранее было зарезервировано адресное пространство для России, на всей территории страны сеть смогла развиваться в большой мере как единое целое. По состоянию на апрель 1999 г. только в Московской сети *FID Onet* насчитывал ось более 1500 узлов и 22 000 пойнтов, более 150 других сетей, организованных на основе технологии FTN.

Можно с уверенностью сказать, что почти за десять лет *FIDOnet* в России стала не просто сетью электронной почты, а крупнейшим явлением, объединяющим сотни тысяч человек во всех концах страны. Российская *FIDOnet* предлагает пользователям русскоязычную среду для общения по самому широкому кругу вопросов, от сугубо технических до узконаправленных тем, включая экономические.

В заключение отметим, что дальнейшее развитие автоматизации информационного обеспечения деятельности должностных лиц в любой профессиональной сфере, в том числе и в экономической, немыслимо без применения сетевых технологий — от локальных до глобальных.

РАЗДЕЛ III. ТЕХНОЛОГИЯ МОДЕЛИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Глава 8. МЕТОДЫ МОДЕЛИРОВАНИЯ СИСТЕМ

8.1. Общие понятия и определения

Понятие модели является ключевым в общей теории систем. Моделирование как мощный, а часто и единственный метод исследования подразумевает замещение реального объекта другим — материальным или идеальным.

Важнейшими требованиями к любой модели являются ее адекватность изучаемому объекту в рамках конкретной задачи и реализуемость имеющимися средствами.

В теории эффективности и информатике *моделью объекта* (системы, операции) называется материальная или идеальная (мысленно представимая) система, создаваемая и/используемая при

решении конкретной задачи с целью получения новых знаний об объекте-оригинале, адекватная ему с точки зрения изучаемых свойств и более простая, чем оригинал, в остальных аспектах [5, 6].

Классификация основных методов моделирования (и соответствующих им моделей) представлена на рис. 8.1.

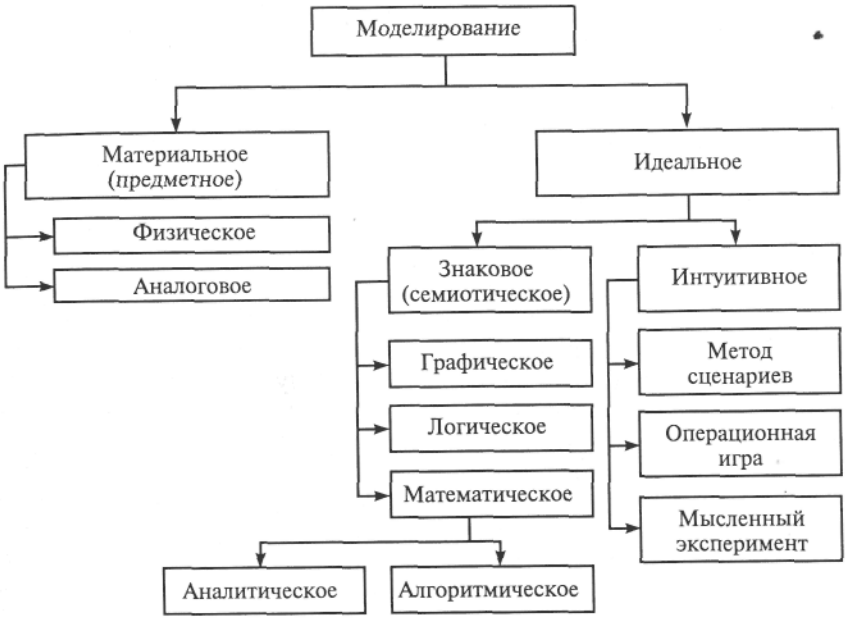


Рис. 8.1. Классификация методов моделирования

При исследовании экономических информационных систем (ЭИС) находят применение все методы моделирования, однако в этом разделе основное внимание будет уделено *семиотическим (знаковым) методам*. Напомним, что семиотикой называют науку об общих свойствах знаковых систем, т.е. систем конкретных или абстрактных объектов (знаков), с каждым из которых сопоставлено некоторое значение [35]. Примерами таких систем являются любые языки (естественные или искусственные, например языки описания данных или моделирования), системы сигнализации в обществе и животном мире и т. п. Семиотика включает три раздела: синтактика, семантика, прагматика.

Синтактика исследует синтаксис знаковых систем безотносительно к каким-либо интерпретациям и проблемам, связанным с восприятием знаковых систем как средств общения и сообщения.

Семантика изучает интерпретацию высказываний знаковой системы и с точки зрения моделирования объектов занимает в семиотике главное место.

Прагматика исследует отношение использующего знаковую систему к самой знаковой системе, в частности восприятие осмысленных выражений знаковой системы.

Из множества семиотических моделей в силу наибольшего распространения, особенно в условиях информатизации современного общества и внедрения формальных методов во все сферы человеческой деятельности, выделим *математические,* которые отображают реальные системы с помощью математических символов.

При этом, учитывая то обстоятельство, что мы рассматриваем методы моделирования применительно к исследованию систем в различных операциях, будем использовать общеизвестную методологию системного анализа, теории эффективности и принятия решений.

8.2. Математическая модель системы

Задача построения математической модели ЭИС может быть поставлена следующим образом [5, 53]: для конкретной цели моделируемой операции с учетом имеющихся ресурсов построить операторы моделирования исхода операции и оценки показателя ее эффективности. Формальная запись этой задачи имеет вид

$$<A_0,\ \theta;\ H,\ \psi>,$$

где A_0 — цель моделируемой операции; θ — ресурсы; H — оператор моделирования исхода операции; ψ

— оператор оценки показателя эффективности операции.

Перед рассмотрением каждого из названных операторов приведем два важных определения.

Оператором в математике называют закон (правило), согласно которому каждому элементу x множества X ставится в соответствие определенный элемент y множества Y. При этом множества X и Y могут иметь самую различную природу (если они представляют, например, множества действительных или комплексных чисел, понятие «оператор» совпадает с понятием «функция»).

Множество Z упорядоченных пар (x, y), где $x \in X$, $y \in Y$, называется *прямым произведением* множеств X и Y и обозначается $X \times Y$. Аналогично множество Z упорядоченных конечных последовательностей $(x_1, x_2, ..., x_n)$, где $x_k \in X_k$, называется прямым произведением множеств $X_1, X_2 ..., X_N$ и обозначается $Z = X_1 \times X_2 \times ... \times X_N$ [5, 12].

Оператором моделирования исхода операции называется оператор H, устанавливающий соответствие между множеством Λ учитываемых в модели факторов, множеством U возможных стратегий управления системой (операцией) и множеством Y значений выходных характеристик модели

$$H : \Lambda \times U \xrightarrow{A_0 \theta_м R_S} Y,$$

где $\theta_м$ — ресурсы на этапе моделирования исходов операции; R_s — учитываемые свойства моделируемой системы.

Оператором оценки показателя эффективности системы (операции) называется оператор ψ, ставящий в соответствие множеству Y значений выходных характеристик модели множество W значений показателя эффективности системы

$$\psi : Y \xrightarrow{A_0 \theta_э R_S} W,$$

где $\theta_э$ — ресурсы исследователя на этапе оценивания эффективности системы.

Особо отметим, что построение приведенных операторов всегда осуществляется с учетом главного системного принципа — *принципа цели.* Кроме того, важным является влияние объема имеющихся в распоряжении исследователя ресурсов на вид оператора моделирования исхода H и состав множества U стратегий управления системой (операцией). Чем больше выделенные ресурсы, тем детальнее (подробнее) может быть модель и тем большее число стратегий управления может быть рассмотрено (из теории принятия решений известно, что первоначально множество возможных альтернатив должно включать как можно больше стратегий, иначе можно упустить наилучшую).

В самом общем виде *математической моделью системы* (операции) называется множество

$$M = <U, \Lambda, H, Y, \psi, W>,$$

элементами которого являются рассмотренные выше множества и операторы.

Способы задания оператора ψ и подходы к выбору показателя эффективности W рассматриваются в теории эффективности; методы формирования множества возможных альтернатив — в теории принятия решений.

Для двух классов задач показатель эффективности в явном виде не вычисляется [27]:

- для *задач «прямой» оценки,* в которых в качестве показателей эффективности используются значения одной или нескольких выходных характеристик модели;
- *демонстрационных задач,* в ходе решения которых для изучения поведения системы используются лишь значения ее выходных характеристик и внутренних переменных.

В таких случаях используют термин *«математическое описание системы»,* представляемое множеством

$$M' = <U, \Lambda, H, Y>.$$

8.3. Классификация математических моделей

В качестве основного классификационного признака для ММ целесообразно использовать свойства

операторов моделирования исхода операции и оценивания показателя ее эффективности [12, 35].

Оператор моделирования исхода H может быть функциональным (т. е. заданным системой аналитических функций) или алгоритмическим (т. е. содержать математические, логические и логико-лингвистические операции, не приводимые к последовательности аналитических функций). Кроме того, он может быть детерминированным (когда каждому элементу множества $U \times \Lambda$ соответствует детерминированное подмножество значений выходных характеристик модели $\subseteq Y$ или стохастическим (когда каждому значению множества $U \times \Lambda$ соответствует случайное подмножество $\tilde{Y} \subseteq Y$).

Оператор, оценивания *показатель эффективности ψ,* может задавать либо точечно-точечное преобразование (когда каждой точке множества выходных характеристик *Y* ставится в соответствие единственное значение показателя эффективности *W),* либо множественно-точечное преобразование (когда показатель эффективности задается на всем множестве полученных в результате моделирования значений выходных характеристик модели).

В зависимости от свойств названных операторов все ММ делятся на три основных класса: аналитические, статистические, имитационные.

Для *аналитических моделей* характерна детерминированная функциональная связь между элементами множеств *U,* Λ, *Y,* а значение показателя эффективности *W* определяется с помощью точечно-точечного отображения. Аналитические модели имеют весьма широкое распространение. Они хорошо описывают качественный характер (основные тенденции) поведения исследуемых систем. В силу простоты их реализации на ЭВМ и высокой оперативности получения результатов такие модели часто применяются при решении задач синтеза систем, а также при оптимизации вариантов применения в различных операциях.

К *статистическим* относят ММ систем, у которых связь между элементами множеств *U,* Λ, *Y* задается функциональным оператором *H,* а оператор ψ является множественно-точечным отображением, содержащим алгоритмы статистической обработки. Такие модели применяются в тех случаях, когда результат операции является случайным, а конечные функциональные зависимости, связывающие статистические характеристики учитываемых в модели случайных факторов с характеристиками исхода операции, отсутствуют. Причинами случайности исхода операции могут быть случайные внешние воздействия; случайные характеристики внутренних процессов; случайный характер реализации стратегий управления. В статистических моделях сначала формируется представительная выборка значений выходных характеристик модели, а затем производится ее статистическая обработка с целью получения значения скалярного или векторного показателя эффективности.

Имитационными называются ММ систем, у которых оператор моделирования исхода операции задается алгоритмически. Когда этот оператор является стохастическим, а оператор оценивания показателя эффективности задается множественно-точечным отображением, имеем классическую имитационную модель, которую более подробно рассмотрим в гл. 9. Если оператор *H* является детерминированным, а оператор ψ задает точечно-точечное отображение, можно говорить об определенным образом вырожденной имитационной модели.

На рис. 8.2 представлена классификация наиболее часто встречающихся математических моделей по рассмотренному признаку.

Способ отображения	Ψ		Вид основных операторов H			
			Функциональный		Алгоритмический	
			детерминированный	стохастический	детерминированный	стохастический
		Множественно-точечное отображение	—	Статистические	—	Имитационные
		Точечно-точечное отображение	Аналитические	—	Имитационные	—

Рис. 8.2. Основная классификация математических моделей

Важно отметить, что при создании аналитических и статистических моделей широко используются их гомоморфные свойства (способность одних и тех же ММ описывать различные по физической природе процессы и явления). Для имитационных моделей в наибольшей степени характерен изоморфизм процессов и структур, т. е. взаимно однозначное соответствие элементов структур и процессов реальной системы элементам ее математического описания и соответственно модели.

Согласно [53], *изоморфизм* — соответствие (отношение) между объектами, выражающее тождество их структуры (строения). Именно таким образом организовано большее число классических имитационных моделей. Названное свойство имитационных моделей проиллюстрировано рис. 8.3.

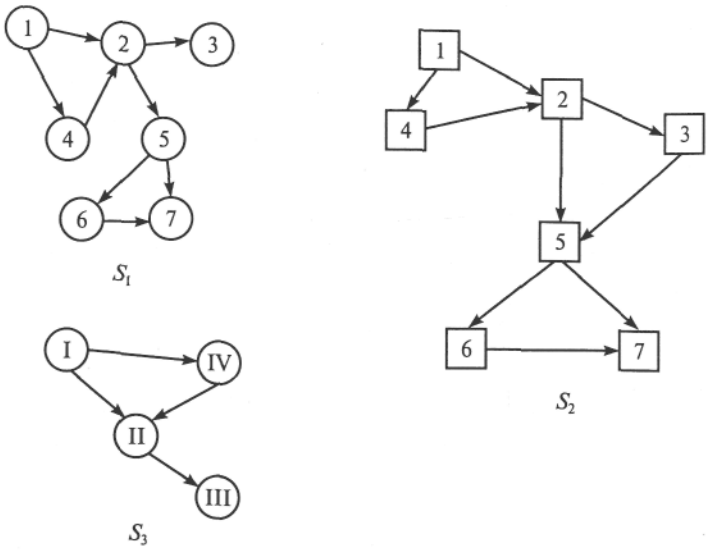


Рис. 8.3. Пример изоморфного и гомоморфного отображений:
S_1 — система-оригинал; S_2 — изоморфное отображение оригинала; S_3 — гомоморфное отображение оригинала

Имитационные модели являются наиболее общими ММ. В силу этого иногда все модели называют имитационными [55]:

- аналитические модели, «имитирующие» только физические законы, на которых основано функционирование реальной системы, можно рассматривать как имитационные модели I уровня;
- статистические модели, в которых, кроме того, «имитируются» случайные факторы, можно называть имитационными моделями II уровня;
- собственно имитационные модели, в которых еще имитируется и функционирование системы во

времени, называют имитационными моделями III уровня.

Классификацию ММ можно провести и по другим признакам [53].

На рис. 8.4 представлена классификация моделей (прежде всего аналитических и статистических) по зависимости переменных и параметров от времени. Динамические модели, в которых учитывается изменение времени, делятся на стационарные (в которых от времени зависят только входные и выходные характеристики) и нестационарные (в которых от времени могут зависеть либо параметры модели, либо ее структура, либо и то, и другое). На рис. 8.5 показана классификация ММ еще по трем основаниям: по характеру изменения переменных; особенностям используемого математического аппарата; способу учета проявления случайностей. Названия типов (видов) моделей в каждом классе достаточно понятны. Укажем лишь, что в сигнально-стохастических моделях случайными являются только внешние воздействия на систему. Имитационные модели, как правило, можно отнести к типам:

- по характеру изменения переменных — к дискретно-непрерывным моделям;
- математическому аппарату — к моделям смешанного типа;
- способу учета случайности — к стохастическим моделям общего вида.

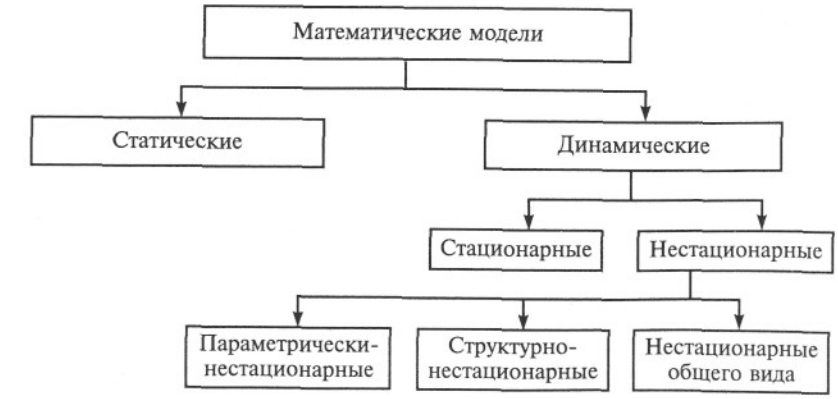


Рис. 8.4. Классификация математических моделей по зависимости переменных и параметров от времени

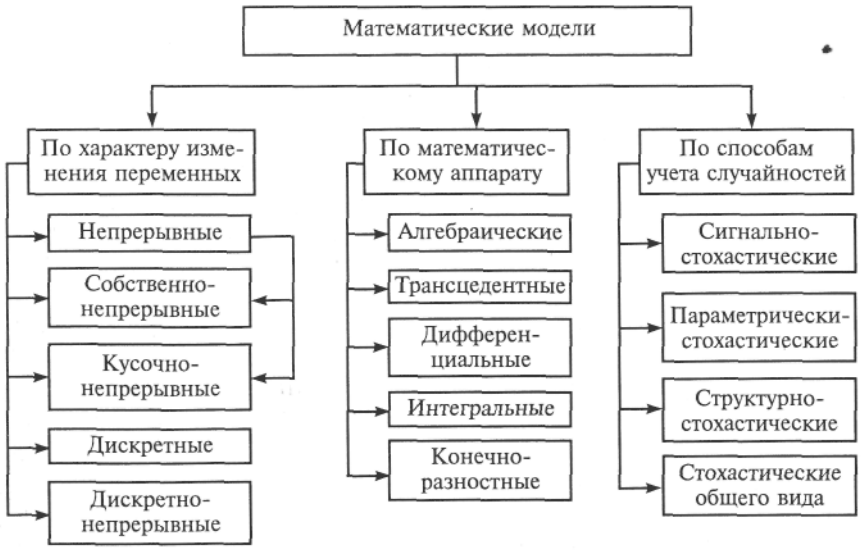


Рис. 8.5. Классификация математических моделей

Глава 9. ИМИТАЦИОННЫЕ МОДЕЛИ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

9.1. Методологические основы применения метода имитационного моделирования

В предыдущем разделе было дано формальное определение имитационной модели с точки зрения свойств двух ее важнейших операторов: оператора моделирования исхода и оператора оценивания показателя эффективности. Приведем классическое вербальное определение имитационного

моделирования и проведем его краткий анализ

По Р.Шеннону *(Robert E.Shannon* — профессор университета в Хантсвилле, штат Алабама, США), *«имитационное моделирование* — есть процесс конструирования на ЭВМ модели сложной реальной системы, функционирующей во времени, и постановки экспериментов на этой модели с целью либо понять поведение системы, либо оценить различные стратегии, обеспечивающие функционирование данной системы» [6].

Выделим в этом определении ряд важнейших обстоятельств, учитывая особенности применения метода для исследования ЭИС.

Во-первых, имитационное моделирование предполагает два этапа: конструирование модели на ЭВМ и проведение экспериментов с этой моделью. Каждый из этих этапов предусматривает использование собственных методов. Так, на первом этапе весьма важно грамотно провести информационное обследование, разработку всех видов документации и их реализацию. Второй этап должен предполагать использование методов планирования эксперимента с учетом особенностей машинной имитации.

Во-вторых, в полном соответствии с системными принципами четко выделены две возможные цели имитационных экспериментов:

- либо понять поведение исследуемой системы (о которой по каким-либо причинам было «мало» информации) — потребность в этом часто возникает, например, при создании принципиально новых образцов продукции;
- либо оценить возможные стратегии управления системой, что также очень характерно для решения широкого круга экономико-прикладных задач.

В-третьих, с помощью имитационного моделирования исследуют сложные системы. Понятие «сложность» является субъективным и по сути выражает отношение исследователя к объекту моделирования. Укажем пять признаков «сложности» системы, по которым можно судить о ее принадлежности к такому классу систем:

- наличие большого количества взаимосвязанных и взаимодействующих элементов;
- сложность функции (функций), выполняемой системой;
- возможность разбиения системы на подсистемы (декомпозиции);
- наличие управления (часто имеющего иерархическую структуру), разветвленной информационной сети и интенсивных потоков информации;
- наличие взаимодействия с внешней средой и функционирование в условиях воздействия случайных (неопределенных) факторов.

Очевидно, что некоторые приведенные признаки сами предполагают субъективные суждения. Вместе с тем становится понятным, почему значительное число ЭИС относят к сложным системам и, следовательно, применяют метод имитационного моделирования. Отметим, что последний признак определяет потребность развития методов учета случайных факторов (см. гл. 10), т. е. проведения так называемой *стохастической имитации.*

В-четвертых, методом имитационного моделирования исследуют системы, функционирующие во времени, что определяет необходимость создания и использования специальных методов (механизмов) управления системным временем.

Наконец, в-пятых, в определении прямо указывается на необходимость использования ЭВМ для реализации имитационных моделей, т.е. проведения машинного эксперимента (машинной имитации), причем в подавляющем большинстве случаев применяются цифровые машины.

Даже столь краткий анализ позволяет сформулировать вывод о целесообразности (а следовательно, и необходимости) использования метода имитационного моделирования для исследования сложных человеко-машинных (эргатических) систем экономического назначения. Особо выделим наиболее характерные обстоятельства применения имитационных моделей [60]:

- если идет процесс познания объекта моделирования;
- если аналитические методы исследования имеются, но составляющие их математические процедуры очень сложны и трудоемки;
- если необходимо осуществить наблюдение за поведением компонентов системы в течение определенного времени;
- если необходимо контролировать протекание процессов в системе путем замедления или ускорения явлений в ходе имитации;
- если особое значение имеет последовательность событий в проектируемых системах и модель

используется для предсказания так называемых «узких» мест;
- при подготовке специалистов для приобретения необходимых навыков в эксплуатации новой техники;
- если имитационное моделирование оказывается единственным способом исследований из-за невозможности проведения реальных экспериментов.

До настоящего момента особое внимание в толковании термина «имитационное моделирование системы» было уделено первому слову. Однако не следует упускать из виду, что создание любой (в том числе и имитационной) модели предполагает, что она будет отражать лишь наиболее существенные с точки зрения конкретной решаемой задачи свойства объекта-оригинала. Английский аналог этого термина — *systems simulation* — при дословном переводе непосредственно указывает на необходимость воспроизводства (симуляции) лишь основных черт реального явления (сравните с термином «симуляция симптомов болезни» из медицинской практики). Важно отметить еще один аспект: создание любой (в том числе и имитационной) модели есть процесс творческий и каждый автор имеет право на собственную версию модели реальной системы. Однако за достаточно длительное время применения метода накоплены определенный опыт и признанные разумными рекомендации, которыми целесообразно руководствоваться при организации имитационных экспериментов.

Укажем ряд основных достоинств и недостатков метода имитационного моделирования [6, 60].

Основные достоинства:

- имитационная модель позволяет в принципе описать моделируемый процесс с большей адекватностью, чем другие;
- имитационная модель обладает известной гибкостью варьирования структуры, алгоритмов и параметров системы;
- применение ЭВМ существенно сокращает продолжительность испытаний по сравнению с натурным экспериментом (если он возможен), а также их стоимость.

Основные недостатки:

- решение, полученное на имитационной модели, всегда носит частный характер, так как оно соответствует фиксированным элементам структуры, алгоритмам поведения и значениям параметров системы;
- большие трудозатраты на создание модели и проведение экспериментов, а также обработку их результатов;
- если использование системы предполагает участие людей при проведении машинного эксперимента, на результаты может оказать влияние так называемый *хауторнский эффект* (заключающийся в том, что люди, зная (чувствуя), что за ними наблюдают, могут изменить свое обычное поведение).

Итак, само использование термина «имитационное моделирование» предполагает работу с такими ММ, с помощью которых результат исследуемой операции нельзя заранее вычислить или предсказать, поэтому необходим эксперимент (имитация) на модели при заданных исходных данных. В свою очередь, сущность машинной имитации заключается в реализации численного метода проведения на ЭВМ экспериментов с ММ, описывающими поведение сложной системы в течение заданного или формируемого периода времени [5].

Каждая имитационная модель представляет собой комбинацию шести основных составляющих [5, 53]компонентов, переменных, параметров, функциональных зависимостей, ограничений, целевых функций.

Под *компонентами* понимают составные части, которые при соответствующем объединении образуют систему. Компоненты называют также элементами системы или ее подсистемами. Например, в модели рынка ценных бумаг компонентами могут выступать отделы коммерческого банка (кредитный, операционный и т.д.), ценные бумаги и их виды, доходы, котировка и т.п.

Параметры — это величины, которые исследователь (пользователь модели) может выбирать произвольно, т. е. управлять ими.

В отличие от них *переменные* могут принимать только значения, определяемые видом данной функции. Так, в выражении для плотности вероятности нормально распределенной случайной величины x

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-m_x)^2}{2\sigma_x^2}},$$

где x — переменная; m_x, σ_x — параметры (математическое ожидание и стандартное отклонение соответственно); π, e — константы.

Различают экзогенные (являющиеся для модели входными и порождаемые вне системы) и эндогенные (возникающие в системе в результате воздействия внутренних причин) переменные. Эндогенные переменные иногда называют переменными состояния.

Функциональные зависимости описывают поведение параметров и переменных в пределах компонента или же выражают соотношения между компонентами системы. Эти соотношения могут быть либо детерминированными, либо стохастическими.

Ограничения — устанавливаемые пределы изменения значений переменных или ограничивающие условия их изменения. Они могут вводиться разработчиком (искусственные) или определяться самой системой вследствие присущих ей свойств (естественные).

Целевая функция предназначена для измерения степени достижения системой желаемой (требуемой) цели и вынесения оценочного суждения по результатам моделирования. Эту функцию также называют функцией критерия. По сути, весь машинный эксперимент с имитационной моделью заключается в поиске таких стратегий управления системой, которые удовлетворяли бы одной из трех концепций ее рационального поведения: оптимизации, пригодности или адаптивизации [26]. Если показатель эффективности системы является скалярным, проблем с формированием критерия не возникает и, как правило, решается оптимизационная задача — поиска стратегии, соответствующей максимуму или минимуму показателя. Сложнее дело обстоит, если приходится использовать векторный показатель. В этом случае для вынесения оценочного суждения используются методы принятия решений по векторному показателю в условиях определенности (когда в модели учитываются только детерминированные факторы) или неопределенности (в противном случае).

При реализации имитационной модели, как правило, рассматриваются не все реально осуществляемые функциональные действия системы (ФД), а только те из них, которые являются наиболее существенными для исследуемой операции. Кроме того, реальные ФД аппроксимируются упрощенными действиями ФД', причем степень этих упрощений определяется уровнем детализации учитываемых в модели факторов. Названные обстоятельства порождают ошибки имитации процесса функционирования реальной системы, что, в свою очередь, обусловливает адекватность модели объекту-оригиналу и достоверность получаемых в ходе моделирования результатов.

На рис. 9.1 схематично представлен пример выполнения некоторых ФД в i-м компоненте реальной системы и ФД′ в i-м компоненте ее модели.

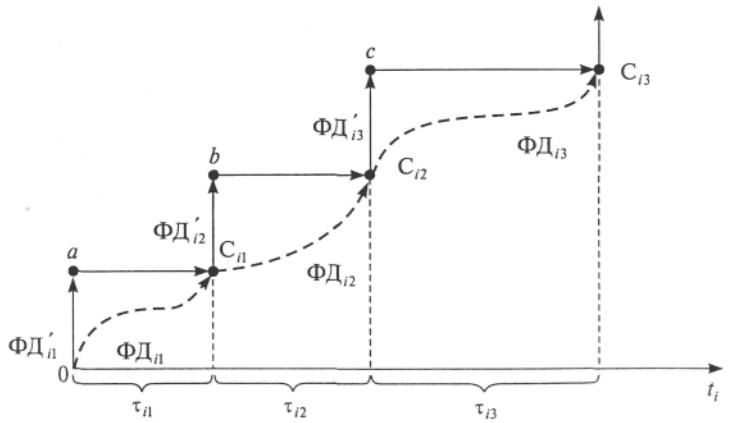


Рис. 9.1. Моделирование функциональных действий в i-м компоненте системы

В i-м компоненте реальной системы последовательно выполняются $ФД_{i1}$, $ФД_{i2}$, $ФД_{i3}$), … за время τ_{i1}, τ_{i2}, τ_{i3}, … соответственно. На рисунке эти действия условно изображены пунктирными («непрямыми») стрелками. В результате ФД наступают соответствующие события: C_{i1} Ci_2, Ci_3, ... В модели последовательность имитации иная: выполняется $ФД'_{i1}$ при неизменном времени, наступает модельное

событие *a,* после чего время сдвигается на величину τ_{i1} инициируя наступление события C_{i1} и т.д. Иными словами, модельной реализации упрощенных ФД (ФД′) соответствует ломаная (0, *a,* C_{i1} *b,* C_{i2}, *d,* C_{i3}, ...). Отметим, что в принципе возможен и другой порядок моделирования: сначала сдвигать время, а затем инициировать наступление соответствующего события.

Очевидно, что в реальной системе в различных ее компонентах могут одновременно (параллельно) производиться функциональные действия и соответственно наступать события. В большинстве же современных ЭВМ в каждый из моментов времени можно отрабатывать лишь один алгоритм какого-либо ФД. Возникает вопрос: каким образом учесть параллельность протекания процессов в реальной системе без потери существенной информации о ней?

Для обеспечения имитации наступления параллельных событий в реальной системе вводят специальную глобальную переменную t_0, которую называют модельным (системным) временем. Именно с помощью этой переменной организуется синхронизация наступления всех событий в модели ЭИС и выполнение алгоритмов функционирования ее компонентов. Принцип такой организации моделирования называется принципом квазипараллелизма [5].

Таким образом, при реализации имитационных моделей используют три представления времени:

- $t_р$ — реальное время системы;
- t_0 — модельное (системное) время;
- $t_м$ — машинное время имитации.

9.2. Классификация имитационных моделей

Имитационные модели принято классифицировать по четырем наиболее распространенным признакам:

- типу используемой ЭВМ;
- способу взаимодействия с пользователем;
- способу управления системным временем (механизму системного времени);
- способу организации квазипараллелизма (схеме формализации моделируемой системы).

Первые два признака позволяют разделить имитационные модели на совершенно понятные (очевидные) классы.

По типу используемой ЭВМ различают аналоговые, цифровые и гибридные имитационные модели. Достоинства и недостатки моделей каждого класса общеизвестны [27]. В дальнейшем будем рассматривать только цифровые модели.

По способу взаимодействия с пользователем имитационные модели могут быть *автоматическими* (не требующими вмешательства исследователя после определения режима моделирования и задания исходных данных) и *интерактивными* (предусматривающими диалог с пользователем в том или ином режиме в соответствии со сценарием моделирования). Отметим, что моделирование сложных систем, относящихся, как уже отмечалось, к классу эргатических систем, как правило, требует применения диалоговых моделей.

Различают два механизма системного времени:

- задание времени с помощью постоянных временных интервалов (шагов);
- задание времени с помощью переменных временных интервалов (моделирование по особым состояниям).

При реализации первого механизма системное время сдвигается на один и тот же интервал (шаг моделирования) независимо от того, какие события должны наступать в системе. При этом наступление всех событий, имевших место на очередном шаге, относят к его окончанию. Рис. 9.2, *а* содержит иллюстрацию данного механизма. Так, для этого механизма считают, что событие A_1 наступило в момент окончания первого шага; событие A_2 — в момент окончания второго шага; события A_3, A_4, A_5 — в момент окончания четвертого шага (эти моменты показаны стрелками) и т.д.

При моделировании по особым состояниям системное время каждый раз изменяется на величину, соответствующую интервалу времени до планируемого момента наступления следующего события, т. е. события обрабатываются поочередно — каждое «в свое время». Если в реальной системе какие-либо события наступают одновременно, это фиксируется в модели. Для реализации этого механизма требуется специальная процедура, в которой отслеживаются времена наступления всех событий и из них выделяется ближайшее по времени. Такую процедуру называют *календарем событий* (см. подразд. 9.3). На рис. 9.2, *б* стрелками обозначены моменты изменения системного времени.

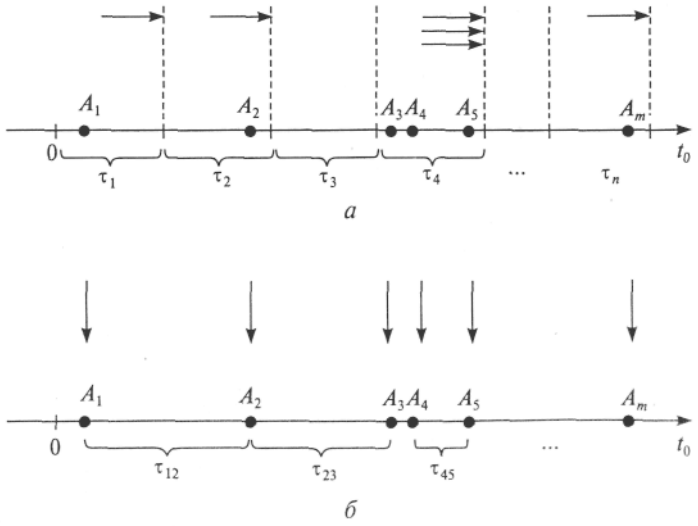


Рис. 9.2. Реализация механизмов системного времени:
а — с постоянным шагом; *б* — с переменным шагом

Существует не столь распространенная разновидность механизма моделирования по особым состояниям, предусматривающая возможность изменения порядка обработки событий, так называемый *механизм моделирования с реверсированием (обращением) шага по времени.* Согласно этому механизму, все события в системе разбиваются на два класса: *фазовые* и *простые.* К первым относят события, порядок моделирования которых нельзя изменять во избежание нарушения причинно-следственных связей в моделируемой системе. Остальные события относят к простым. Таким образом, сначала моделируют очередное фазовое событие, а затем все простые события до этого фазового, причем в произвольном порядке.

На рис. 9.3 приведены перечисленные способы управления системным временем.

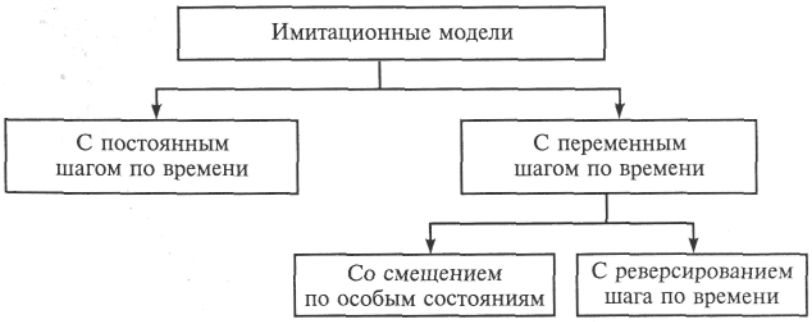


Рис. 9.3. Механизмы управления системным временем

Очевидно, что механизм системного времени с постоянным шагом легко реализуем: достаточно менять временную координату на фиксированный шаг и проверять, какие события уже наступили.

Метод фиксированного шага целесообразно применять в следующих случаях:

- события в системе появляются регулярно;
- число событий велико;
- все события являются для исследователя существенными (или заранее неизвестно, какие из них существенны).

Как уже отмечалось, механизм с переменным шагом по времени требует наличия специального программного средства, способного определять интервал временного сдвига до очередного особого состояния, что осложняет его реализацию.

Вопрос о том, каким механизмом системного времени воспользоваться, решается путем анализа достоинств и недостатков каждого применительно к конкретной модели и требует от разработчика высокой квалификации. В некоторых моделях используют комбинированные механизмы системного

времени в целях исключения перечисленных недостатков.

Важнейшим классификационным признаком имитационных моделей является схема формализации моделируемой системы (способ организации квазипараллелизма).

Наибольшее распространение получили пять способов: просмотр активностей; составление расписания событий; управление обслуживанием транзактов; управление агрегатами; синхронизация процессов.

Характеристика этих способов требует введения ряда понятий [53].

Основными составными частями модели ЭИС являются объекты, которые представляют компоненты реальной системы. Для задания свойств объектов используются *атрибуты (параметры)*. Совокупность объектов с одним и тем же набором атрибутов называют *классом объектов.* Все объекты делят на активные (представляющие в модели те объекты реальной системы, которые способны функционировать самостоятельно и выполнять некоторые действия над другими объектами) и пассивные (представляющие реальные объекты, самостоятельно в рамках данной модели не функционирующие).

Работа (активность) представляется в модели набором операторов, выполняемых в течение некоторого времени и приводящих к изменению состояний объектов системы. В рамках конкретной модели любая работа рассматривается как единый дискретный шаг (возможно, состоящий из других работ). Каждая работа характеризуется временем выполнения и потребляемыми ресурсами.

Событие представляет собой мгновенное изменение состояния некоторого объекта системы (т. е. изменение значений его атрибутов). Окончание любой активности в системе является событием, так как приводит к изменению состояния объекта (объектов), а также может служить инициатором другой работы в системе.

Под *процессом* понимают логически связанный набор активностей, относящихся к одному объекту. Выполнение таких активностей называют *фазой процесса.* Различие между понятиями «активность» и «процесс» полностью определяется степенью детализации модели. Например, смена позиций мобильным объектом в одних моделях может рассматриваться как сложный процесс, а в других — как работа по изменению за некоторое время номера позиции. Процессы, включающие одни и те же типы работ и событий, относят к одному классу. Таким образом, моделируемую систему можно представить соответствующим числом классов процессов. Между двумя последовательными фазами (работами) некоторого процесса может иметь место любое число фаз других процессов, а их чередование в модели, собственно, и выражает суть квазипараллелизма.

В ряде случаев ФД компонентов (объектов) реальной системы одинаковы, а общее их число ограниченно. Каждое ФД можно описать простейшими работами, которые приводят лишь к изменению значений временных координат компонентов системы. Взаимодействие такого рода активностей аналогично функционированию системы массового обслуживания. Однотипные активности объединяются и называются приборами массового обслуживания. Инициаторами появления событий в такой модели становятся заявки (транзакты) на обслуживание этими приборами.

В некоторых реальных системах ФД отдельных компонентов тесно взаимодействуют друг с другом. Компоненты обмениваются между собой сигналами, причем выходной сигнал одного компонента может поступать на вход другой, а сами ФД можно в явном виде описать математическими зависимостями. Если появление выходного сигнала таким образом определяется соответствующим набором «входов», можно реализовать так называемый модульный принцип построения модели. Каждый из модулей строится по стандартной (унифицированной, типовой) структуре и называется агрегатом. С помощью агрегатов (на базе одной из типовых математических схем описания объектов) можно решать весьма широкий круг задач [54].

Вернемся к характеристике способов организации квазипараллелизма.

Способ просмотра активностей применяется при следующих условиях [54]:

- все ФД компонента реальной системы различны, причем для выполнения каждого из них требуется выполнение некоторых (своих) условий;
- условия выполнимости известны исследователю заранее и могут быть заданы алгоритмически;
- в результате ФД в системе наступают различные события;
- связи между ФД отсутствуют, и они осуществляются независимо друг от друга.

В этом случае имитационная модель состоит из двух частей: множества активностей (работ); набора процедур проверки выполнимости условий инициализации активностей, т. е. возможности передачи управления на реализацию алгоритма этой активности.

Проверка выполнимости условия инициализации работы основана либо на анализе значений параметров и/или переменных модели, либо вычислении моментов времени, когда должно осуществляться данное ФД.

После выполнения каждой активности производится модификация системного времени для данного компонента и управление передается в специальный управляющий модуль, что и составляет суть имитации для этого способа организации квазипараллелизма.

Составление расписания событий применяется в тех случаях, когда реальные процессы характеризуются рядом достаточно строгих ограничений [54]:

- различные компоненты выполняют одни и те же ФД;
- начало выполнения этих ФД определяются одними и теми же условиями, причем они известны исследователю и заданы алгоритмически;
- в результате ФД происходят одинаковые события независимо друг от друга;
- связи между ФД отсутствуют, а каждое ФД выполняется независимо.

В таких условиях имитационная модель, по сути, состоит из двух процедур: проверки выполнимости событий; обслуживания (обработки) событий.

Выполнение этих процедур синхронизируется в модельном времени так называемым списковым механизмом планирования. Процедура проверки выполнимости событий схожа с ранее рассмотренными для просмотра активностей (напомним, что окончание любой работы является событием и может инициализировать другую активность) с учетом того, что при выполнении условия происходит не инициализация работы, а обслуживание (розыгрыш) события с последующим изменением системного времени для данного компонента. Корректировка системного времени осуществляется календарем событий, о котором более подробно будет сказано ниже.

Условия применимости транзактного способа организации квазипараллелизма были приведены при определении понятия «транзакт». Связь между приборами массового обслуживания устанавливается с помощью системы очередей, выбранных способов генерации, обслуживания и извлечения транзактов. Так организуется появление транзактов, управление их движением, нахождение в очереди, задержки в обслуживании, уход транзакта из системы и т.п. Событием в такой имитационной модели является момент инициализации любого транзакта. Типовыми структурными элементами модели являются источники транзактов; их поглотители; блоки, имитирующие обслуживание заявок; управляющий модуль. Имитация функционирования реальной системы производится путем выявления очередной (ближайшей по времени) заявки, ее обслуживания, обработки итогов обслуживания (появления нового транзакта; поглощения заявки; изменения возможного времени поступления следующего транзакта и т.п.), изменения системного времени до момента наступления следующего события.

В случае построения *имитационной модели с агрегатным способом организации квазипараллелизма* особое внимание следует уделять оператору перехода системы из одного состояния в другое. Имитация производится за счет передачи управления от агрегата к агрегату при выполнении определенных условий, формирования различных сигналов и их доставки адресату, отработки внешних сигналов, изменения состояния агрегата и т.п. При этом в управляющем модуле осуществляется временная синхронизация состояний всех агрегатов. Отметим, что выделение такого способа реализации квазипараллелизма является достаточно условным, так как квазипараллельная работа агрегатов системы может быть организована другими способами — активностями, планированием событий, взаимодействием транзактов, процессами. Иными словами, агрегатный способ прежде всего ориентирован на использование типовых математических схем (типовых агрегатов) для описания компонентов системы и организации их взаимодействия одним из перечисленных способов.

Процессный способ организации квазипараллелизма применяется в следующих случаях [54]:

- все ФД компонентов реальной системы различны;
- условия инициализации ФД также различны;
- в любой момент времени в данном компоненте может выполняться только одно ФД;
- последовательность ФД в каждом компоненте определена.

Принято считать, что процессный подход объединяет лучшие черты других способов: краткость описания активностей и эффективность событийного представления имитации. Процессным способом можно организовать имитацию ЭИС любой сложности, но такой способ особенно эффективен в тех случаях, когда требуется высокий уровень детализации выполнения ФД, а сама имитационная модель используется для поиска «узких» мест в работе системы. При таком подходе особо важно соблюдение

сходства структуры модели и объекта исследования. Имитационная модель представляет собой набор описаний процессов, каждое из которых описывает один класс процессов, и информационных и управляющих связей между компонентами модели. Каждому компоненту объекта моделирования соответствует свой процесс. Переход от выполнения одной активности к другой активности того же процесса считают изменением его состояния и называют *активизацией процесса.* Проверка выполнимости условий активизации процесса и появление событий осуществляются самим процессом. Процессный способ широко применяется в задачах моделирования проектируемых систем. Он позволяет реализовать многоуровневый модульный подход к моделированию, предусматривающий внесение в модель частичных изменений по результатам исследований, причем значение этого обстоятельства возрастает по мере роста размеров модели [54].

На рис. 9.4 представлена классификация способов организации квазипараллелизма.

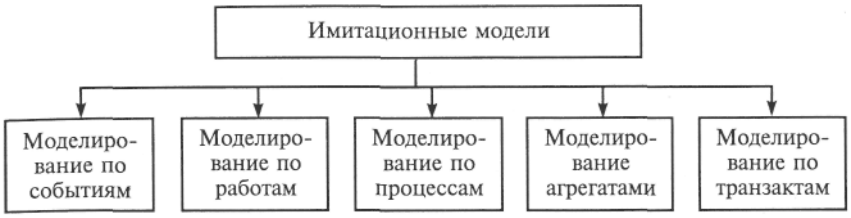


Рис. 9.4. Классификация имитационных моделей по способу организации квазипараллелизма

Отметим, что в настоящее время для реализации всех перечисленных схем формализации моделируемой системы созданы специализированные программные средства, ориентированные на данный способ организации квазипараллелизма, что, с одной стороны, облегчает программную реализацию модели, но, с другой стороны, повышает ответственность исследователя за правильность выбора соответствующей схемы. Подробнее о языках моделирования см. гл. 11.

9.3. Структура типовой имитационной модели с календарем событий

Как уже отмечалось, составление расписания событий как способ организации квазипараллелизма получило широкое распространение в силу прежде всего простоты и наглядности реализации.

На рис. 9.5 представлена структура типовой имитационной модели с календарем событий.

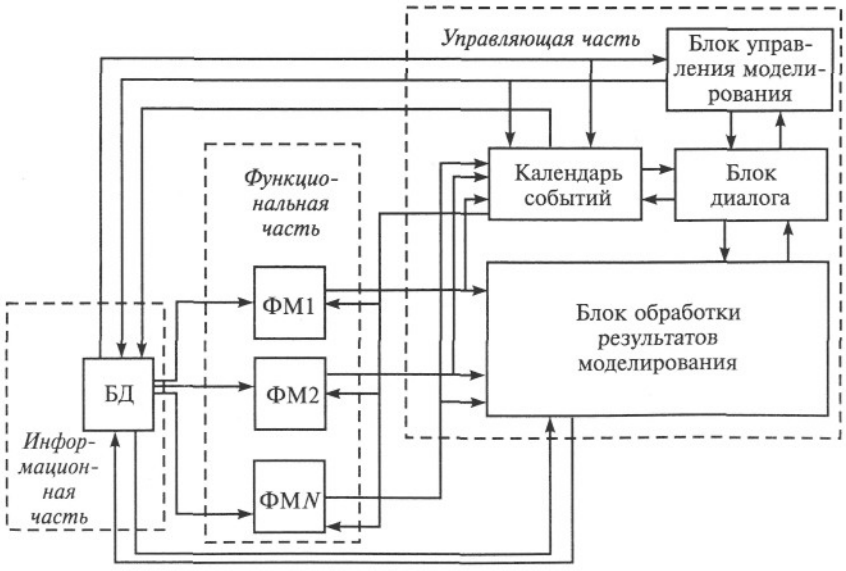


Рис. 9.5. Типовая имитационная модель с календарем событий

Такая имитационная модель состоит из трех частей: управляющей; функциональной, состоящей из функциональных модулей (ФМ); информационной, включающей БД.

В свою очередь, управляющая часть содержит: блок управления моделированием; блок диалога; блок

обработки результатов моделирования; календарь событий.

Блок управления предназначен для реализации принятого плана имитационного эксперимента. В соответствии с назначением в его состав обычно включают управляющий модуль (УМ), определяющий основные временные установки — моменты начала, остановки, продолжения, окончания моделирования, а также моменты изменения режимов моделирования, и модуль реализации плана эксперимента, устанавливающий для каждого прогона модели необходимые значения (уровни) управляемых факторов.

Блок диалога предназначен для обеспечения комфортной работы пользователя с интерактивной моделью (в автоматических моделях этого блока нет). Отметим, что, кроме понятных процедур ввода-вывода информации в требуемых форматах различным потребителям, во многих («больших») имитационных моделях блок диалога включает систему интерактивной многоуровневой помощи пользователю.

В *блоке обработки результатов моделирования* осуществляется обмен информацией с базой данных и реализуются процедуры расчета показателя эффективности (прежде всего за счет статистической обработки результатов моделируемой операции). Если отсутствует блок диалога, на блок обработки возлагаются функции выдачи результатов моделирования на внешние устройства.

Календарь событий является важнейшим элементом имитационной модели, предназначенным для управления процессом появления событий в системе с целью обеспечения необходимой причинно-следственной связи между ними.

Календарем событий решаются следующие основные задачи:

- ранжирование по времени плановых событий, т.е. составление упорядоченной временной последовательности плановых событий с учетом вида возможного события и модуля, в котором оно может наступить (для отработки этой задачи в календаре содержится важнейший элемент — каталог плановых событий, представленный в табл. 9.1);

Таблица 9.1

Каталог плановых событий

Наименование модуля	Вид события	Планируемое время наступления
УМ		
ФМ1		
ФМ2		
ФМ3		
...		
ФМ*N*		

- вызов необходимых функциональных модулей в моменты наступления соответствующих событий;
- получение информационных выходных сигналов от всех функциональных модулей, их хранение и передача в нужные моменты времени адресатам в соответствии с оператором сопряжения модели (эта задача решается с помощью специального программного средства — цепи сигналов и ее основного элемента — таблицы сигналов, табл. 9.2).

Таблица 9.2

Таблица сигналов

№	Адресат	Содержание сигнала	Признак передачи
1			
2			
...			
...			
...			
M			

Перед началом моделирования в первую строку каталога плановых событий (см. табл. 9.1) заносится время инициализации первого прогона модели, а в последнюю — время его окончания, после чего управление передается на тот ФМ, в котором может наступить ближайшее к начальному по времени событие (если на каждом шаге моделирования проводить ранжирование событий по времени, соответствующая этому событию строка каталога будет первой, поскольку для всех уже наступивших (отработанных, обслуженных) событий устанавливается и записывается в третий столбец каталога фиктивное время, заведомо превышающее время окончания моделирования).

Таким образом, если в результате работы очередного ФМ через таблицу сигналов появляется информация о возможном времени наступления в этом или любом другом модуле какого-либо события, это время, а также вид события и модуль, в котором оно может произойти, заносятся в каталог плановых событий, после чего осуществляется новое ранжирование событий по времени. Затем управление передается ФМ (или УМ), информация о котором находится в первой строке каталога до тех пор, пока в первой строке не окажется событие, соответствующее окончанию моделирования.

Подобным же образом организуется работа и таблицы сигналов с учетом того, что в ней содержится информация не о событиях как таковых, а о сигналах различных типов. Так, если в результате работы очередного ФМ возникла необходимость передать какую-либо информацию, соответствующий сигнал (сигналы) помещается в очередную строку таблицы сигналов, после чего осуществляется их передача адресатам. После получения адресатом сигнала в четвертый столбец таблицы заносится установленный признак, и данный сигнал считается отработанным. Понятно, что передача сигналов продолжается до тех пор, пока четвертый столбец таблицы не будет заполнен этим признаком для всех сигналов. Затем управление передается календарю событий, от него — очередному ФМ и т.д.

Функциональная часть имитационной модели состоит из функциональных модулей, являющихся основными ее элементами. Именно в ФМ описываются и реализуются все процессы в моделируемой системе. Обычно один ФМ описывает либо отдельный процесс в системе, либо ее отдельный элемент (подсистему) — в зависимости от выбранной схемы моделирования.

Обобщенная блок-схема работы ФМ представлена на рис. 9.6.

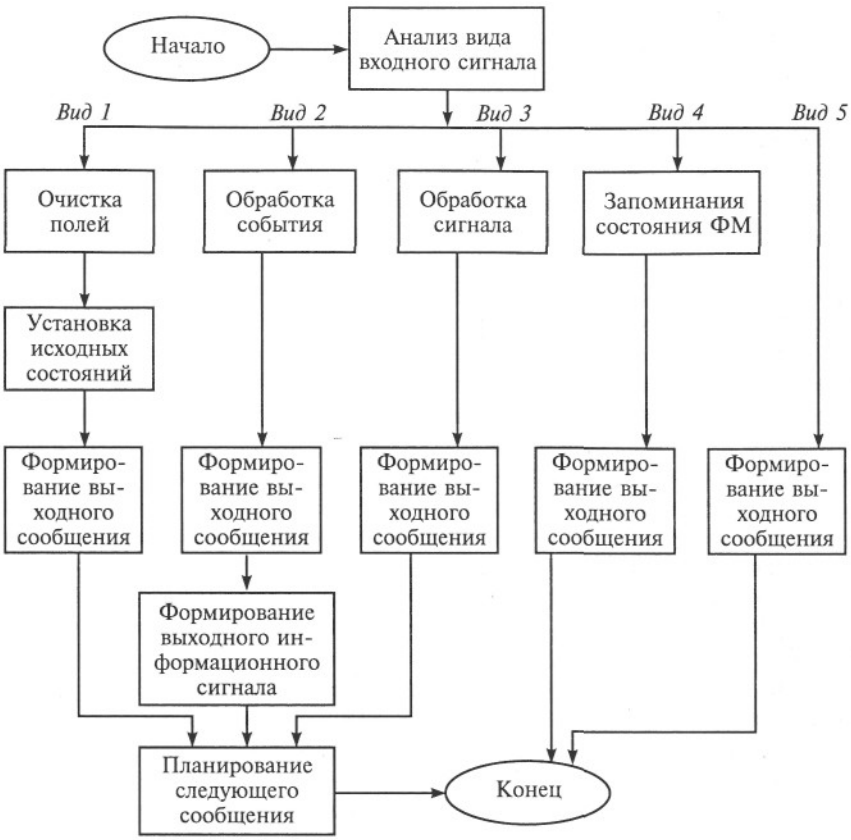


Рис. 9.6. Работа типового ФМ

В ФМ могут поступать пять видов входных сигналов: стартовый сигнал (сигнал о начале

моделирования); сигнал о наступлении планового события; информационный сигнал; сигнал о прерывании моделирования; сигнал об окончании моделирования.

Отметим, что какой бы сигнал ни поступил на вход ФМ, обязательно формируется выходное сообщение о том, что в ФМ данный сигнал отработан, т.е. проведены соответствующие виду входного сигнала действия: подготовка к моделированию (по входному сигналу вида 1); обработка события (по входному сигналу вида 2); обработка информационного сигнала (по входному сигналу вида 3); запоминание состояния ФМ с целью дальнейшего продолжения моделирования с данного «шага» (по входному сигналу вида 4); завершение моделирования в случае выполнения плана имитационного эксперимента (по входному сигналу вида 5). Более подробные сведения об особенностях отработки различных сигналов в имитационных моделях приведены в [50, 55].

Важнейшей задачей любого ФМ является планирование следующих событий, т.е. определение их видов и ожидаемых моментов наступления. Для выполнения этой функции в ФМ реализуется специальный оператор планирования. Для «больших» моделей остро стоит вопрос о «глубине планирования», т.е. о длительности интервала времени, на который прогнозируется наступление событий, поскольку для больших интервалов почти наверняка придется осуществлять повторное планирование после прихода очередного информационного сигнала и соответствующего изменения состояния ФМ.

База (базы) данных представляет собой совокупность специальным образом организованных (структурированных) данных о моделируемой системе (операции), а также программных средств работы с этими данными.

Как правило, информация из БД выдается в другие части имитационной модели в автоматическом режиме (в этом смысле можно говорить, что потребителями информации из БД являются пользователи-задачи). Наличие БД в имитационной модели не является обязательным и полностью определяется масштабами модели, объемами необходимой информации и требованиями по оперативности получения результатов моделирования и их достоверности. Если принято решение о включении БД в состав имитационной модели, проектирование БД не имеет каких-либо специфических особенностей и проводится по стандартной методике.

Глава 10. ТЕХНОЛОГИЯ МОДЕЛИРОВАНИЯ СЛУЧАЙНЫХ ФАКТОРОВ

10.1. Генерация псевдослучайных чисел

Как уже отмечалось ранее, имитационное моделирование ЭИС, как правило, предполагает необходимость учета различных случайных факторов — событий, величин, векторов (систем случайных величин), процессов.

В основе всех методов и приемов моделирования названных случайных факторов лежит использование случайных чисел, равномерно распределенных на интервале [0; 1].

До появления ЭВМ в качестве генераторов случайных чисел применяли механические устройства — колесо рулетки, специальные игральные кости и устройства, которые перемешивали фишки с номерами, вытаскиваемые вручную по одной.

По мере роста объемов применения случайных чисел для ускорения их моделирования стали обращаться к помощи электронных устройств. Самым известным из таких устройств был электронный импульсный генератор, управляемый источником шума, разработанный широко известной фирмой *RAND Corporation.* Фирмой в 1955 г. была выпущены книга, содержащая миллион случайных чисел, сформированных этим генератором, а также случайные числа в записи на магнитной ленте. Использовались и другие подобные генераторы — например, основанные на преобразовании естественного случайного шума при радиоактивном распаде. Все эти генераторы обладают двумя недостатками:

- невозможно повторно получить одну и ту же последовательность случайных чисел, что бывает необходимо при экспериментах с имитационной моделью;
- технически сложно реализовать физические генераторы, способные длительное время выдавать случайные числа «требуемого качества».

В принципе можно заранее ввести полученные таким образом случайные числа в память машины и обращаться к ним по мере необходимости, что сопряжено с понятными негативными обстоятельствами — большим (причем неоправданным) расходом ресурсов ЭВМ и затратой времени на обмен данными

между долгосрочной и оперативной памятью (особенно существенно для «больших» имитационных моделей).

В силу этого наибольшее распространение получили другие генераторы, позволяющие получать так называемые псевдослучайные числа (ПСЧ) с помощью детерминированных рекуррентных формул. Псевдослучайными эти числа называют потому, что фактически они, даже пройдя все тесты на случайность и равномерность распределения, остаются полностью детерминированными. Это значит, что если каждый цикл работы генератора начинается с одними и теми же исходными данными, то на выходе получаем одинаковые последовательности чисел. Это свойство генератора обычно называют воспроизводимостью последовательности ПСЧ. Программные генераторы ПСЧ должны удовлетворять следующим требованиям:

- ПСЧ должны быть равномерно распределены на интервале [0; 1] и независимы, т.е. случайные последовательности должны быть некоррелированы;
- цикл генератора должен иметь возможно большую длину;
- последовательность ПСЧ должна быть воспроизводима;
- генератор должен быть быстродействующим;
- генератор должен занимать малый объем памяти.

Первой расчетной процедурой генерации ПСЧ, получившей достаточно широкое распространение, можно считать метод срединных квадратов, предложенный Дж.фон Нейманом и Ф.Метрополисом в 1946 г. Сущность метода заключается в последовательном нахождении квадрата некоторого m-значного числа; выделении из него m средних цифр, образующих новое число, которое и принимается за очередное в последовательности ПСЧ; возведении этого числа в квадрат; выделении из квадрата m средних цифр и далее до получения последовательности требуемой длины. Как следует из описания процедуры метода, он весьма прост в вычислительном отношении и, следовательно, легко реализуем программно. Однако ему присущ очень серьезный недостаток — обусловленность статистических свойств генерируемой последовательности выбором ее корня (начального значения), причем эта обусловленность не является «регулярной», т.е. трудно определить заранее, можно ли использовать полученные данным методом ПСЧ при проведении исследований. Это обстоятельство иллюстрируется рис. 10.1, на котором представлены результаты генерации последовательности из ста ПСЧ при следующих исходных данных: число знаков m = 4; корни последовательности X_0 = 2152; X_0 = 2153; X_0 = 3789; X_0 = 3500.

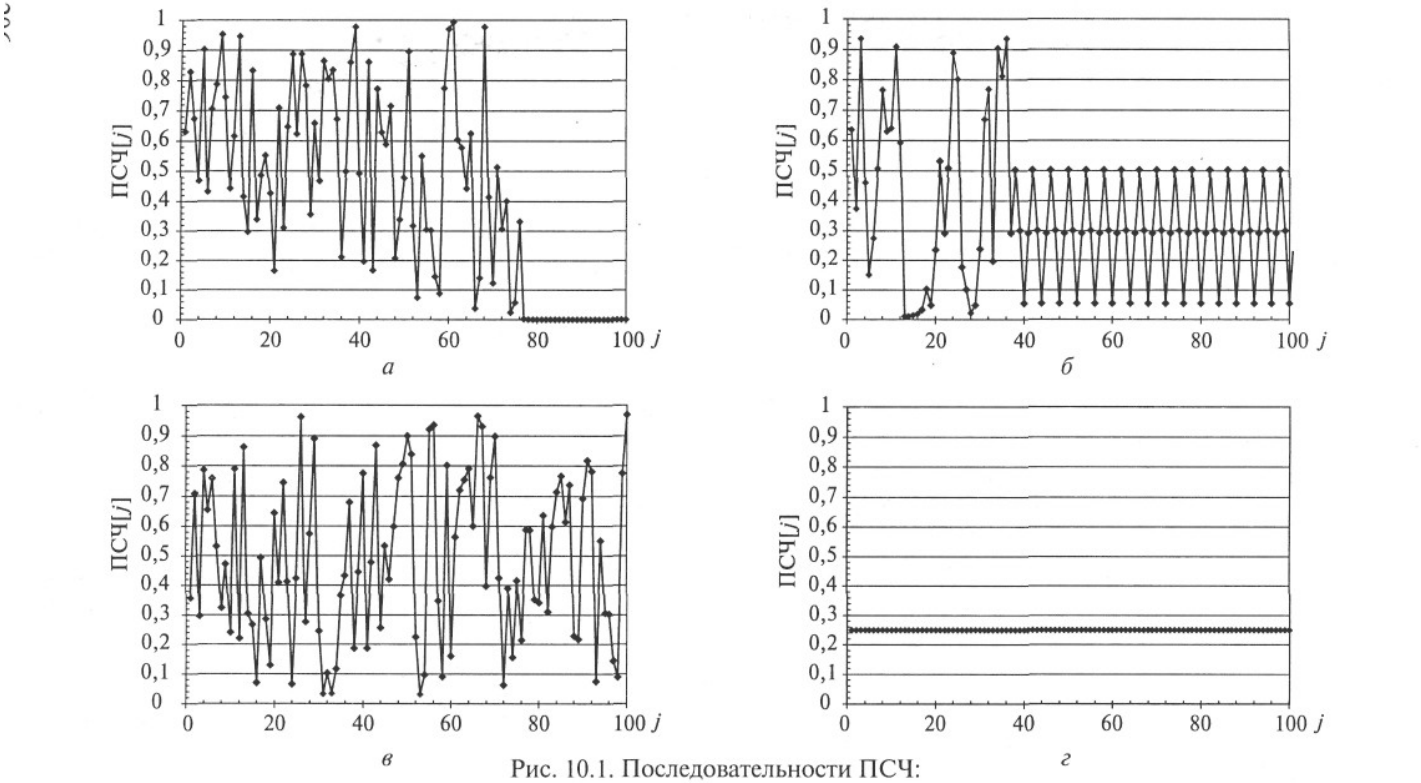


Рис. 10.1. Последовательности ПСЧ:

а — корень последовательности X_0 = 2152; *б* — корень последовательности X_0 = 2153; *в* — корень последовательности X_0 = 3789; *г* — корень последовательности X_0 = 3500

Из анализа рисунка видно, что при X_0 = 2152 уже с 78-го члена последовательности все ПСЧ принимают нулевые значения; при X_0 = 2153, начиная с 36-го значения, последовательность перестает быть случайной; при X_0 = 3789 первые 100 членов последовательности можно использовать в качестве ПСЧ (дальнейшее поведение последовательности ПСЧ требует дополнительных исследований); при X_0 = 3500 (2500; 4500 и т.д.) нулевые значения принимают все ПСЧ. Иными словами, метод срединных квадратов не позволяет по начальному значению оценить качество последовательности ПСЧ, в частности ее период.

Мультипликативный метод. Основная формула мультипликативного генератора для расчета значения очередного ПСЧ по значению предыдущего имеет вид

$$X_{i+1} = aX_i(\mathrm{mod}m),$$

где *a, m* — неотрицательные целые числа (их называют множитель и модуль).

Как следует из формулы, для генерации последовательности ПСЧ необходимо задать начальное значение (корень) последовательности, множитель и модуль, причем период (длина) последовательности *P* зависит от разрядности ЭВМ и выбранного модуля, а статистические свойства — от выбранного начального значения и множителя. Таким образом, следует выбирать перечисленные величины так, чтобы по возможности максимизировать длину последовательности и минимизировать корреляцию между генерируемыми ПСЧ. В специальной литературе приводятся рекомендации по выбору значений параметров метода, использование которых обеспечивает (гарантирует) получение определенного количества ПСЧ с требуемыми статистическими свойствами (отметим, что данное замечание можно отнести ко всем конгруэнтным методам). Так, если для машины с двоичной системой счисления задать $m = 2^b$, $a = 8T \pm 3V$, где b — число двоичных цифр (бит) в машинном слове; T — любое целое положительное число; V — любое положительное нечетное число, получим последовательность ПСЧ с периодом, равным $p = 2^{b-2} = m/4$. Заметим, что в принципе возможно за счет другого выбора модуля m увеличить длину последовательности до $P = m - 1$, частично пожертвовав скоростью вычислений [48]. Кроме того, важно, что получаемые таким образом ПСЧ оказываются нормированными, т.е. распределенными от 0 до 1.

На рис. 10.2 приведены последовательности ПСЧ, полученные по мультипликативному методу со следующими параметрами: X_0 = 15; T = 3; V = 1; b = 4 и b = 6 (столь малые значения b объясняются стремлением проиллюстрировать работоспособность рекомендованных формул). Очевидно, что в первом случае длина последовательности до повторений равна 4, а во втором — 16 ПСЧ. Легко показать, что от выбора корня последовательности ее длина не зависит (при равенстве остальных параметров).

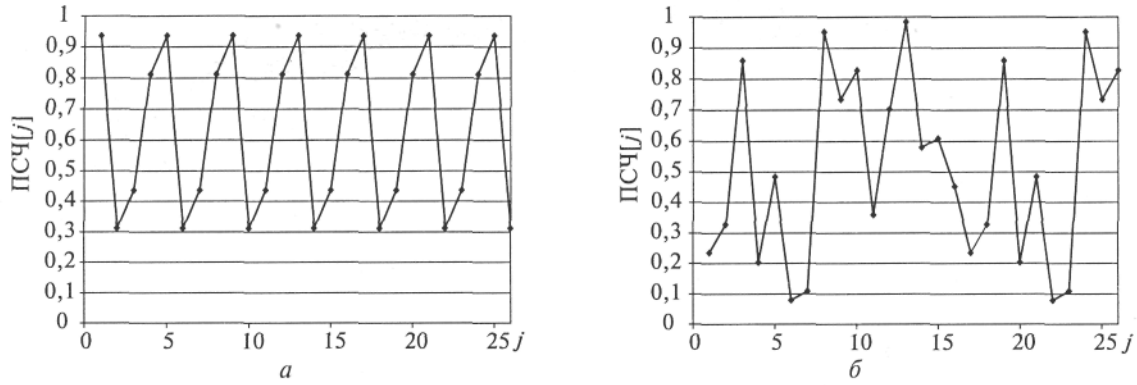


Рис. 10.2. Последовательности ПСЧ, полученные по мультипликативному методу:
а — число двоичных цифр в машинном слове $b = 4$; *б* — число двоичных цифр в машинном слове $b = 6$

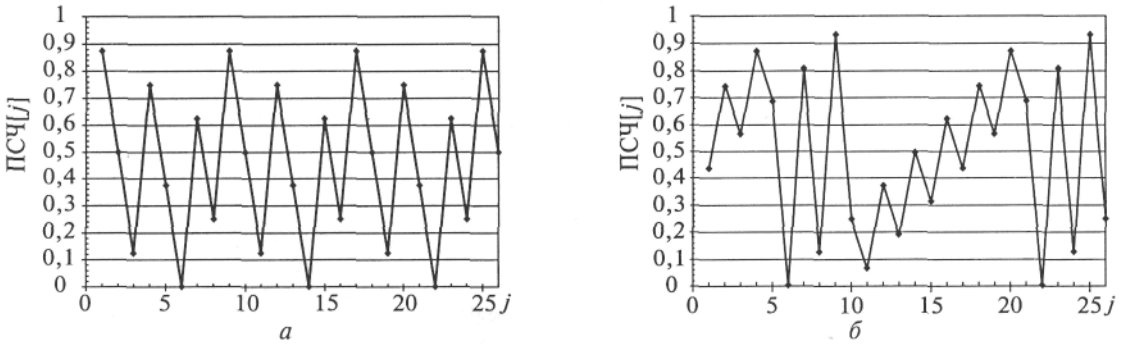


Рис. 10.3. Последовательности ПСЧ, полученные по смешанному методу при параметрах:
а — число двоичных цифр в машинном слове $b = 3$; *б* — число двоичных цифр в машинном слове $b = 4$

Аддитивный метод. Основная формула для генерации ПСЧ по аддитивному методу имеет вид

$$X_{i+1} = (X_i + X_{i-1}) (\mathrm{mod} m),$$

где m — целое число.

Очевидно, что для инициализации генератора, построенного по этому методу, необходимо, помимо модуля m, задать два исходных члена последовательности. При $X_0 = 0$; $X_1 = 1$ последовательность превращается в ряд Фибоначчи. Рекомендации по выбору модуля совпадают с предыдущим случаем; длину последовательности можно оценить по приближенной формуле

$$p = 2^{b+1} - 2(b - 1).$$

На рис. 10.3 приведены две последовательности ПСЧ, полученные при исходных данных: $b = 3$; $X_0 = 1$; $X_1 = 3$ и $b = 4$; $X_0 = 5$; $X_1 = 7$. В обоих случаях период ПСЧ равен 12.

Смешанный метод. Данный метод несколько расширяет возможности мультипликативного генератора за счет введения так называемого коэффициента сдвига c. Формула метода имеет вид

$$X_{i+1} = (aX_i + c) (\mathrm{mod} m).$$

За счет выбора параметров генератора можно обеспечить максимальный период последовательности ПСЧ $p = 2^b$. На рис. 10.4 показаны две последовательности ПСЧ, полученные при следующих исходных данных: $X_0 = 7$; $c = 13$; $a = 9$; $b = 3$ и $b = 4$. В первом случае длина последовательности равна 8, а во втором — 16 ПСЧ.

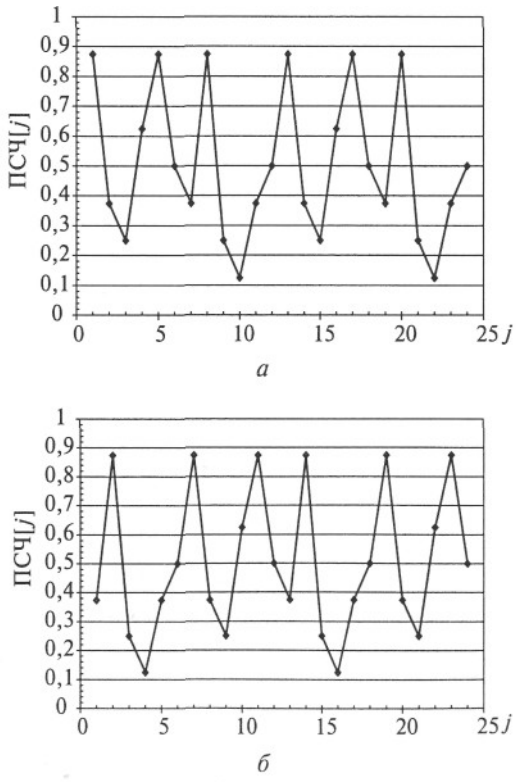


Рис. 10.4. Последовательности ПСЧ, полученные по аддитивному методу при параметрах:
а — число двоичных цифр в машинном слове $b = 3$; *б* — число двоичных цифр в машинном слове $b = 4$

Разработано множество модификаций перечисленных конгруэнтных методов, обладающих определенными преимуществами при решении конкретных практических задач, а также рекомендаций по выбору того или иного метода [48]. Для весьма широкого круга задач вполне удовлетворительными оказываются типовые генераторы ПСЧ, разработанные, как правило, на основе смешанного метода и входящие в состав стандартного общего программного обеспечения большинства ЭВМ. Специальным образом генерацию ПСЧ организуют либо для особо масштабных имитационных исследований, либо при повышенных требованиях к точности имитации реального процесса (объекта).

Подводя итог вышеизложенному, подчеркнем, что разработка конгруэнтных методов зачастую осуществляется на основе эвристического подхода, основанного на опыте и интуиции исследователя. После модификации известного метода тщательно проверяют, обладают ли генерируемые в соответствии с новой формулой последовательности ПСЧ требуемыми статистическими свойствами, и в случае положительного ответа формулируют рекомендации по условиям ее применения.

10.2. Моделирование случайных событий

В теории вероятностей реализацию некоторого комплекса условий называют *испытанием.* Результат испытания, регистрируемый как факт, называют *событием.*

Случайным называют *событие,* которое в результате испытания может наступить, а может и не наступить (в отличие от *достоверного события,* которое при реализации данного комплекса наступает всегда, и *невозможного события,* которое при реализации данного комплекса условий не наступает никогда). Исчерпывающей характеристикой случайного события является вероятность его наступления. Примерами случайных событий являются отказы в экономических системах; объемы выпускаемой продукции каждым предприятием в каждый день; котировки валют в обменных пунктах; состояние рынка ценных бумаг и биржевого дела и т. п.

Моделирование случайного события заключается в определении («розыгрыше») факта его наступления.

Для моделирования случайного события A, наступающего в опыте с вероятностью P_A, достаточно одного случайного (псевдослучайного) числа R, равномерно распределенного на интервале [0; 1]. В случае попадания ПСЧ R в интервал [0; P_A] событие A считают наступившим в данном опыте; в

противном случае — не наступившим в данном опыте. На рис. 10.5 показаны оба исхода: при ПСЧ R_1 событие следует считать наступившим; при ПСЧ R_2 — событие в данном испытании не наступило.

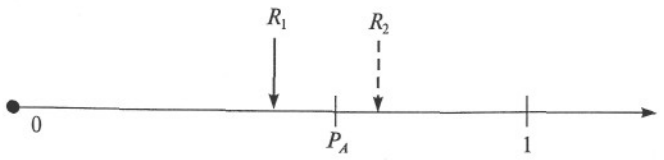


Рис. 10.5. Моделирование случайных событий

Очевидно, что чем больше вероятность наступления моделируемого события, тем чаще ПСЧ, равномерно распределенные на интервале [0; 1], будут попадать в интервал [0; P_A], что и означает факт наступления события в испытании.

Для моделирования одного из полной группы N случайных несовместных событий $A1, A2, ..., AN$ с вероятностями наступления $\{P_{A1}, P_{A2}, ..., P_{AN}\}$ соответственно также достаточно одного ПСЧ R.

Для таких случайных событий можно записать

$$\sum_{i=1}^{N} P_{Ai} = 1.$$

Факт наступления одного из событий группы определяют, исходя из условия принадлежности ПСЧ R тому или иному интервалу, на который разбивают интервал [0; 1]. Так, на рис. 10.6 для ПСЧ R_1 считают, что наступило событие $A2$. Если ПСЧ оказалось равным R_2, считают, что наступило событие $A(N - 1)$.

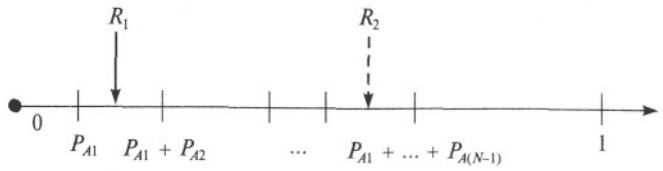


Рис. 10.6. Моделирование полной группы несовместных событий

Если группа событий не является полной, вводят дополнительное (фиктивное) событие $A(N + 1)$, вероятность которого определяют по формуле

$$P_{A(N+1)} = 1 - \sum_{i=1}^{N} P_A$$

Далее действуют по уже изложенному алгоритму для полной группы событий с одним изменением: если ПСЧ попадает в последний $(N + 1)$-й интервал, считают, что ни одно из N событий, составляющих неполную группу, не наступило.

В практике имитационных исследований часто возникает необходимость моделирования *зависимых событий,* для которых вероятность наступления одного события оказывается зависящей от того, наступило или не наступило другое событие. В качестве одного из примеров зависимых событий приведем доставку груза потребителю в двух случаях: когда маршрут движения известен и был поставщиком дополнительно уточнен и когда уточнения движения груза не проводилось. Понятно, что вероятность доставки груза от поставщика к потребителю для приведенных случаев будет различной.

Для того чтобы провести моделирование двух зависимых случайных событий A и B, необходимо задать следующие *полные* и *условные вероятности:*

$$P(A);\ P(B);\ P(B/A);\ P(B/\overline{A}).$$

Заметим, что, если вероятность наступления события B при условии, что событие A не наступило, не задана, ее можно определить по формуле

$$P(B/\bar{A}) = \frac{P(B) - P(A)P(B/A)}{1 - P(A)}.$$

Существуют два алгоритма моделирования зависимых событий. Один из них условно можно назвать «последовательным моделированием»; другой — «моделированием после предварительных расчетов».

Последовательное моделирование. Алгоритм последовательного моделирования представлен на рис. 10.7.

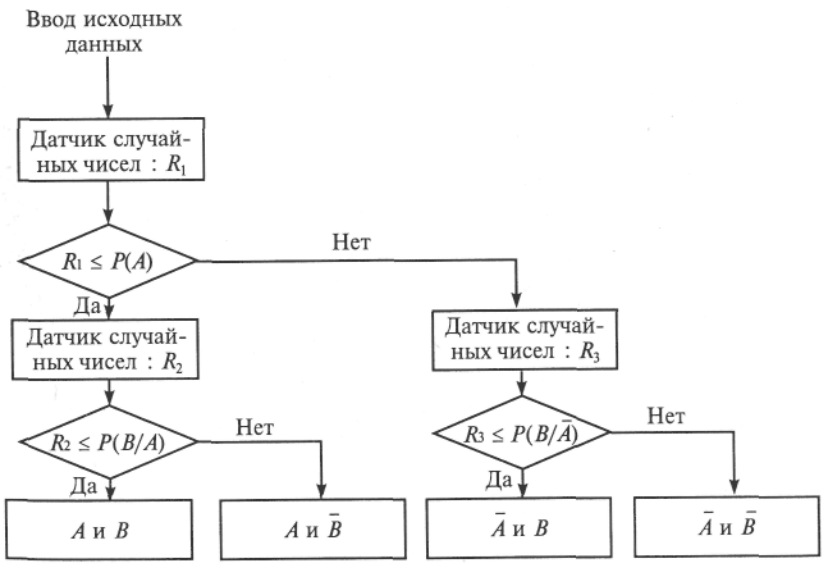


Рис. 10.7. Последовательное моделирование зависимых событий

Несомненными достоинствами данного алгоритма являются его простота и естественность, поскольку зависимые события «разыгрываются» последовательно — так, как они наступают (или не наступают) в реальной жизни, что и является характерной особенностью большинства имитационных моделей. Вместе с тем алгоритм предусматривает троекратное обращение к датчику случайных чисел, что увеличивает время моделирования.

Моделирование после предварительных расчетов. Как легко заметить, приведенные на рис. 10.7 четыре исхода моделирования зависимых событий образуют полную группу несовместных событий. На этом основан алгоритм моделирования, предусматривающий *предварительный расчет вероятностей* каждого из исходов и «розыгрыш» факта наступления одного из них, как для любой группы несовместных событий. Рис. 10.8 иллюстрирует разбивка интервала [0; 1] на четыре отрезка, длины которых соответствуют вероятностям исходов наступления событий.

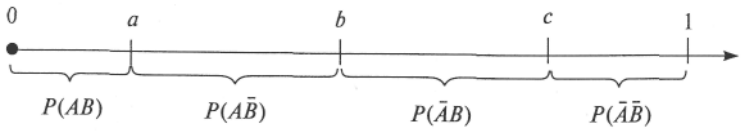


Рис. 10.8. Разбивка интервала [0; 1] для реализации алгоритма моделирования зависимых событий после предварительных расчетов

На рис. 10.9 представлен алгоритм моделирования. Данный алгоритм предусматривает одно обращение к датчику случайных чисел, что обеспечивает выигрыш во времени имитации по сравнению с последовательным моделированием, однако перед началом работы алгоритма исследователь должен рассчитать и ввести вероятности реализации всех возможных исходов (естественно, эту несложную процедуру можно также оформить программно, но это несколько удлинит алгоритм).

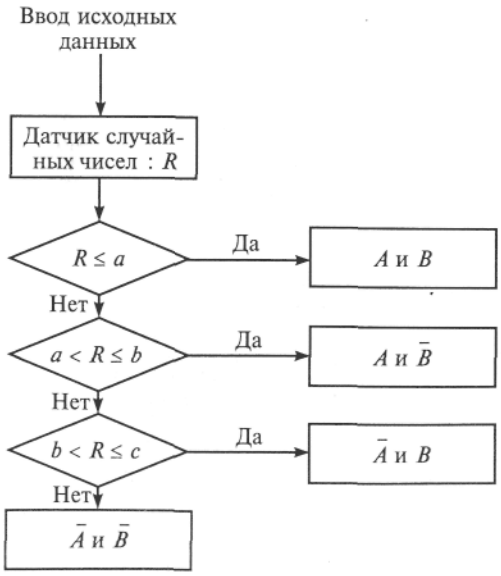


Рис. 10.9. Алгоритм моделирования зависимых случайных событий после предварительных расчетов

10.3. Моделирование случайных величин

В практике создания и использования имитационных моделей весьма часто приходится сталкиваться с необходимостью моделирования важнейшего класса факторов — случайных величин (СВ) различных типов.

Случайной называют переменную величину, которая в результате испытания принимает то или иное значение, причем заранее неизвестно, какое именно. При этом под испытанием понимают реализацию некоторого (вполне определенного) комплекса условий.

В зависимости от множества возможных значений различают три типа СВ: непрерывные, дискретные, смешанного типа.

Исчерпывающей характеристикой любой СВ является ее *закон распределения,* который может быть задан в различных формах: *функции распределения* — для всех типов СВ; *плотности вероятности* (распределения) — для непрерывных СВ; *таблицы* или *ряда распределения* — для дискретных СВ.

В данном подразделе изложены основные методы моделирования СВ первых двух типов как наиболее часто встречающихся на практике.

Моделирование непрерывных случайных величин. Моделирование СВ заключается в определении («розыгрыше») в нужный по ходу имитации момент времени конкретного значения СВ в соответствии с требуемым (заданным) законом распределения.

Наибольшее распространение получили три метода: метод обратной функции, метод исключения (фон Неймана), метод композиций.

Метод обратной функции. Метод позволяет при моделировании СВ учесть все ее статистические свойства и основан на следующей теореме.

Если непрерывная СВ Y имеет плотность вероятности $f(y)$, то СВ X, определяемая преобразованием

$$x = \int_{-\infty}^{y} f(y)\mathrm{d}y = F(y),$$

имеет равномерный закон распределения на интервале [0; 1].

Данную теорему поясняет рис. 10.10, на котором изображена функция распределения СВ Y. Теорему доказывает цепочка рассуждений, основанная на определении понятия «функция распределения» и условии теоремы

$$F(x) = P(X < x) = P(Y < y) = F(y) = \int_{-\infty}^{y} f(y)\mathrm{d}y = x.$$

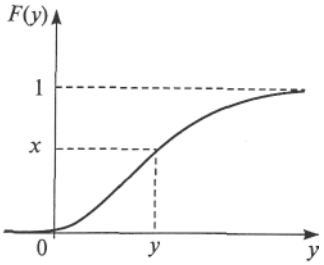


Рис. 10.10. Функция распределения СВ *Y*

Таким образом, получили равенство

$$F(x) = x,$$

а это и означает, что СВ *X* распределена равномерно в интервале [0; 1].

Напомним, что в общем виде функция распределения равномерно распределенной на интервале [*a; b*] СВ *X* имеет вид

$$F(x) = \begin{cases} 0, & x < a; \\ \dfrac{x-a}{b-a}, & a \le x \le b; \\ 1, & x > b. \end{cases}$$

Теперь можно попытаться найти обратное преобразование функции распределения $F^{(-1)}(x)$.

Если такое преобразование существует (условием этого является наличие первой производной у функции распределения), алгоритм метода включает всего два шага:

- моделирование ПСЧ, равномерно распределенного на интервале [0; 1];
- подстановка этого ПСЧ в обратную функцию и вычисление значения СВ *Y*

$$x = F(y) \Rightarrow y = F^{(-1)}(x).$$

При необходимости эти два шага повторяются столько раз, сколько возможных значений СВ *Y* требуется получить.

Пример. Длина свободного пробега нейтрона в однородном веществе d ($d > 0$) имеет следующее распределение [18, 35]:

$$F(d) = 1 - e^{-\sigma_d d},$$

где σ_d — среднее квадратическое отклонение длины пробега.

Тогда формула для генерации возможного значения СВ *D* имеет вид

$$d = -\frac{\ln(1-R)}{\sigma_d},$$

где *R* — ПСЧ, распределенное равномерно в интервале [0; 1].

Простота метода обратной функции позволяет сформулировать такой вывод: если обратное преобразование функции распределения СВ, возможные значения которой необходимо получить, существует, следует использовать именно этот метод. К сожалению, круг СВ с функциями распределения, допускающими обратное преобразование, не столь широк, что потребовало разработки иных методов.

Метод исключения (фон Неймана). Метод фон Неймана позволяет из совокупности равномерно

распределенных ПСЧ R_i, по определенным правилам выбрать совокупность значений y_i с требуемой функцией распределения $f(y)$ [18, 35].

Алгоритм метода следующий.

1. Выполняется усечение исходного распределения таким образом, чтобы область возможность значений СВ Y совпадала с интервалом $[a; b]$.

В результате формируется плотность вероятности $f^*(y)$ такая, что

$$\int_a^b f^*(y)\mathrm{d}y = 1.$$

Длина интервала $[a; b]$ определяется требуемой точностью моделирования значений СВ в рамках конкретного исследования.

2. Генерируется пара ПСЧ R_1 и R_2, равномерно распределенных на интервале [0; 1].

3. Вычисляется пара ПСЧ R_1^* и R_2 по формулам

$$R_1^* = a + (b - a)R_1;\ R_2^* = MR_2,$$

где

$$M = \max_{y\in[a,b]} f^*(y).$$

На координатной плоскости пара чисел (R_1^*; R_2^*) определяет точку — например, точку Q_1 на рис. 10.11.

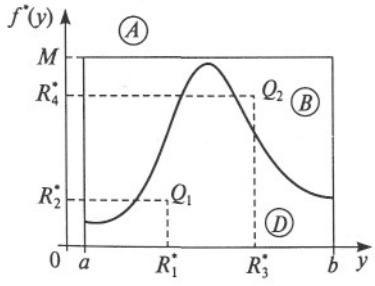


Рис. 10.11. Моделирование СВ методом фон Неймана:

A — прямоугольник, ограничивающий график плотности распределения моделируемой СВ; B — область прямоугольника A, находящаяся выше графика $f^*(y)$; D — область прямоугольника A, находящаяся ниже графика $f^*(y)$

4. Если точка (Q_1) принадлежит области D, считают, что получено первое требуемое значение СВ $y_1 = R_1^*$.

5. Генерируется следующая пара ПСЧ R_3 и R_4, равномерно распределенных на интервале [0; 1], после пересчета по п. 3 задающих на координатной плоскости вторую точку — Q_2.

6. Если точка (Q_2) принадлежит области B, переходят к моделированию следующей пары ПСЧ $(R_5; R_6)$ и далее до получения необходимого количества ПСЧ.

Очевидно, что в ряде случаев (при попадании изображающих точек в область B соответствующие ПСЧ с нечетными индексами не могут быть включены в требуемую выборку возможных значений моделируемой СВ, причем это будет происходить тем чаще, чем сильнее график $f^*(y)$ по форме будет «отличаться» от прямоугольника A. Оценить среднее относительное число q «пустых» обращений к генератору ПСЧ можно геометрическим методом, вычислив отношение площадей соответствующих областей (B и A):

$$S_A = M(b-a) = S_B + S_D = S_B + 1;$$
$$S_B = S_A - 1;$$
$$q = \frac{S_B}{S_A} = 1 - \frac{1}{S_A} = 1 - \frac{1}{M(b-a)}.$$

На рис. 10.12 [27] показаны две функции плотности вероятности, вписанные в прямоугольники A и B соответственно. Первая функция соответствует β-распределению с параметрами $\eta = \lambda = 2$. Вторая функция соответствует γ-распределению с параметрами $\lambda = 0{,}5$; $\sigma = 1$.

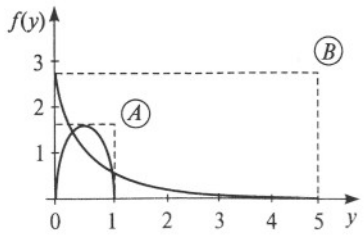


Рис. 10.12. Иллюстрация результатов оценки эффективности метода фон Неймана

Для первой функции $q \approx 0{,}33$; для второй — $q \approx 0{,}92$. Таким образом, для β-распределения метод фон Неймана почти в три раза эффективнее, чем для γ-распределения. В целом для многих законов распределения (особенно для островершинных и имеющих длинные «хвосты») метод исключения приводит к большим затратам машинного времени на генерацию требуемого количества ПСЧ.

Главным достоинством метода фон Неймана является его универсальность — применимость для генерации СВ, имеющих любую вычислимую или заданную таблично плотность вероятности.

Метод композиции. Применение метода основано на теоремах теории вероятностей, доказывающих представимость одной СВ композицией двух или более СВ, имеющих относительно простые, более легко реализуемые законы распределения. Наиболее часто данным методом пользуются для генерации ПСЧ, имеющих нормальное распределение. Согласно центральной предельной теореме, распределение СВ *Y,* задаваемой преобразованием

$$y = \frac{1}{\sqrt{\frac{k}{12}}}\left(\sum_{i=1}^{k} R_i - \frac{k}{2}\right),$$

где R_i — равномерно распределенные на интервале [0; 1] ПСЧ, при росте k неограниченно приближается к нормальному распределению со стандартными параметрами $(m_y = 0; \sigma = 1)$.

Последнее обстоятельство легко подтверждается следующим образом. Введем СВ Z и найдем параметры ее распределения, используя соответствующие теоремы о математическом ожидании и дисперсии суммы СВ:

$$Z = \sum_{i=1}^{k} R_i;$$

$$M[Z] = m_z = M\left[\sum_{i=1}^{k} m_r\right] = \sum_{i=1}^{k} m_r = \sum_{i=1}^{k} \frac{1}{2} = \frac{k}{2};$$

$$\sigma_z = \sqrt{D_z} = \sqrt{\sum_{i=1}^{k} D_r} = \sqrt{\sum_{i=1}^{k} \frac{1}{12}} = \sqrt{\frac{k}{12}},$$

где m — математическое ожидание; r — значение случайных чисел.

При равномерном распределении в интервале [0; 1] СВ имеет параметры: $m_r = {}^1/_2$; $D_r = {}^1/_{12}$.

Очевидно, что

$$Y = \frac{Z - m_z}{\sigma_z},$$

и, как любая центрированно-нормированная СВ, имеет стандартные параметры.

Как правило, берут $k = 12$ и считают, что для подавляющего числа практических задач обеспечивается должная точность вычислений. Если же к точности имитации предъявляются особые требования, можно улучшить качество моделирования СВ за счет введения нелинейной поправки [8]:

$$y_{\text{мод}} = y(k) + \frac{1}{20k}[y(k)^3 - 3y(k)],$$

где *y (k)* — возможное значение СВ *Y*, полученное в результате сложения, центрирования и нормирования *k* равномерно распределенных ПСЧ R_i.

Еще одним распространенным вариантом применения метода композиции является моделирование возможных значений СВ, обладающей χ^2 распределением с *n* степенями свободы: для этого нужно сложить «квадраты» *n* независимых нормально распределенных СВ со стандартными параметрами.

Возможные значения СВ, подчиненной закону распределения Симпсона (широко применяемого, например, в радиоэлектронике), моделируют, используя основную формулу метода при *k = 2.* Существуют и другие приложения этого метода.

В целом можно сделать вывод о том, что метод композиции применим и дает хорошие результаты тогда, когда из теории вероятностей известно, композиция каких легко моделируемых СВ позволяет получить СВ с требуемым законом распределения.

Моделирование дискретных случайных величин. Дискретные СВ (ДСВ) достаточно часто используются при моделировании систем. Основными методами генерации возможных значений ДСВ являются: метод последовательных сравнений, метод интерпретации.

Метод последовательных сравнений. Алгоритм метода практически совпадает с ранее рассмотренным алгоритмом моделирования полной группы несовместных случайных событий, если считать номер события номером возможного значения ДСВ, а вероятность наступления события — вероятностью принятия ДСВ этого возможного значения. На рис. 10.13 показана схема определения номера возможного значения ДСВ, полученного на очередном шаге.

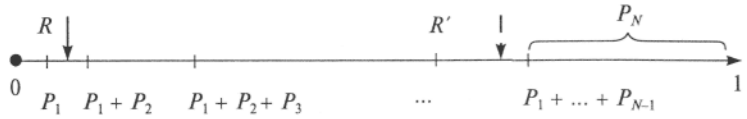


Рис. 10.13. Моделирование ДСВ методом последовательных сравнений

Из анализа ситуации, показанной на рис. 10.13, для ПСЧ *R,* «попавшего» в интервал $[P_1; P_1 + P_2]$, следует сделать вывод, что ДСВ приняла свое второе возможное значение; а для ПСЧ *R′* — что ДСВ приняла свое *(N* - 1)-е значение и т.д. Алгоритм последовательных сравнений можно улучшить (ускорить) за счет применения методов оптимизации перебора — дихотомии (метода половинного деления); перебора с предварительным ранжированием вероятностей возможных значений по убыванию и т. п.

Метод интерпретации. Метод основан на использовании модельных аналогий с сущностью физических явлений, описываемых моделируемыми законами распределения.

На практике метод чаще всего используют для моделирования биномиального закона распределения, описывающего число успехов в *n* независимых опытах с вероятностью успеха в каждом испытании *p* и вероятностью неудачи *q* = 1 - *p.* Алгоритм метода для этого случая весьма прост:

- моделируют *n* равномерно распределенных на интервале [0; 1] ПСЧ;
- подсчитывают число *m* тех из ПСЧ, которые меньше *p;*
- это число *m* считают возможным значением моделируемой ДСВ, подчиненной биномиальному закону распределения.

Помимо перечисленных, существуют и другие методы моделирования ДСВ, основанные на специальных свойствах моделируемых распределений или на связи между распределениями [7].

10.4. Моделирование случайных векторов

Случайным вектором (системой случайных величин) называют совокупность случайных величин, совместно характеризующих какое-либо случайное явление

$$X = (X_1, X_2, ..., X_n),$$

где X_i — СВ с теми или иными законами распределения.

Данный подпункт содержит материал по методам моделирования непрерывных случайных векторов (все компоненты которых представляют собой непрерывные СВ).

Исчерпывающей характеристикой случайного вектора является совместная многомерная функция распределения его компонент $F(x_1, x_2, ..., x_n)$ или соответствующая ему совместная многомерная плотность вероятности

$$f(x_1, x_2, ..., x_n).$$

Проще всего моделировать случайный вектор с независимыми компонентами, для которого справедливо

$$f(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} f_i(x_i),$$

т.е. каждую из компонент случайного вектора можно моделировать независимо от других в соответствии с ее «собственной» плотностью вероятности $f_i(x_i)$.

В случае, когда компоненты случайного вектора статистически зависимы, необходимо использовать специальные методы: условных распределений, исключения (фон Неймана), линейных преобразований.

Метод условных распределений. Метод основан на рекуррентном вычислении условных плотностей вероятностей для каждой из компонент случайного вектора x с многомерной совместной плотностью вероятности $f(x_1, x_2, ..., x_n)$.

Для плотности распределения случайного вектора x можно записать:

$$f(x_1, x_2, ..., x_n) = f_1(x_1) f_2(x_2/x_1) ... f_n(x_n/x_{n-1}, x_{n-2}, ..., x_1),$$

где $f_1(x_1)$ — плотность распределения СВ X_1, $f_n(x_n/x_{n-1}, x_{n-2}, ..., x_1)$ — плотность условного распределения СВ X_n при условии: $X_1 = x_1; X_2 = x_2; ... ; X_{n-1} = x_{n-1}$.

Для получения указанных плотностей необходимо провести интегрирование совместной плотности распределения случайного вектора в соответствующих пределах:

$$\begin{cases} f_1(x_1) = \int ... \int_{S_1} f(x_1, x_2, ..., x_n) dx_2 ... dx_n; \\ f_2(x_2/x_1) = \int ... \int_{S_2} \dfrac{f(x_1, ..., x_n)}{f_1(x_1)} dx_3 ... dx_n; \\ ... \\ f_n(x_n/x_{n-1}, ..., x_1) = \int ... \int_{S_n} \dfrac{f(x_1, ..., x_n)}{f_n(x_n)} dx_{n+1}. \end{cases}$$

Порядок моделирования:

- моделировать значение x_1^* СВ X_1 по закону $f_1(x_1)$

$$S_i = \bigvee_{j=i+1}^{n} Q_j;\ X_j \in Q_j;\ i = 1, 2, ..., n-1,$$

где Q_j — множество случайных величин x_j,

- моделировать значение x_2^* СВ X_2 по закону $f_2(x_2/x_1^*)$;
- ...;
- моделировать значение x_n^* СВ X_n по закону $f_n(x_n/x^*_{n-1}, x^*_{n-2}, ..., x_1^*)$.

Тогда вектор $(x_1^*, x_2^*, ..., x_n^*)$ и есть реализация искомого случайного вектора X.

Метод условных распределений (как и метод обратной функции для скалярной СВ) позволяет учесть

все статистические свойства случайного вектора. Поэтому справедлив вывод: если имеется возможность получить условные плотности распределения $f_n(x_n/x_{n-1}, x_{n-2}, ..., x_1)$, следует пользоваться именно этим методом.

Метод исключения (фон Неймана). Метод является обобщением уже рассмотренного для СВ метода фон Неймана на случай n переменных. Предполагается, что все компоненты случайного вектора распределены в конечных интервалах $x_i \in [a_i, b_i]$, $i = 1, 2, ..., n$. Если это не так, необходимо произвести усечение плотности распределения для выполнения данного условия.

Алгоритм метода следующий.

1. Генерируются $(n + 1)$ ПСЧ

$$R_1,\ R_2,\ ...,\ R_n;\ R_f,$$

распределенных соответственно на интервалах

$$[a_1,\ b_1],\ [a_2,\ b_2],\ ...,\ [a_n,\ b_n];\ [0,\ f_{\max}];$$

$$f_{\max} = \max ... \max_{x_i \in [a_i,\ b_i];\ i=1,\ 2,\ ...,\ n} f\ (x_1,\ x_2,\ ...,\ x_n).$$

2. Если выполняется условие

$$R_f \le f(R_1,\ R_2,\ ...,\ R_n),$$

то вектор

$$(R_1,\ R_2,\ ...,\ R_n)$$

и есть искомая реализация случайного вектора.

3. Если данное условие не выполняется, переходят к первому пункту и т.д.

Рис. 10.14 содержит иллюстрацию данного алгоритма для двумерного случая $R_f \le f(R_1, R_2)$.

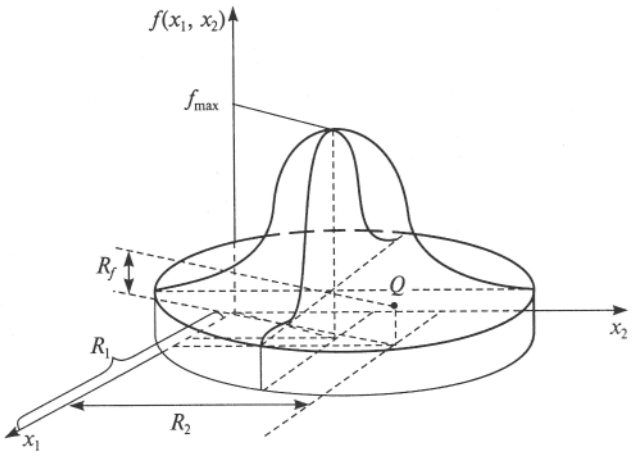


Рис. 10.14. Моделирование двумерного случайного вектора методом фон Неймана

Возврат к п. 1 после «неудачного» моделирования n ПСЧ происходит тогда, когда точка Q окажется выше поверхности, представляющей двумерную плотность вероятности $f(x_1, x_2)$. Для случая, представленного на рисунке, в качестве (очередной) реализации двумерного случайного вектора следует взять пару ПСЧ (R_1, R_2).

Среднюю относительную частоту «неудач» можно вычислить геометрическим способом, взяв отношение объемов соответствующих фигур.

Как уже отмечалось для одномерного случая, основным достоинством метода фон Неймана является его универсальность. Однако для плотностей вероятностей, поверхности которых имеют острые пики, достаточно часто будут встречаться «пустые» прогоны, когда очередные n ПСЧ бракуются. Этот

недостаток тем существеннее, чем больше размерность моделируемого вектора *(n)* и длиннее требуемая выборка реализаций случайного вектора. На практике такие ситуации встречаются не слишком часто, поэтому метод исключений и имеет столь широкое распространение.

Метод линейных преобразований. Метод линейных преобразований является одним из наиболее распространенных так называемых корреляционных методов, применяемых в случаях, когда при моделировании непрерывного *n*-мерного случайного вектора достаточно обеспечить лишь требуемые значения элементов корреляционной матрицы этого вектора (это особенно важно для случая нормального распределения, для которого выполнение названного требования означает выполнение достаточного условия полного статистического соответствия теоретического и моделируемого распределений). Идея метода заключается в линейном преобразовании случайного *n*-мерного вектора *Y* с независимыми (чаще всего нормально распределенными) компонентами в случайный вектор *X* с требуемыми корреляционной матрицей и вектором математических ожиданий компонент.

Математическая постановка задачи выглядит следующим образом. Даны корреляционная матрица и математическое ожидание вектора *X*

$$Q = \|q_{ij}\| = \|M[(X_i - m_{x_i})(X_j - m_{x_j})]\|;$$
$$M = (m_{x_1}, m_{x_2}, \ldots, m_{x_n})^T.$$

Требуется найти такую матрицу *B,* которая позволяла бы в результате преобразования

$$X = BY + M, \qquad (3.1)$$

где *Y* — *n*-мерный вектор с независимыми нормально распределенными компонентами со стандартными параметрами, получить вектор *X* с требуемыми характеристиками.

Будем искать матрицу *B* в виде нижней треугольной матрицы, все элементы которой, расположенные выше главной диагонали, равны 0. Перейдем от матричной записи к системе алгебраических уравнений:

$$\begin{vmatrix} x_1 \\ x_2 \\ \ldots \\ \ldots \\ X_n \end{vmatrix} = \begin{vmatrix} b_{11} \\ b_{12} b_{22} \\ \ldots \\ \ldots \\ b_{n1} b_{n2} b_{n3} \ldots b_{nn} \end{vmatrix} \times \begin{vmatrix} Y_1 \\ Y_2 \\ \ldots \\ \ldots \\ Y_n \end{vmatrix} + \begin{vmatrix} m_{x_1} \\ m_{x_2} \\ \ldots \\ \ldots \\ m_{x_n} \end{vmatrix} \Rightarrow$$

$$\begin{cases} X_1 - m_{x_1} = b_{11} Y_1; \\ X_2 - m_{x_2} = b_{21} Y_1 + b_{22} Y_2; \\ \ldots \\ \ldots \\ X_n - m_{x_n} = b_{n1} Y_1 + b_{n2} Y_2 + \ldots + b_{nn} Y_n. \end{cases} \qquad (3.2)$$

Поскольку компоненты вектора *y* независимы и имеют стандартные параметры, справедливо выражение

$$M[Y_i Y_j] = \begin{cases} 0, i \neq j; \\ 1, i = j. \end{cases}$$

Почленно перемножив сами на себя и между собой соответственно левые и правые части уравнений системы (3.2) и взяв от результатов перемножения математическое ожидание, получим систему уравнений вида

$$\begin{cases} M[(X_1 - m_{x_1})(X_1 - m_{x_1})] = M[b_{11}Y_1 b_{11}Y_1]; \\ M[(X_1 - m_{x_1})(X_2 - m_{x_2})] = M[(b_{21}Y_1 + b_{22}Y_2)b_{11}Y_1]; \\ \ldots \end{cases}$$

Как легко увидеть, в левых частях полученной системы уравнений — элементы заданной корреляционной матрицы *Q,* а в правых — элементы искомой матрицы *B.* Последовательно решая эту систему, получаем формулы для расчета элементов b_{ij}:

$$b_{11} = \sqrt{q_{11}};\ b_{21} = q_{12} / \sqrt{q_{11}};\ b_{22} = \sqrt{q_{22} - \frac{q_{12}}{q_{11}}},\ \ldots.$$

Формула для расчета любого элемента матрицы преобразования *B* имеет вид

$$b_{ij} = \frac{q_{ij} - \sum_{k=1}^{j-1} b_{ik} b_{jk}}{\sqrt{q_{ij} - \sum_{k=1}^{j-1} b_{jk}^2}};\ 1 \le j \le i \le n.$$

Таким образом, алгоритм метода линейных преобразований весьма прост:

- по заданной корреляционной матрице рассчитывают значения коэффициентов матрицы преобразования *B;*
- генерируют одну реализацию вектора *Y,* компоненты которого независимы и распределены нормально со стандартными параметрами;
- полученный вектор подставляют в выражение (3.1) и определяют очередную реализацию вектора *X,* имеющего заданные корреляционную матрицу и вектор математических ожиданий компонентов;
- при необходимости два предыдущих шага алгоритма повторяют требуемое число раз (до получения нужного количества реализаций вектора *X).*

В данной главе рассмотрены основные методы генерации ПСЧ, равномерно распределенных на интервале [0; 1], и моделирования случайных событий, величин и векторов, часто используемые в практике имитационных исследований ЭИС. Как правило, все современные программные средства, применяемые для реализации тех или иных имитационных моделей, содержат встроенные генераторы равномерно распределенных ПСЧ, что позволяет исследователю легко моделировать любые случайные факторы.

Глава 11. ОСНОВЫ ОРГАНИЗАЦИИ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

11.1. Этапы имитационного моделирования

Как уже отмечалось, имитационное моделирование применяют для исследования сложных экономических систем. Естественно, что и имитационные модели оказываются достаточно сложными как с точки зрения заложенного в них математического аппарата, так и в плане машинной реализации. При этом сложность любой модели определяется двумя факторами: сложностью исследуемого объекта-оригинала; точностью, предъявляемой к результатам расчетов.

Использование машинного эксперимента как средства решения сложных прикладных проблем, несмотря на присущую каждой конкретной задаче специфику, имеет ряд общих черт (этапов). На рис. 11.1 представлены этапы применения математической (имитационной) модели (по взглядам академика А. А. Самарского).

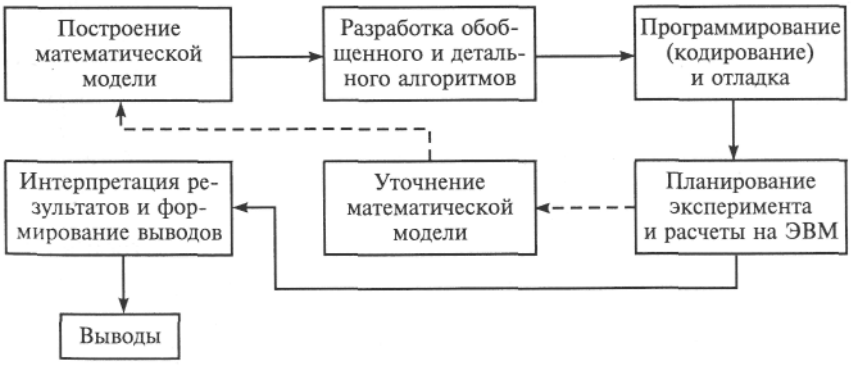


Рис. 11.1. Этапы машинного эксперимента:
——→ прямая связь; - - - → обратная связь

Каждому из показанных на рисунке этапов присущи собственные приемы, методы, технологии. В данном учебнике вопросы построения (разработки) математической модели, алгоритмизации и программирования (за исключением выбора языка) не рассматриваются. Отметим лишь, что все эти этапы носят ярко выраженный творческий характер и требуют от разработчика модели особой подготовки.

После того как имитационная модель реализована на ЭВМ, исследователь должен выполнить последовательность следующих этапов (их часто называют технологическими) [29, 55]:

- испытание модели;
- исследование свойств модели;
- планирование имитационного эксперимента;
- эксплуатация модели (проведение расчетов).

Кратко охарактеризуем первые два этапа (изложение методов математической теории планирования эксперимента и организации проведения модельных расчетов и обработки их результатов выходят за рамки настоящего учебника).

Испытание имитационной модели. Включает работы по четырем направлениям:

- задание исходной информации;
- верификацию имитационной модели;
- проверку адекватности модели;
- калибровку имитационной модели.

Задание исходной информации. Процедура задания исходной информации полностью определяется типом моделируемой системы:

- если моделируется функционирующая (существующая) система, проводят измерение характеристик ее функционирования и затем используют эти данные в качестве исходных при моделировании;
- если моделируется проектируемая система, проводят измерения на прототипах;
- если прототипов нет, используют экспертные оценки параметров и переменных модели, формализующих характеристики реальной системы.

Каждому из этих вариантов присущи собственные особенности и сложности. Так, проведение измерений на существующих и проектируемых системах требует применения качественных измерительных средств, а проведение экспертного оценивания исходных данных представляет собой комплекс достаточно сложных процедур получения, обработки и интерпретации экспертной информации.

Верификация имитационной модели. Она состоит в доказательстве утверждений соответствия алгоритма ее функционирования цели моделирования путем формальных и неформальных исследований реализованной программы модели.

Неформальные исследования представляют собой ряд процедур, входящих в автономную и комплексную отладку.

Формальные методы включают:

- использование специальных процессоров-«читателей» программ;
- замену стохастических элементов модели детерминированными;
- тест на так называемую непрерывность моделирования и др.

Проверка адекватности модели. Количественную оценку адекватности модели объекту исследования проводят для случая, когда можно определить значения отклика системы в ходе натурных испытаний.

Наиболее распространены три способа проверки:

- по средним значениям откликов модели и системы;
- дисперсиям отклонений откликов;
- максимальному значению абсолютных отклонений откликов.

Если возможность измерения отклика реальной системы отсутствует, оценку адекватности модели проводят на основе субъективного суждения соответствующего должностного лица о возможности использования результатов, полученных с использованием этой модели, при выполнении им служебных обязанностей (в частности, при обосновании решений — подробнее см. [53]).

Калибровка имитационной модели. К калибровке имитационной модели приступают в случае, когда модель оказывается неадекватной реальной системе. За счет калибровки иногда удается уменьшить неточности описания отдельных подсистем (элементов) реальной системы и тем самым повысить достоверность получаемых модельных результатов.

В модели при калибровке возможны изменения трех типов:

- глобальные структурные изменения;
- локальные структурные изменения;
- изменение так называемых калибровочных параметров в результате реализации достаточно сложной итерационной процедуры, включающей многократное построение регрессионных зависимостей и статистическую оценку значимости улучшения модели на очередном шаге.

При необходимости проведения некоторых локальных и особенно глобальных структурных изменений приходится возвращаться к содержательному описанию моделируемой системы и искать дополнительную информацию о ней.

Исследование свойств имитационной модели. После испытаний имитационной модели переходят к изучению ее свойств. При этом наиболее важны четыре процедуры:

- оценка погрешности имитации;
- определение длительности переходного режима в имитационной модели;
- оценка устойчивости результатов имитации;
- исследование чувствительности имитационной модели.

Оценка погрешности имитации, связанной с использованием в модели генераторов ПСЧ. Исследование качества генераторов ПСЧ проводится известными методами теории вероятностей и математической статистики (см. подразд. 10.1). Важнейшим показателем качества любого генератора ПСЧ является период последовательности ПСЧ (при требуемых статистических свойствах). В большинстве случаев о качестве генератора ПСЧ судят по оценкам математических ожиданий и дисперсий отклонений компонент функции отклика. Как уже отмечалось, для подавляющего числа практических задач стандартные (встроенные) генераторы дают вполне пригодные последовательности ПСЧ.

Определение длительности переходного режима. Обычно имитационные модели применяются для изучения системы в типичных для нее и повторяющихся условиях. В большинстве стохастических моделей требуется некоторое время T_0 для достижения моделью установившегося состояния.

Под *статистическим равновесием* или установившимся состоянием модели понимают такое состояние, в котором противодействующие влияния сбалансированы и компенсируют друг друга. Иными словами: модель находится в равновесии, если ее отклик не выходит за предельные значения.

Существует три способа уменьшения влияния начального периода на динамику моделирования сложной системы:

- использование «длинных прогонов», позволяющих получать результаты после заведомого выхода модели на установившийся режим;
- исключение из рассмотрения начального периода прогона;
- выбор таких начальных условий, которые ближе всего к типичным.

Каждый из этих способов не свободен от недостатков: «длинные прогоны» приводят к большим затратам машинного времени; при исключении из рассмотрения начального периода теряется часть информации; выбор типичных начальных условий, обеспечивающих быструю сходимость, как правило, затруднен отсутствием достаточного объема исходных данных (особенно для принципиально новых

систем).

Для отделения переходного режима от стационарного у исследователя должна быть возможность наблюдения за моментом входа контролируемого параметра в стационарный режим. Часто используют такой метод: строят графики изменения контролируемого параметра в модельном времени и на нем выявляют переходный режим. На рис. 11.2 представлен график изменения k-го контролируемого параметра модели g_k в зависимости от модельного времени t_0. На рисунке видно, что, начиная со времени $t_{\text{перех}}$, этот параметр «вошел» в установившийся режим со средним значением $\bar{g}_k$.

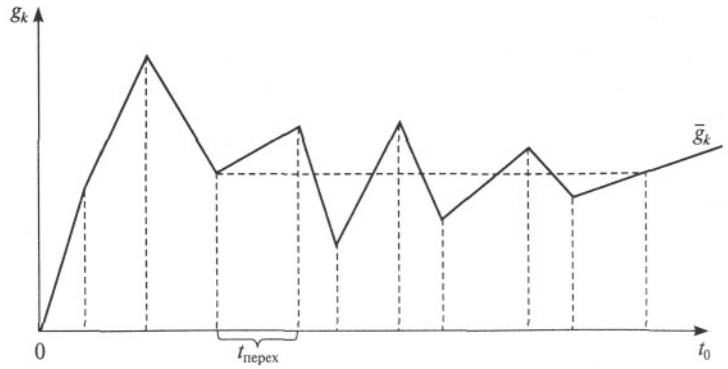


Рис. 11.2. Определение длительности переходного периода для k-го контролируемого параметра модели

Если построить подобные графики для всех (или большинства существенных) контролируемых параметров модели, определить для каждого из них длительность переходного режима и выбрать из них наибольшую, в большинстве случаев можно считать, что после этого времени все интересующие исследователя параметры находятся в установившемся режиме.

На практике встречаются случаи, когда переходные режимы исследуются специально. Понятно, что при этом используют «короткие прогоны», исключают из рассмотрения установившиеся режимы и стремятся найти начальные условия моделирования, приводящие к наибольшей длительности переходных процессов. Иногда для увеличения точности результатов проводят замедление изменения системного времени.

Оценка устойчивости результатов имитации. Под устойчивостью результатов имитации понимают степень их нечувствительности к изменению входных условий. Универсальной процедуры оценки устойчивости нет. Практически часто находят дисперсию отклика модели Y по нескольким компонентам и проверяют, увеличивается ли она с ростом интервала моделирования. Если увеличения дисперсии отклика не наблюдается, результаты имитации считают устойчивыми.

Важная практическая рекомендация: чем ближе структура модели к структуре реальной системы и чем выше степень детализации учитываемых в модели факторов, тем шире область устойчивости (пригодности) результатов имитации.

Исследование чувствительности модели. Работы на этом этапе имеют два направления:

- установление диапазона изменения отклика модели при варьировании каждого параметра;
- проверка зависимости отклика модели от изменения параметров внешней среды.

В зависимости от диапазона изменения откликов Y при изменении каждой компоненты вектора параметров X определяется стратегия планирования экспериментов на модели. Если при значительной амплитуде изменения некоторого компонента вектора параметров модели отклик меняется незначительно, то точность представления ее в модели не играет существенной роли.

Проверка зависимости отклика модели Y от изменений параметров внешней среды основана на расчете соответствующих частных производных и их анализе.

11.2. Языки моделирования

Чтобы реализовать на ЭВМ модель сложной системы, нужен аппарат моделирования, который в принципе должен быть специализированным. Он должен предоставлять исследователю:

- удобные способы организации данных, обеспечивающие простое и эффективное моделирование;
- удобные средства формализации и воспроизведения динамических свойств моделируемой системы;

- возможность имитации стохастических систем, т.е. процедур генерации ПСЧ и вероятностного (статистического) анализа результатов моделирования;
- простые и удобные процедуры отладки и контроля программы;
- доступные процедуры восприятия и использования языка и др.

Вместе с тем существующие языки программирования общего назначения (ЯОН) для достаточно широкого круга задач позволяют без значительных затрат ресурсов создавать весьма совершенные имитационные модели. Можно сказать, что они способны составить конкуренцию специализированным языкам моделирования. Для систематизации представлений о средствах реализации имитационных моделей приведем основные определения и краткие сведения о подходах к выбору соответствующего языка. *Языком программирования* называют набор (систему) символов, распознаваемых ЭВМ и обозначающих операции, которые можно реализовать на ЭВМ. Выделяют машинно-ориентированные, проблемно (процедурно)-ориентированные и объектно-ориентированные языки.

Классические языки моделирования являются процедурно-ориентированными и обладают рядом специфических черт. Можно сказать, что основные языки моделирования разработаны как средство программного обеспечения имитационного подхода к изучению сложных систем.

Языки моделирования позволяют описывать моделируемые системы в терминах, разработанных на базе основных понятий имитации. С их помощью можно организовать процесс общения заказчика и разработчика модели. Различают языки моделирования *непрерывных* и *дискретных процессов.*

В настоящее время сложилась ситуация, когда не следует противопоставлять ЯОН и языки имитационного моделирования (ЯИМ). На рис. 11.3 представлена классификация языков программирования по различным основаниям, которая может служить основой для формирования рационального подхода к выбору конкретного языка реализации имитационной модели исследуемой ЭИС, о чем будет подробнее сказано ниже [48].

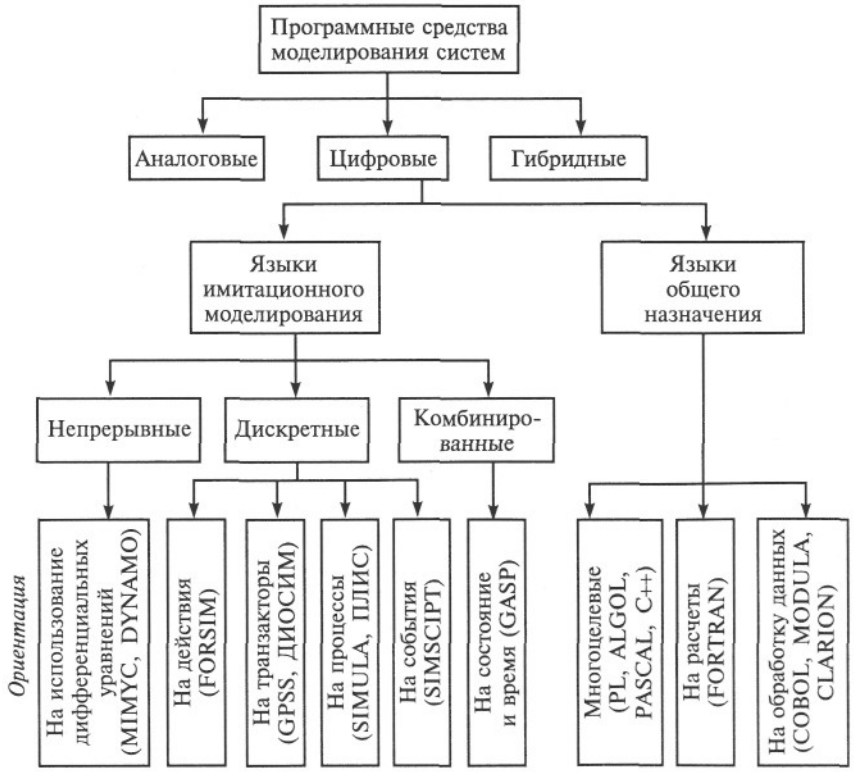


Рис. 11.3. Классификация программных средств моделирования систем

Легко заметить из названий, что некоторые ЯИМ базируются на конструкциях ЯОН: например, FORSIM — на языке FORTRAN; ПЛИС — на языке PL и т.д.

В силу своего целевого назначения при правильном выборе и использовании языки моделирования обладают рядом неоспоримых достоинств.

Вместе с тем им присущи и определенные недостатки, главными из которых являются сугубо индивидуальный характер соответствующих трансляторов, затрудняющий их реализацию на различных ЭВМ; низкая эффективность рабочих программ; сложность процесса отладки программ; нехватка

документации (литературы) для пользователей и специалистов-консультантов и др. В ряде случаев эти недостатки способны перечеркнуть любые достоинства.

Существует несколько подходов к выбору языка, на котором будет реализовываться разрабатываемая имитационная модель. Предлагается классическая двухэтапная схема выбора, имеющая широкое практическое применение.

На первом этапе следует найти ответы на следующие вопросы.

1. Имеются ли хорошо написанные руководства и инструкции для пользователей?
2. Совместим ли язык транслятора с имеющимися ВС?
3. Можно ли данный язык использовать на других ВС, способных решать задачи пользователя?
4. Обеспечивает ли транслятор языка выдачу информации об ошибках и глубокую их диагностику?
5. Насколько эффективен данный язык с учетом общего времени подготовки, программирования, отладки программы, компиляции и прогона ее на ЭВМ?
6. Какова стоимость внедрения, эксплуатации и обновления ПО для данного языка?
7. Знаком ли язык и, если нет, легко ли его изучить?
8. Оправдает ли частота использования языка в различных будущих моделях затраты на его изучение и освоение?

По результатам ответов на данные вопросы, как правило, отбирается несколько языков. Окончательный выбор основывается на учете характеристик конкретной задачи при ее решении на определенной машине.

Второй этап выбора предусматривает поиск ответов на такие вопросы.

1. Какова область применения языка и его пригодность для описания явлений реального мира (методы прогнозирования; ориентация; способность генерировать случайные факторы)?
2. Насколько легко осуществляется хранение и извлечение данных, характеризующих состояния системы и работу отдельных ее частей?
3. Обеспечивается ли необходимая гибкость и каковы возможности языка в отношении модифицирования состояний системы?
4. Насколько легко данный язык может описывать динамическое поведение?
5. Каковы выходные формы документов, чем они полезны и какой статистический анализ возможен на основе этих данных?
6. Насколько просто вставлять в модель стандартные подпрограммы, написанные пользователями?

Приведенные вопросы можно конкретизировать или расширять с учетом современного уровня и перспектив развития технических, программных средств и информационных технологий, но изложенный подход к выбору языка является неизменно актуальным и конструктивным.

Если попытаться обобщить направленность данных вопросов, то можно заметить, что важнейшими проблемами применения языков моделирования являются их эффективность, совместимость с другими программными средствами и возможность установки на имеющиеся технические средства, а также затраты различных ресурсов. Иными словами, при выборе программного средства моделирования следует руководствоваться известным критерием «эффективность — время — стоимость», причем зачастую важность каждого из этих частных показателей меняется в зависимости от существа задачи; объема располагаемых ресурсов; резерва (дефицита) времени; сложившихся условий и т. п. Вообще говоря, данная рекомендация справедлива для выбора весьма широкого круга сложных объектов различной природы.

РАЗДЕЛ IV. ОСНОВЫ ПОСТРОЕНИЯ И ИСПОЛЬЗОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Глава 12. МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ ТЕОРИИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

12.1. Историческая справка

Термин «искусственный интеллект» относится к группе терминов и понятий, получивших весьма широкое распространение, в том числе и у неспециалистов. Не пытаясь сразу четко и полно определить это понятие, заметим, что большинство людей трактуют ИИ как сравнительно новое научно-техническое направление, с которым связывают надежды на резкое увеличение функциональных

возможностей технических объектов, в частности ВС, используемых в качестве средств автоматизации различных сфер профессиональной деятельности человека: управления; проектирования; производства; обучения; индустрии обслуживания и развлечений и т. п. История ИИ насчитывает всего несколько десятилетий, хотя создать «думающую машину» мечтали многие поколения людей. Первый международный конгресс по искусственному интеллекту состоялся в США в 1969 г., ему предшествовали исследования в разных странах и направлениях:

- в 50-е гг. XX в. были начаты работы по машинному переводу с одного языка на другой;
- в 1957 г. А. Розенблатом было предложено устройство для распознавания образов — персептрон, положивший начало разработке большого числа других устройств подобного типа (в том числе и в СССР — например, в Риге Л.Гореликом);
- в 1959 г. Г.Саймон и А.Ньюэлл разработали программу *GPS (General Problem Solving Programme)* — универсальный решатель, предназначенный для разрешения различных задач из самых разнообразных областей [12] и др.

Как явствует из перечисленных примеров, первоначально на системы, связанные с ИИ, пытались возложить различные, но весьма универсальные задачи.

Однако надежды, порожденные первыми успехами в данной области, полностью не оправдались. Задача машинного перевода оказалась гораздо сложнее, чем предполагалось, и ее реализация, прогнозировавшаяся на 60-е гг. XX в., отодвинулась на два десятилетия (кстати, гораздо большее распространение получили автоматизированные словари, позволяющие получить из исходного текста так называемый «подстрочный перевод», а не «переводчики» в полном смысле этого слова). Универсальный решатель задач, довольно успешно доказывавший достаточно простые логические теоремы, оказался крайне неэффективным при решении других задач, в частности при многочисленных попытках автоматизировать игру в шахматы. Не удавалось также достаточно эффективно распознавать реальные изображения устройствами персептронного типа.

Вместе с тем, несмотря на неудачи, первый этап имел большое значение для ИИ с точки зрения двух обстоятельств: во-первых, он показал возможности использования ЭВМ в автоматическом режиме при решении задач, ранее решаемых только человеком; во-вторых, на этом этапе были отработаны различные способы и приемы решения интеллектуальных задач, хотя строгой теории ИИ еще не было.

Работы по ИИ продолжались, разрабатывались глубинные теоретические вопросы и их программная реализация. В частности, многие исследователи продолжили разработку алгоритмов и программ шахматной игры, завершившейся в 1967 г. первым чемпионатом мира по шахматам среди ЭВМ (его выиграла советская программа «Каисса», разработанная в Институте проблем управления АН СССР). Отметим, что до последнего времени продолжаются попытки решить спор о том, способна ли программа на ЭВМ переиграть человека (иногда такой процесс называют соревнованиями «кремниевого» и «белкового» шахматистов). В частности, чемпион мира Г.Каспаров неоднократно играл с шахматными программами, разработанными в разных странах (США, Германии), с переменным успехом. Многие специалисты связывают успех или неудачу человека в игре с машиной с регламентом проведения матча: при проведении блиц-партий человек, действующий по интуиции, имеет больше шансов на выигрыш; при ограничении времени на партию 5 или 25 мин («быстрые шахматы») возможности ЭВМ по просчету вариантов в ряде случаев превышают человеческие; сложнее отдать преимущество одной из сторон при классической игре.

Новое развитие работы по ИИ получили в 80-е гг. XX в. Теоретический задел, созданный на первом этапе развития интеллектуальных систем при решении достаточно «мелких» задач (машинные игры в шашки, шахматы; сочинение стихов и музыки; перевод и т.п.), привел к важным практическим результатам — таким, как создание экспертных систем, интеллектуальных расчетно-логических и информационно-поисковых систем, интеллектуальных пакетов прикладных программ (ИППП) и т.д. Кроме того, практические успехи в создании СИИ вызвали к жизни новые проекты, в частности проекты разработки ЭВМ следующего (по наиболее распространенной классификации), 5-го поколения, которые должны уметь общаться с пользователем на естественном (или близком к нему) языке; решать не вполне структурированные задачи; давать разумные советы по широкому кругу проблем и т.д.

Вместе с тем специалисты продолжают обсуждать многие основополагающие вопросы, например: что такое ИИ? искусственный разум? в чем их отличия? что является предметом теории ИИ?

12.2. Основные понятия и определения теории интеллектуальных информационных систем

Следует отметить, что строгого (формального, научного) определения понятия «естественный интеллект», вообще говоря, не существует. В силу этого еще труднее определить понятие «искусственный интеллект». Для того чтобы решить эту задачу, необходимо уяснить значение таких терминов, как интеллект, разум, сознание, психика.

Интеллект. Различают формулировки данного понятия по нескольким направлениям: философскую, биологическую, психологическую [21, 22].

В философии под интеллектом понимают познание, понимание, рассудочную способность к абстрактно-аналитическому расчленению (Г. Гегель), способность к образованию понятий (Э. Кант).

В психологии под интеллектом понимают характеристику умственного развития индивидуума, определяющую его способность целенаправленно действовать, рационально мыслить и эффективно взаимодействовать с окружающим миром.

В биологии под интеллектом понимают способность адекватно реагировать (принимать решения) в ответ на изменение окружающей обстановки.

Важно отметить: интеллект — это свойство отдельного субъекта. В частности, интеллектом может обладать не только человек, но и любой объект, обладающий указанными выше качествами — способностью к образованию понятий, абстрактно-аналитическому мышлению, целенаправленному действию.

Разум. В отличие от интеллекта разум — категория сугубо человеческая, опирающаяся на сознание как высшую форму психологической деятельности. Принципиальным моментом в определении разума, так же как и сознания, является его общественный, социальный характер, поскольку и то, и другое понятия сформировались в результате совместной человеческой деятельности.

Часто используют совместно понятия «рассудок» и «разум». Интересно, что в античной философии считалось, что если рассудок — способность рассуждения — познает все относительное, земное и конечное, то разум, сущность которого состоит в целеполагании, открывает абсолютное, божественное и бесконечное. В настоящее время с рассудком связывают способность: 1) строго оперировать понятиями; 2) правильно классифицировать факты и явления; 3) приводить знания в определенную систему. Опираясь на рассудок, разум выступает как творческая познавательная деятельность, раскрывающая сущность действительности. Посредством разума мышление синтезирует результаты познания, создает новые идеи, выходящие за пределы сложившихся систем знания.

Сознание. Это понятие также трактуется различными науками неоднозначно.

С точки зрения философии сознание — свойство высокоорганизованной материи — мозга, выступающее как осознанное бытие, субъективный образ объективного мира, субъективная реальность.

При социологическом подходе сознание рассматривается прежде всего как отображение в духовной жизни людей, интересов и представлений различных социальных групп, классов, наций, общества в целом.

В психологии сознание трактуется как особый, высший уровень организации психической жизни субъекта, выделяющего себя из окружающей действительности, отражающего эту действительность в форме психических образов, которые служат регуляторами целенаправленной деятельности [26]. Важнейшей функцией сознания является мысленное построение действий и предвидение их последствий, контроль и управление поведением личности, ее способность отдавать себе отчет в том, что происходит как в окружающем, так и в собственном духовном мире.

Психика. Это свойство высокоорганизованной материи — мозга, являющееся особой формой отражения действительности и включающее такие понятия, как ощущение, восприятие, память, чувства, воля, мышление и др. Отметим, что мышление и память, которыми обычно характеризуют интеллект, входят в понятие психики составными частями.

В психике выделяют две компоненты: ч у в с т в е н н у ю (ощущения, восприятие, эмоции) и р а ц и о н а л ь н у ю , мыслительную (интеллект, мышление). Другие составляющие психики — память и волю — можно разделить на память чувств и память мыслей; волю чувств и волю мыслей (инстинкты и долг перед собой и обществом соответственно).

Например, можно помнить, как берется сложный интеграл (память мыслей), а можно помнить ощущение напряжения и усталости при изучении способа его взятия (память чувств), когда воля чувств (инстинкт самосохранения, желание отдохнуть) боролась с волей мыслей (сознанием необходимости изучения этого способа).

Перечисленные понятия обычно разделяют на две пары: психика и интеллект как ее составляющая; сознание и разум как его составляющая, причем интеллект и разум — рассудочные, мыслительные

составляющие соответственно психики и сознания.

Основное отличие второй пары от первой состоит в том, что она образовалась в результате социальной, общественной деятельности людей, и поэтому социальный компонент — неотъемлемая и существенная черта сознания и разума.

Отсюда следует очень важный вывод: принципиально невозможно моделировать сознание и разум во всей полноте, так как для этого пришлось бы моделировать не только человека, но и всю систему его социально-общественных отношений. В то же время моделировать интеллект как один из компонентов психики отдельных индивидуумов вполне возможно, хотя и очень сложно.

К этому выводу «примыкает» еще один: ИИ — это модель рациональной, мыслительной составляющей психики. Не моделируются эмоции, ощущения, воля, память чувств и т.п. Машинное сочинение стихов и музыки — это моделирование лишь логического компонента психической деятельности, сопровождающей эти виды творчества (соблюдение рифмы, размера, законов композиции, гармонии и т.п.). Именно с этим связано неудовлетворительное для большинства людей качество машинных «сочинений».

Учитывая сказанное, можно заключить, что понятие «искусственный интеллект» объединяет три других [21, 22]:

- искусственный бессловесный интеллект — модель компоненты психики живых существ, отражающая их способность принимать решения, изменять поведение и т.д. на уровне инстинктов, не имеющих словесного выражения (самосохранение, размножение, приспособление и т.п.);
- искусственный словесный интеллект — модель рационального компонента психической деятельности человека без учета ее социального содержания;
- искусственный разум — искусственный словесный интеллект, дополненный социальным компонентом.

В дальнейшем, если не будет специальных оговорок, под ИИ будем понимать искусственный словесный интеллект.

Приведенные определения основаны на теоретических рассуждениях и в силу этого носят достаточно общий характер.

Существует по крайней мере три подхода к определению этого понятия, носящие гораздо большую практическую направленность (рис. 12.1).

Рис. 12.1. Подходы к определению понятия «искусственный интеллект»

Достаточно полным определением понятия «искусственный интеллект» первого типа является следующее: ИИ — это область исследований, в рамках которых разрабатываются модели и методы решения задач, традиционно считавшихся интеллектуальными и не поддающимися формализации и автоматизации [22].

Применительно к данному определению является справедливым суждение, что интеллектуальной может считаться такая искусственно созданная система, для которой выполняется тест Тьюринга, состоящий в следующем: «Испытатель через посредника общается с невидимым для него собеседником — человеком или системой. Интеллектуальной может считаться та система, которую испытатель в процессе такого общения не может отличить от человека» [54].

В качестве другого определения, достаточно точно отражающего характер второго подхода, может рассматриваться следующее: ИИ — это область исследований, в которой изучаются системы, строящие результирующий вывод для задач с неизвестным алгоритмом решения на основе неформализованной исходной информации, использующие технологии символьного программирования и средства вычислительной техники со специальной (не фоннеймановской) архитектурой [22, 54].

Наконец, наиболее цитируемым определением третьего типа является следующее: ИИ — это область знаний, которая находит применение при решении задач, связанных с обработкой информации на

естественном языке, автоматизацией программирования, управлением роботами, машинным зрением, автоматическим доказательством теорем, разумными машинами извлечения информации и т.д. [54].

Можно рассмотреть и такое — в определенной степени обобщающее — определение: ИИ — научная дисциплина, задачей которой является разработка математических описаний функций человеческого (словесного) интеллекта с целью аппаратурной, программной и технической реализации этих описаний средствами вычислительной техники [54].

В заключение рассмотрения данного подраздела отметим, что в последние годы многие специалисты согласились, что дискуссия по вопросу об определении самого термина «искусственный интеллект» приобрела схоластический характер, не дает конструктивных результатов теории и практике и может быть бесконечной. Поэтому вместо термина «искусственный интеллект» предлагается использовать другой — «новая информационная технология решения инженерных задач», что подчеркивает приоритетную роль поиска, анализа и синтеза информации в СИИ.

12.3. Классификация интеллектуальных информационных систем

Рассмотрим классификацию этих систем, имея в виду, что, несмотря на значительное число попыток провести такую классификацию (см., например, [21, 22, 26, 27, 41]), ни одна из них, на наш взгляд, не является совершенной. На рис. 12.2 представлена классификация СИИ, полученная путем сопоставления и обобщения известных классификаций этих систем.

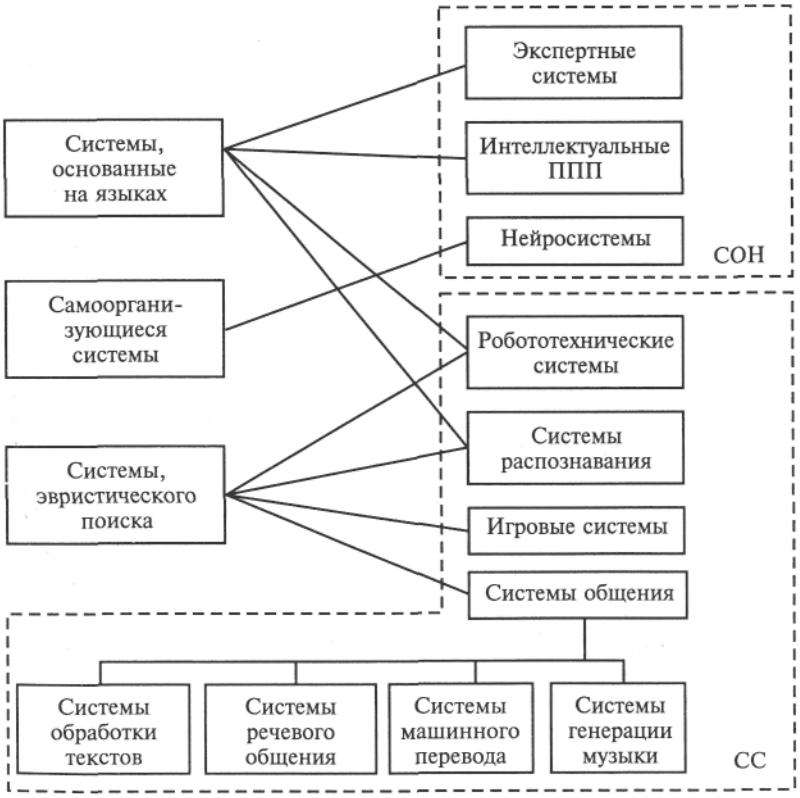


Рис. 12.2. Классификация систем искусственного интеллекта:
СОН — системы общего назначения; СС — специализированные системы

Наиболее широкое распространение на практике в настоящее время получили СИИ, основанные на знаниях. Понятие «знания» для этих систем имеет принципиальное значение. Под «знанием» в СИИ понимается информация о предметной области, представленная определенным образом и используемая в процессе логического вывода. По своему содержанию данная информация является некоторым набором суждений и умозаключений, описывающих состояние и механизмы (логику) функционирования в выбранной, как правило, весьма ограниченной предметной области. Указанные суждения и умозаключения высказываются экспертом (специалистом) в этой области, либо формулируются в результате анализа литературы по данному предметному направлению.

Способы получения и представления знаний в интересах проектирования СИИ в настоящее время составляют предмет сравнительно нового научного направления — инженерии знаний.

Форма представления знаний имеет отличие от формы представления данных. Обычно (см., например, [53]) под данными в АИС понимаются факты и идеи, представленные в формализованном виде, позволяющие (лишь) передавать, хранить или обрабатывать эти факты и идеи при помощи некоторого процесса. В отличие от данных знания предполагают сосредоточение не только фактов и идей в указанном выше смысле (так называемых первичных данных), но и дополнительных данных, которые описывают (интерпретируют) первичные данные с точки зрения следующих составляющих: того, что собой представляют эти данные, какие между ними имеются связи, какие действия с ними и каким образом могут выполняться и т. п.

В системах, основанных на знаниях, предполагается, что исходные знания способны в соответствии с запросами пользователей к системе порождать новые знания. При этом сама процедура порождения новых знаний называется *логическим выводом* (или просто выводом). Термин «логический» в данном случае не случаен с двух точек зрения. Системы, основанные на знаниях, моделируют мыслительную деятельность людей лишь на логическом (а не на физиологическом) уровне, и, кроме того, основным математическим аппаратом, лежащим в основе систем этого типа, является аппарат математической логики.

К СИИ, полностью основанным на знаниях, относятся два класса систем: экспертные системы (ЭС) и ИППП. Основные идеи этого направления частично (или даже в значительной части) реализуются и в других СИИ, в частности робототехнических системах распознавания и др. (см. рис. 12.2).

Наиболее последовательно идеи, на которых базируются СИИ, основанные на знаниях, воплощены в ЭС, которые достаточно подробно рассмотрены в гл. 14.

Под ИППП понимаются инструментальные пакеты прикладных программ, в которых механизм сборки отдельных подпрограмм (решения частных задач) в общую программу решения требуемой задачи осуществляется автоматически, на основе механизма логического вывода.

В *самоорганизующихся системах* реализуется попытка осуществить моделирование интеллектуальной деятельности человека (или более простых живых существ) не на логическом, а на физиологическом уровне работы головного мозга. В данном случае мозг человека моделируется сетью идеальных нейронов. В соответствии с доказанной Дж. фон Нейманом [37] теоремой при воздействии на такую сеть некоторых раздражителей она начинает вырабатывать адекватную реакцию, т.е. способна к самообучению путем самоорганизации. Несмотря на значительную теоретическую перспективность этого (исторически первого) направления в области ИИ, практически значимых результатов этот путь пока не дал. Последнее объясняется технической нереализуемостью на современном уровне достаточного числа взаимосвязанных нейронов в искусственно создаваемой сети.

В то же время данное направление позволило получить весомые результаты в области исследования возможностей создания компьютеров сверхвысокого быстродействия и тем самым повысить возможности СИИ, создаваемых на других принципах. Кроме того, реальные результаты получены в создании нейросистем распознавания образов.

Основная идея, лежащая в основе создания нейросетей, базируется на теореме Мак-Каллока и Питтса [37], которая утверждает: любую вычислимую функцию можно реализовать с помощью сети идеальных нейронов. Эксперименты показывают, что реализация этих функций таким путем может осуществляться значительно быстрее, чем на традиционном компьютере. Компьютеры новой архитектуры, воплощающие данную идею, получили название *нейрокомпьютеры*.

Третье направление разработки СИИ связано с реализацией *эвристического подхода* к построению таких систем. Главной особенностью, характерной для данного направления, является полный отказ от следования принципу аналогии при моделировании механизма интеллектуальной деятельности (ни на логическом, ни на физиологическом уровнях). Методологической основой систем эвристического поиска служит то утверждение, что любая интеллектуальная деятельность начинается с некоторых данных и завершается получением определенных результатов также в виде данных. Если техническое устройство позволяет по аналогичным исходным данным получить эквивалентные результаты, то оно может быть отнесено к классу интеллектуальных (см. первое определение ИИ). При этом механизм переработки исходных данных в результаты не оговаривается и, вообще говоря, может быть совершенно иным по сравнению с реальным. Системы этого типа выполняют функции, которые традиционно производятся человеком, однако реализуют их другими способами.

Широкое распространение данное направление получило при решении различных игровых задач (шахматы, шашки и т.д.). Однако подходы, присущие этому направлению, нашли применение и в других СИИ, в частности системах общения (особенно в части речевого общения), системах

распознавания, робототехнических системах и др. В то же время следует заметить: специфика эвристического подхода такова, что рецепты создания программ для решения интеллектуальных задач в одной области практики, как правило, неприменимы в другой области, а возникающая необходимость изменения характера учета факторов при решении прикладных задач вызывает существенную перестройку программы в целом.

При разработке интеллектуальных робототехнических систем основная задача состоит в решении теоретических и практических вопросов организации целесообразного поведения подвижных роботов, снабженных сенсорными и эффекторными (исполнительными) механизмами [54]. Принципиальное отличие робототехнических систем от СИИ других типов заключается в том, что эти системы не только воспринимают информацию из окружающего мира и вырабатывают на ее основе определенные оценочные выводы, но и, сообразуясь с этими выводами, вносят изменение в окружающий (анализируемый ими) мир.

К настоящему времени в практике находят применение робототехнические системы с относительно простыми сенсорными и эффекторными механизмами, которые способны выполнять действия только в простых средах с заранее зафиксированными свойствами.

Основа проблемы распознавания образов или в более широком контексте — машинное зрение — заключается в придании системе способности разрешения задач преобразования огромного количества сенсорных данных (например, присутствующих в телевизионном изображении) к относительно краткому и осмысленному описанию наблюдаемой проблемной ситуации. Содержанием такого описания, как правило, является тот минимальный (самый характерный) набор данных, которые отличают изучаемую ситуацию от стандартной. Основная сложность такого описания связана с ответом на следующие вопросы: какие объекты имеют место в наблюдаемом кадре? какие из них являются ключевыми для выявленной ситуации? что надо принять за стандартную ситуацию для выявленных ключевых объектов? в чем отличие рассматриваемой ситуации от стандартной? откуда первоначально получать наборы стандартных ситуаций? Трудности, с которыми сталкивается практика при решении каждой из перечисленных задач, указывают на то, что, как и в случае робототехнических систем, данное направление находит реализацию только в самых простых случаях.

В дальнейшем будем рассматривать системы, основанные на знаниях, как получившие наибольшее практическое развитие и распространение в различных отраслях профессиональной деятельности, в том числе и в экономике, что обусловливает необходимость более подробного рассмотрения методов представления знаний в памяти ЭВМ.

Глава 13. МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

13.1. Знания и их свойства

Выше уже частично рассматривались такие понятия, как «знания» и «системы, основанные на знаниях», и отмечалась их особая значимость в теории ИИ. Сделаем еще одно весьма важное замечание: в настоящее время в области разработки СИИ сложилась следующая аксиома: никакой самый сложный и изощренный алгоритм извлечения информации (так называемый механизм логического вывода) из интеллектуальной системы не может компенсировать «информационную бедность» ее базы знаний.

Несмотря на широкое распространение и использование понятия «знания» в различных научных дисциплинах и на практике, строгого определения данного термина нет.

Довольно часто используют так называемый прагматический подход: говорят, что знания — это формализованная информация, на которую ссылаются и/или которую используют в процессе логического вывода. Однако такое определение ограниченно: оно фиксирует сознание на уже существующих методах представления знаниях и соответственно механизмах вывода, не давая возможности представить себе другие (новые).

Возможен и другой подход: попытаться на основе определения уже рассмотренного понятия «данные» (см. гл. 12) выявить их свойства и особенности, сформировать дополнительные требования к ним и уже затем перейти к понятию «знания».

Напомним, что данными называют формализованную информацию, пригодную для последующей обработки, хранения и передачи средствами автоматизации профессиональной деятельности.

Какие же свойства «превращают» данные в знания? Перечислим и кратко охарактеризуем шесть основных свойств знаний (часть из них присуща и данным).

Таблица 13.1

Внутренняя интерпретация

Предприятие	Место нахождения	Что выпускает
Завод им. Хруничева	Москва	Космическую технику
НПО «Энергия»	Королев	Космическую технику
НПО «Комета»	Москва	Конструкторскую документацию

1. *Внутренняя интерпретация* (интерпретируемость). Это свойство предполагает, что в ЭВМ хранятся не только собственно (сами) данные, но и данные о данных, что позволяет содержательно их интерпретировать (табл. 13.1). Имея такую информацию, можно ответить на вопросы типа «Где находится НПО «Энергия»?» или «Какие предприятия выпускают космическую технику?». При этом в первой строке находятся данные о данных (метаданные), а в остальных — сами данные.

2. *Внутренняя структура связей.* Предполагается, что в качестве информационных единиц используются не отдельные данные, а их упорядоченные определенными отношениями (родовидовыми, причинно-следственными и др.) структуры (эти отношения называют классифицирующими).

Пример: факультет — курс — учебная группа—студент.

3. *Внешняя структура связей.* Внутренняя структура связей позволяет описывать отдельный объект (понятие). Однако объекты (понятия) способны находиться и в других отношениях (вступать в ситуативную связь).

Пример: объекты «курс Государственного университета управления им. С. Орджоникидзе» и «урожай овощей в совхозе «Зареченский» могут находиться в ситуативной связи «принимает участие в уборке».

4. *Шкалирование.* Предполагает введение соотношений между различными информационными единицами (т.е. их измерение в какой-либо шкале — порядковой, классификационной, метрической и т.п.) и упорядочение информационных единиц путем измерения интенсивности отношений и свойств.

Пример: «97/ЭИ.6-01 учебная группа занимает первое место на курсе по успеваемости».

5. *Семантическая метрика.* Шкалирование позволяет соотнести информационные единицы, но прежде всего для понятий, имеющих «количественное» толкование (характеристики). На практике довольно часто встречаются понятия, к которым неприменимы количественные шкалы, но существует потребность в установлении их близости (например, понятия «искусственный интеллект» и «искусственный разум»). Семантики классифицируются следующим образом:

- значение, т. е. объективное содержание;
- контекстуальный смысл, определяемый связями данного понятия с другими, соседствующими в данной ситуации;
- личностный смысл, т.е. объективный значение, отраженное через систему взглядов эксперта;
- прагматический смысл, определяемый текущим знанием о конкретной ситуации (например, фраза «информация получена» может иметь как негативную, так и позитивную оценку — в зависимости от того, нужно это было или нет) [22].

6. *Активность.* Данное свойство принципиально отличает понятие «знание» от понятия «данные». Например, знания человека, как правило, активны, поскольку ему свойственна познавательная активность (обнаружение противоречий в знаниях становится побудительной причиной их преодоления и появления новых знаний, стимулом активности является неполнота знаний, выражается в необходимости их пополнения). В отличие от данных знания позволяют выводить (получать) новые знания. Будучи активными, знания позволяют человеку решать не только типовые, но и принципиально новые, нетрадиционные задачи.

Кроме перечисленных, знаниям присущи такие свойства, как *омонимия* (слово «коса» может иметь три смысла, связанных с определениями: девичья; песчаная; острая) и *синонимия* (знания «преподаватель читает лекцию» и «студенты слушают лекцию» во многих случаях являются синонимами) и др.

Классифицировать знания можно по самым различным основаниям.

По способу существования различают *факты* (хорошо известные обстоятельства) и *эвристики* (знания из опыта экспертов).

По способу использования в экспертных системах — *фактические знания* (факты) — знания типа «А — это А»; *правила* — знания для принятия решений («Если —то»); *метазнания* (знания о знаниях — указывают системе способы использования знаний и определяют их свойства). Классическими примерами метазнаний являются народные пословицы и поговорки, каждая из которых характеризует знания (рекомендации по деятельности) в широком классе конкретных ситуаций (например, пословица «Семь раз отмерь, один — отрежь» применима не только в среде хирургов или портных).

По формам представления знания делят на *декларативные* (факты в виде наборов структурированных данных) и *процедуральные* (алгоритмы в виде процедур обработки фактов).

По способу приобретения знания делятся на *научные* (полученные в ходе систематического обучения и/или изучения) и *житейские, бытовые* (полученные в «ходе жизни»).

Дадим еще ряд определений, часто встречающихся в литературе [21].

Интенсиональные знания — знания, характеризующие или относящиеся к некоторому классу объектов.

Экстенсиональные знания — знания, относящиеся к конкретному объекту из какого-либо класса (факты, сведения, утверждения и т.д.).

Заметим: отношения интенсиональных и экстенсиональных знаний — это родовидовые отношения. Например, понятие «технологическая операция» — это интенсивная, а понятие «пайка» — это экстенсионал, так как пайка — одна из технологических операций. Очевидно, что эти понятия относительны. Так, понятие «пайка», в свою очередь, можно считать интенсионалом по отношению к понятиям «пайка серебром» и «пайка оловом». Как правило, такого рода знания относятся к декларативным.

Физические знания — знания о реальном мире.

Ментальные знания — знания об отношениях объектов.

Мир задачи — совокупность знаний, используемых в задаче.

Мир пользователя — совокупность знаний пользователя.

Мир программы — совокупность знаний, используемых в программе.

Морфологические и *синтаксические знания* — знания о правилах построения структуры описываемого явления или объекта (например, правила написания букв, слов, предложений и др.).

Семантические знания — знания о смысле и значении описываемых явлений и объектов.

Прагматические знания — знания о практическом смысле описываемых объектов и явлений в конкретной ситуации (например, редкая монета для нумизмата и филателиста имеет различную прагматическую ценность).

Предметные знания — знания о предметной области, объектах из этой области, их отношениях, действиях над ними и др.

13.2. Классификация методов представления знаний

Для того чтобы манипулировать всевозможными знаниями из реального мира с помощью компьютера, необходимо осуществить их моделирование.

При проектировании модели представления знаний следует учесть два требования: однородность представления; простоту понимания.

Выполнение этих требований позволяет упростить механизм логического вывода и процессы приобретения знаний и управления ими, однако, как правило, создателям интеллектуальной системы приходится идти на некоторый компромисс в стремлении обеспечить одинаковое понимание знаний и экспертами, и инженерами знаний, и пользователями.

Классификация методов моделирования знаний с точки зрения подхода к их представлению в ЭВМ показана на рис. 13.1.

Рис. 13.1. Классификация моделей представления знаний

Дадим общую характеристику основных методов представления знаний.

Модели на основе эвристического подхода. *Представление знаний тройкой «объект — атрибут — значение».* Один из первых методов моделирования знаний. Как правило, используется для представления фактических знаний в простейших системах.

Примеры.

Объект	*Атрибут*	*Значение*
Студент	Успеваемость	Отличник
Дом	Цвет	Белый
Пациент	Температура	Нормальная

Очевидно, что для моделирования знаний даже об одном объекте (например, о «студенте» или «доме») из предметной области необходимо хранить значительное число «троек».

Продукционная модель. Модель правил; модель продукций — от англ. *production* — изготовление, выработка. В настоящее время наиболее проработанная и распространенная модель представления знаний, в частности в ЭС.

Модель предусматривает разработку системы продукционных правил (правил продукций), имеющих вид:

$$\text{ЕСЛИ } A_1 \text{ И } A_2 \text{ И ... И } A_n, \text{ ТО } B_1 \text{ ИЛИ } B_2 \text{ ИЛИ ... ИЛИ } B_m,$$

где A_i и B_j — некоторые высказывания, к которым применены логические операции И и ИЛИ. Если высказывания в левой части правила (ее часто называют *антецедент* — условие, причина) истинно, истинно и высказывание в правой части *(консеквент* — следствие).

Полнота базы знаний (базы правил) определяет возможности системы по удовлетворению потребностей пользователей. Логический вывод в продукционных системах основан на построении прямой и обратной цепочек заключений, образуемых в результате последовательного просмотра левых и правых частей соответствующих правил, вплоть до получения окончательного заключения.

Пусть в некоторой области памяти (базе знаний) хранятся следующие правила (суждения):

- правило 1: ЕСЛИ в стране происходит падение курса национальной валюты, ТО материальное положение населения ухудшается;
- правило 2: ЕСЛИ объемы производства в стране падают, ТО курс национальной валюты снижается;
- правило 3: ЕСЛИ материальное положение населения ухудшается, ТО уровень смертности в стране возрастает.

Если на вход системы поступит новый факт (факт 1) «В стране высокий уровень падения объемов производства», то из правил можно построить цепочку рассуждений и сформулировать два заключения:

факт 1 — правило 2 — правило 1 — заключение 1 — правило 3 — заключение 2,

где заключение 1 (промежуточный вывод) — «Материальное положение населения ухудшается»; заключение 2 (окончательный вывод) — «В стране возрастает уровень смертности».

Отметим, что в современных ЭС в базе знаний могут храниться тысячи правил, а коммерческая стоимость одного невыводимого (нового, дополнительного) правила весьма высока.

Главными достоинствами продукционных систем являются простота пополнения и изъятия правил; простота реализации механизма логического вывода и наглядность объяснений результатов работы системы.

Основной недостаток подобных систем — трудность обеспечения непротиворечивости правил при их большом числе, что требует создания специальных правил (так называемых метаправил) разрешения возникающих в ходе логического вывода противоречий. Кроме того, время формирования итогового заключения может быть достаточно большим.

Фреймовая модель. Сравнительно новая модель представления знаний. Само понятие «фрейм» (англ. *frame* — рама, рамка, скелет, сгусток, сруб и т.д.) было введено в 1975 г. М.Минским (M.Minsky, США).

Фрейм — это минимальная структура информации, необходимая для представления знаний о стереотипных классах объектов, явлений, ситуаций, процессов и др. С помощью фреймов можно моделировать знания о самых разнообразных объектах интересующей исследователя предметной области — важно лишь, чтобы эти объекты составляли класс концептуальных (повторяющихся: стереотипных) объектов, процессов и т. п. Примерами стереотипных жизненных ситуаций могут служить собрание, совещание; сдача экзамена или зачета; защита курсовой работы и др. Примеры стереотипных бытовых ситуаций: отъезд в отпуск, встреча гостей, выбор телевизора, ремонт и др. Примеры стереотипных понятий: алгоритм; действие; методика и др. На рис. 13.2 представлен фрейм технологической операции «соединять» [21].

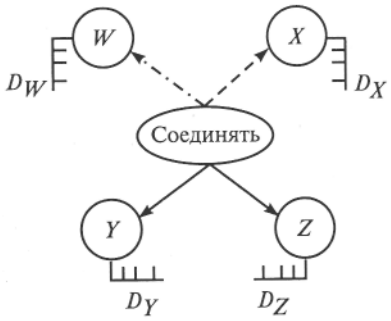


Рис. 13.2. Фрейм ситуации «соединять»:
дуги — отношения: ⟶ «объект», - - - ⟶ «субъект», -·-·⟶ «посредством чего?»; вершины *X*, *Y*, *Z*, *W* — *слоты* (англ. *slot* — прорез; щель; пустота) — составляющие фрейма; D_X, D_Y, D_Z, D_W — шанции — области возможных значений соответствующих слотов

Данный фрейм описывает ситуацию «Субъект *X* соединяет объект *Y* с объектом *Z* способом *W*».

Наполняя слоты конкретным содержанием, можно получить фрейм конкретной ситуации, например: «Радиомонтажник соединяет микросхему с конденсатором способом пайки». Заполнение слотов шанциями называют *активизацией фрейма.*

С помощью фреймов можно моделировать как процедурные, так и декларативные знания. На рис. 13.2 представлен пример представления процедурных знаний.

На рис. 13.3 приведен пример фрейма «технологическая операция», иллюстрирующий представление декларативных знаний для решения задачи проектирования технологического процесса.

Рис. 13.3. Фрейм понятия «технологическая операция»

По содержательному смыслу фрейма выделяют [21]:

- фреймы-понятия;
- фреймы-меню;
- фреймы с иерархически вложенной структурой.

Фрейм-понятие — это фрейм типа И. Например, фрейм «операция» содержит объединенные связкой И имена слотов «что делать», «что это дает», «как делать», «кто делает», «где делать» и т.д., а фрейм «предмет» — слоты с именами «назначение», «форма», «вес», «цвет» и т.д.

Фрейм-меню — это фрейм типа ИЛИ. Он служит для организации процедурных знаний с помощью оператора «выбрать». Например, фрейм «что делать» может состоять из объединенных связкой ИЛИ слотов «решить уравнение», «подставить данные», «уточнить задачу» и т.д., причем каждый из этих слотов может иметь несколько значений.

Фрейм с иерархически вложенной структурой предполагает, что в нем в качестве значений слотов можно использовать имена других фреймов, слотов и т.д., т. е. использовать иерархическую структуру, в которой комбинируются другие виды фреймов (в итоге получают так называемые фреймы-сценарии).

Значения слотов могут содержать ссылки на так называемые присоединенные процедуры. Различают два вида присоединенных процедур: процедуры-демоны; процедуры-слуги.

Процедуры-демоны присоединяются к слоту и активизируются при изменении информации в этом слоте (выполняют вспомогательные операции — например, автоматически корректируют информацию во всех других структурах, где используется значение данного слота).

1. Процедура «Если — добавлено» (IF —ADDED) выполняется, когда новая информация помещается в слот.

2. Процедура «Если — удалено» (IF —REMOVED) выполняется, когда информация удаляется из слота.

3. Процедура «Если —нужно» (IF —NEEDED) выполняется, когда запрашивается информация из пустого слота.

Процедуры-слуги активизируются при выполнении некоторых условий относительно содержимого слотов (часто по запросу). Данные процедуры определяются пользователем при создании фрейма. Например, во фрейме «Учебная аудитория» можно предусмотреть слоты «Длина» и «Ширина», а по их значениям вычислять значение слота «площадь».

Фреймы позволяют использовать многие свойства знаний и достаточно широко употребляются. Их достоинства и недостатки схожи с достоинствами и недостатками семантических сетей, которые будут рассмотрены ниже.

Модель семантической сети (модель Куилиана). Семантическая сеть — это направленный граф с поименованными вершинами и дугами, причем узлы обозначают конкретные объекты, а дуги —

отношения между ними [21]. Как следует из определения, данная модель представления знаний является более общей по отношению к фреймовой модели (иными словами, фреймовая модель — частный случай семантической сети). Семантическую сеть можно построить для любой предметной области и для самых разнообразных объектов и отношений.

В семантических сетях используют три типа вершин:

- вершины-понятия (обычно это существительные);
- вершины-события (обычно это глаголы);
- вершины-свойства (прилагательные, наречия, определения).

Дуги сети (семантические отношения) делят на четыре класса:

- лингвистические (падежные, глагольные, атрибутивные);
- логические (И, ИЛИ, НЕ);
- теоретико-множественные (множество — подмножество, отношения целого и части, родовидовые отношения);
- квантифицированные (определяемые кванторами общности $\forall$ и существования $\exists$).

(Кванторы — это логические операторы, переводящие одну высказывательную форму в другую и позволяющие указывать объем тех значений предметных переменных, для которых данная высказывателъная форма истинна).

Приведем два примера.

На рис. 13.4 представлена семантическая сеть для предложения (ситуации) «студент Табуреткин добросовестно изучает новый план счетов на 2002 г. перед сдачей экзамена по дисциплине «Бухгалтерский учет».

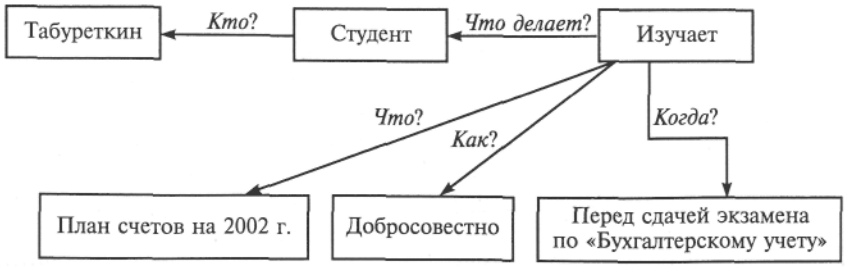


Рис. 13.4. Семантическая сеть для предложения (ситуации)

Рис. 13.5 содержит фрагмент семантической сети для понятия «автомобиль».

Рис. 13.5. Фрагмент семантической сети понятия «автомобиль»:
IS-A — есть, является; HAS-PART — имеет часть

Из приведенных примеров понятно, почему многие специалисты по ИИ считают фрейм частным случаем семантической сети со строго структурированными знаниями.

Основное достоинство методов моделирования знаний с помощью семантических сетей и фреймов — универсальность, удобство представления как декларативных, так и процедуральных знаний. Существует и два недостатка:

- громоздкость, сложность построения и изменения;
- потребность в разнообразных процедурах обработки, связанная с разнообразием типов дуг и вершин.

Модели на основе теоретического подхода. В рамках реализации теоретического подхода применяют логические модели, прежде всего использующие представления знаний в системе *логики предикатов.* Преимущества такого подхода очевидны: единственность теоретического обоснования и возможность реализации системы путем введения формально точных определений и правил получения выводов. Однако в полной мере претворить в жизнь данный подход даже для «простых» задач оказалось весьма сложно. Поэтому появились попытки перейти от формальной логики к так называемой *человеческой логике* (модальной логике, многозначной логике и др.), модели которой в большей или меньшей степени учитывают «человеческий фактор», т.е. являются в определенном смысле компромиссными в плане использования и теоретического, и эвристического подходов.

Очень коротко остановимся на ставшей классической предикатной модели представления знаний. Первые попытки использовать такую модель относятся к 50-м гг. прошлого века. Дадим несколько определений.

Пусть имеется некоторое множество объектов, называемое предметной областью. Выражение $P(x_1, x_2, ..., x_n)$, где $x_1 ... x_n$ — предметные переменные, а P принимает значения 0 или 1, и называется *логической функцией,* или *предикатом.*

Предикат $P(x_1, x_2, ..., x_n)$ задает отношение между элементами $x_1\ x_2, ..., x_n$ и обозначает высказывание, что «$x_1, x_2, ..., x_n$ находятся между собой в отношении P». Например, если A — множество целых чисел, а $P(a)$ — высказывание «a — положительное число», то $P(a) = 1$ при $a > 0$ и $P(a) = 0$ при $a \le 0$.

Из подобного рода элементарных высказываний с помощью логических связок образуют более сложные высказывания, которые могут принимать те же значения — «истина» и «ложь». В качестве связок используются конъюнкция, дизъюнкция, импликация, отрицание, эквивалентность.

Предикат от n переменных называют n-местным.

Одноместные (унарные) предикаты отражают свойства определенного объекта или класса объектов. Многоместные предикаты позволяют записывать отношения, которые существуют между группой элементов.

Если a — тоже предикат, то $P(a)$ — предикат 2-го порядка и далее до n-го порядка.

Приведем примеры различных предикатов.

1. Унарный предикат (высказывание) «река впадает в Каспийское море» имеет значение 1, если «река» = «Волга», и значение 0, если «Река» = «Днепр».

2. Двухместный предикат «x_1 не меньше x_2» может иметь значение 1 или 0 в зависимости от значений x_1 и x_2. Если значение предиката тождественно равно 1 при любых значениях предметных переменных, он называется *тавтологией.*

В аппарат исчисления предикатов входят также символы функций (обычно обозначаемые латинскими буквами f, g, h и т.д.), задаваемых на множестве предметных переменных, и кванторы общности $\forall$ и существования $\exists$.

3. Представление с помощью предиката знаний, заключенных в теореме Пифагора: $P\{g[f(x),f(y)],f(z)\}$, где предикат P— «быть равным», функция $g(x, y) = x + y$; функция $f(x) = x^2$.

Иногда используется такая форма записи:

РАВНЫ [СУММА (КВАДРАТ(x), КВАДРАТ(y)), КВАДРАТ (z)].

Предикат P равен 1, если x, y, z — соответственно длины катетов и гипотенузы прямоугольного треугольника.

Как уже отмечалось, предикаты удобны для описания декларативных знаний (фактов, событий и т.п.). Их главные достоинства — возможность реализации строгого вывода знаний (исчисления предикатов) и сравнительная компактность модели. К сожалению, предикаты мало пригодны для записи процедуральных знаний. Кроме того, опыт показал, что человеческое знание по своей структуре много сложнее структуры языков предикатного типа, поэтому требуются специальные навыки «подгонки» структуры реального знания под структуру модели (как правило, значительно обедняющей исходные знания).

Глава 14. ЭТАПЫ ПРОЕКТИРОВАНИЯ ЭКСПЕРТНЫХ СИСТЕМ

14.1. Структура и назначение экспертных систем

В настоящее время среди всех СИИ наибольшее распространение (по некоторым оценкам до 90 %) получили ЭС различных типов. Объяснение этому находится в самой истории развития технологии ИИ. Если условно проследить начало этой истории по десятилетиям, увидим, что в 60-х гг. XX в. специалисты в области ИИ пытались моделировать сложный процесс мышления, отыскивая общие методы решения широкого класса задач и реализуя их в универсальных программах [38]. Как уже отмечалось, большая часть таких попыток была неудачной.

Дальнейшие исследования в 70-е гг. XX в. были сконцентрированы на разработке двух групп методов:

- методов представления задач (в стремлении сформулировать решаемую проблему так, чтобы ее было легче решить);
- методов поиска (вывода) ответа (в стремлении создать достаточно хитроумные способы управления ходом решения задачи, обеспечивающие приемлемый расход машинных ресурсов).

Однако и эта стратегия не принесла реальных успехов. Только в конце 70-х гг. XX в. был сделан принципиальный вывод: эффективность программы при решении интеллектуальных задач в большей степени зависит от знаний, которыми она обладает, а не только от используемых формализмов и схем вывода. Чтобы сделать систему интеллектуальной, ее нужно снабдить множеством высококачественных знаний о некоторой предметной области. Это послужило основой новой концепции развития СИИ — создания специализированных программных систем, каждая из которых является как бы экспертом в некоторой узкой предметной области. Такие программы в дальнейшем и стали называть ЭС.

Огромный интерес к ЭС обусловлен тремя основными обстоятельствами [22]:

- ЭС ориентированы на решение широкого круга задач в ранее неформализуемых областях, которые считались малодоступными для использования ЭВМ;
- ЭС предназначены для решения задач в диалоговом режиме со специалистами (конечными пользователями), от которых не требуется знания программирования — это резко расширяет сферу использования вычислительной техники, которая в данном случае выступает как инструмент подкрепления (поддержки) памяти специалиста и усиления его способностей к логическому выводу;
- специалист, использующий ЭС для решения своих задач, может достигать, а иногда и превосходить по результатам возможности экспертов в данной области знаний, что позволяет резко повысить квалификацию рядовых специалистов за счет аккумуляции знаний в ЭС, в том числе знаний экспертов высшей квалификации.

Свое название ЭС получили по двум причинам:

- информацию (знания) для них поставляют эксперты;
- ЭС выдает решения, аналогичные тем, которые формулируют эксперты.

Понятие «эксперт» заслуживает отдельного обсуждения.

По Д. Уотермену, *эксперт* (англ. *domain expert* — знаток, специалист в области, сфере деятельности) — человек, который за годы обучения и практики научился чрезвычайно эффективно решать задачи, относящиеся к конкретной предметной области [54]. Главным в этом определении является требование к эксперту, которое предъявляются и к ЭС: эффективность решения конкретных задач из узкой предметной области.

В соответствии с определением П.Джонса, «эксперт — это человек, который благодаря обучению и опыту может делать то, что мы все, остальные люди, делать не умеем; эксперты работают не просто профессионально, но к тому же уверенно и эффективно. Эксперты обладают огромными познаниями и пользуются различными приемами и уловками для применения своих знаний к проблемам и заданиям; они также умеют быстро переворошить массу несущественной информации, чтобы добраться до главного, и хорошо умеют распознавать в проблемах, с которыми сталкиваются, примеры тех типовых проблем, с которыми они уже знакомы. В основе поведения экспертов лежит совокупность практически применимых знаний, которую мы будем называть компетентностью. Поэтому разумно предположить, что эксперты — это те люди, к которым надо обратиться, когда мы желаем проявить компетентность, делающую возможным такое поведение, как у них» [54].

Отметим, что в обоих определениях подчеркиваются источники знаний экспертов — обучение и практика (опыт).

Таким образом, можно дать следующее определение: под ЭС понимается программная система, выполняющая действия, аналогичные тем, которые выполняет эксперт в некоторой прикладной предметной области, делая определенные заключения в ходе выдачи советов и консультаций.

Основные области применения ЭС (в порядке уменьшения числа ЭС, используемых в данной области):

- проектирование ЭС;
- медицинский диагноз и консультации по лечению;
- консультации и оказание помощи пользователю по решению задач в различных предметных областях;
- автоматическое программирование, проверка и анализ ПО;
- проектирование сверхбольших интегральных схем. Обучение в различных предметных областях;
- техническая диагностика и выработка рекомендаций по ремонту оборудования;
- планирование в различных предметных областях. Анализ данных в различных предметных областях (в том числе и статистический). Интерпретация геологических данных и выработка рекомендаций по обнаружению полезных ископаемых;
- интерпретация данных и планирование эксперимента в ходе научных исследований в области биологии. Решение задач, связанных с космическими исследованиями;
- обеспечение научных исследований в химии, выработка рекомендаций по синтезу соединений;
- управление проектированием, технологическими процессами и промышленным производством. Анализ и синтез электронных схем. Формирование математических понятий, преобразование математических выражений;
- анализ рисков в политике и экономике.

Структура типовой ЭС представлена на рис. 14.1.

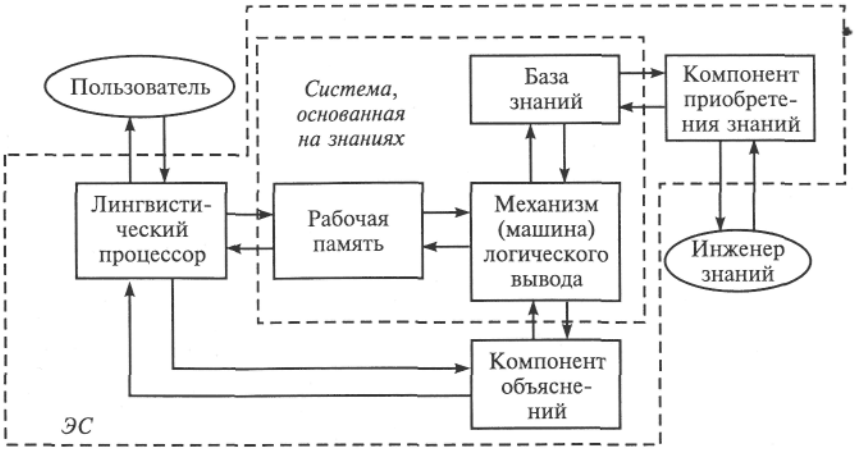


Рис. 14.1. Структура экспертной системы

Дадим краткую характеристику структурных элементов ЭС.

Система, основанная на знаниях, представляет собой программную систему, состоящую из трех элементов: базы знаний, механизма (машины) логического вывода и рабочей памяти.

База данных — часть ЭС (системы, основанной на знаниях), предназначенная для генерации и поддержания динамической модели знаний о предметной области (в качестве возможных моделей знаний могут использоваться рассмотренные в гл. 6 продукционные, сетевые или фреймовые модели).

Механизм (машина) логического вывода — часть ЭС (системы, основанной на знаниях), реализующая анализ поступающей в ЭС и имеющейся в ней информации и формирование (вывод) на ее основе новых заключений (суждений) в ответ на запрос к системе.

Рабочая память — часть ЭС (системы, основанной на знаниях), предназначенная для информационного обеспечения работы механизма логического вывода, прежде всего в части хранения и обработки поступивших (новых) фактов (суждений) и промежуточных результатов логического вывода (подробнее см. подразд. 15.4).

Лингвистический процессор предназначен для обеспечения комфортного интерфейса между конечным пользователем и ЭС. В нем реализуются процедуры морфологического, синтаксического и семантического контроля поступающих в систему запросов и приведение их к виду, «понятному» ЭВМ. При выдаче ответной информации осуществляется обратная операция — заключение «переводится» на ограниченный естественный язык, понятный конечному пользователю. Отметим, что в первых ЭС лингвистический процессор отсутствовал, так как общение с машиной осуществлялось на формальном языке. В дальнейшем (особенно при переходе к ЭВМ пятого поколения) значимость лингвистического процессора в составе ЭС будет возрастать.

Компонент приобретения знаний предназначен для обеспечения работы инженера знаний по поддержанию модели знаний, адекватной реальной предметной области (генерации базы знаний, ее тестирования, пополнения новыми знаниями, исключения неверных (ставших таковыми) знаний и т.п.).

Наличие *компонента объяснений,* обеспечивающего по запросу пользователя выдачу информации о ходе и исходе логического вывода, принципиально отличает ЭС от всех других программных систем. Дело в том, что в большинстве случаев конечному пользователю недостаточно сообщить лишь конечное заключение ЭС, которое он должен (может) использовать в своей профессиональной деятельности. Гораздо большее доверие вызывает у него конечный вывод, подтвержденный понятными промежуточными рассуждениями. Кроме того, с помощью компонента объяснений можно организовать процесс обучения конечных пользователей работе с ЭС. В обучающих ЭС компонент объяснений играет еще более важную роль.

Важным классом систем, основанных на знаниях, является класс ИППП. Структура такого пакета приведена на рис. 14.2.

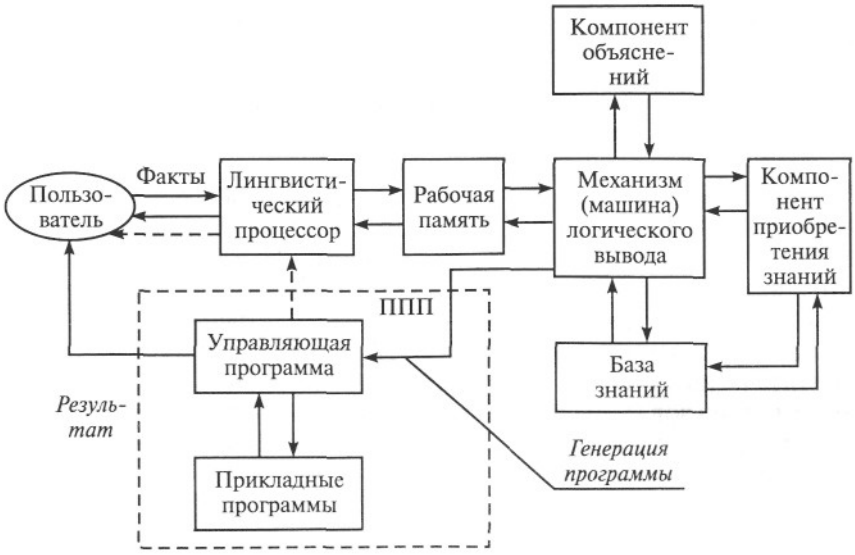


Рис. 14.2. Структура интеллектуального пакета прикладных программ:
→ прямые связи; - - - → обратные связи

ИППП дают возможность конечному пользователю решать прикладные задачи по их описаниям и исходным данным без программирования — генерация («сборка») программы «под задачу» осуществляется автоматически механизмом логического вывода. База знаний в ИППП может строиться по любому из известных эвристических методов (часто используются семантические сети и фреймы), лишь бы настраиваемая машиной логического вывода программа была эффективна для решения поставленной задачи.

14.2. Классификация, этапы и средства разработки экспертных систем

Существует множество признаков, по которым можно (весьма условно) классифицировать ЭС [49]. По степени сложности различают *поверхностные* и *глубинные* ЭС, по степени связанности правил продукционные ЭС делят на *связные* и *малосвязные,* по типу предметной области выделяют *статические, динамические ЭС* и *ЭС реального времени* и т.п. Процесс создания ЭС занимает немало времени, поэтому определенный интерес представляет

классификация ЭС по стадиям разработки (заметим, что аналогичные стадии в своем ЖЦ имеют практически все — достаточно сложные — программные системы):

- демонстрационный прототип (база знаний содержит 10—100 правил);
- исследовательский прототип (200 — 500 правил);
- действующий прототип (500— 1000 правил);
- промышленный образец (1000— 1500 правил);
- коммерческий образец (1500 — 3000 правил).

Масштабы разработки ЭС предопределили создание специальных инструментальных (аппаратных и программных) средств, систематизированное представление которых составляет содержание рис. 14.3.

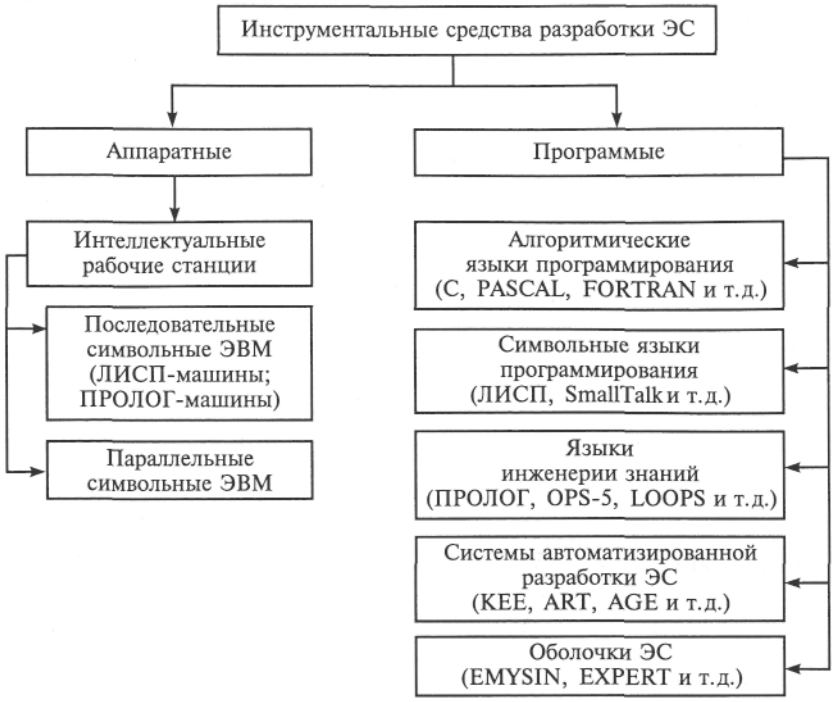


Рис. 14.3. Инструментальные средства разработки экспертных систем

Следует отметить, что первоначально разработка ЭС осуществлялась на традиционных алгоритмических языках программирования с реализацией на универсальных ЭВМ. В дальнейшем были созданы как специализированные аппаратные и программные средства, так и средства автоматизации программирования. Появились и оболочки ЭС, которые по задумке авторов должны были существенно упростить (и удешевить) разработку систем. Однако в полной мере эти надежды не оправдались (как показало дальнейшее развитие прикладных программных средств не только в области ИИ, и не могли оправдаться). Это связано с принципиальной сложностью использования конкретной ЭС (даже весьма эффективной в своей предметной области) для решения совершенно других задач, а именно таким путем создавались первые оболочки ЭС. Еще более проблематичной представляются попытки создания так называемых универсальных оболочек, пригодных для применения «во всех» предметных областях.

При создании ЭС наибольшую трудность представляет разработка совершенной базы знаний, т. е. моделирование знаний экспертов о некоторой предметной области. Разработка любой модели — в том числе и модели знаний — представляет собой полностью неформализуемый процесс, содержащий элементы творчества и строго формальных действий. Условное соотношение «искусства» и «науки» при создании ЭС представлено на рис. 14.4.

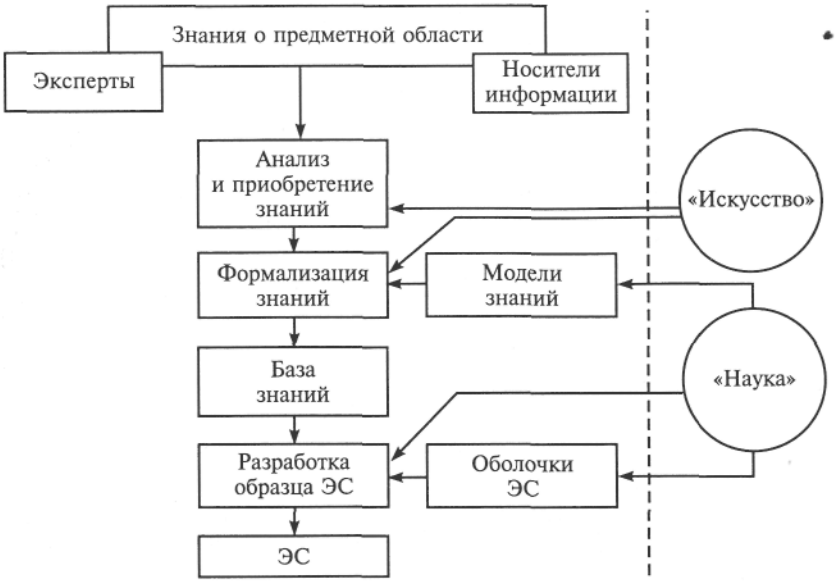


Рис. 14.4. Соотношение формальных и неформальных процедур при разработке экспертной системы

Разработка ЭС включает нескольких этапов [38], основное содержание которых применительно к продукционным системам отражено на рис. 14.5.

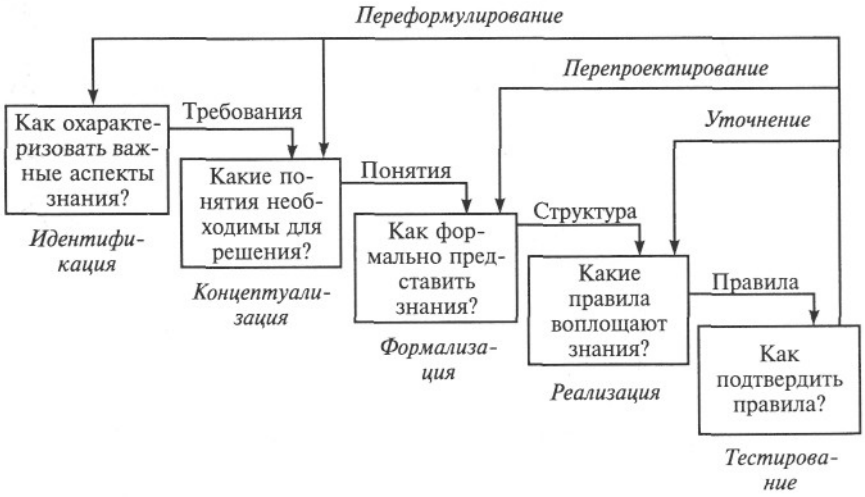


Рис. 14.5. Этапы разработки экспертной системы

Процедуры уточнения, перепроектирования и переформулирования не являются обязательными, они характерны для разработки достаточно сложных ЭС и, как правило, предполагают проведение нескольких итераций. Отметим, что перечисленные этапы работ (идентификация — концептуализация — формализация — реализация — тестирование), как и стадии разработки, являются обязательными при создании любой программной системы.

Очевидно, что разработка ЭС является коллективным трудом, в котором принимают участие различные специалисты. Центральное место в схеме взаимодействия участников создания ЭС занимает инженер знаний (англ *knowledge engineer).* Именно он организует все важнейшие работы и осуществляет их координацию. Ему принадлежит право выбора типовых или — при необходимости и наличии соответствующих ресурсов — заказа новых инструментальных средств разработки ЭС. Он работает с предметными экспертами, генерирует, тестирует, уточняет и пополняет базу знаний и т.д. Направления взаимодействия создателей ЭС (этот процесс иногда называют игрой [38]) представлены на рис. 14.6.

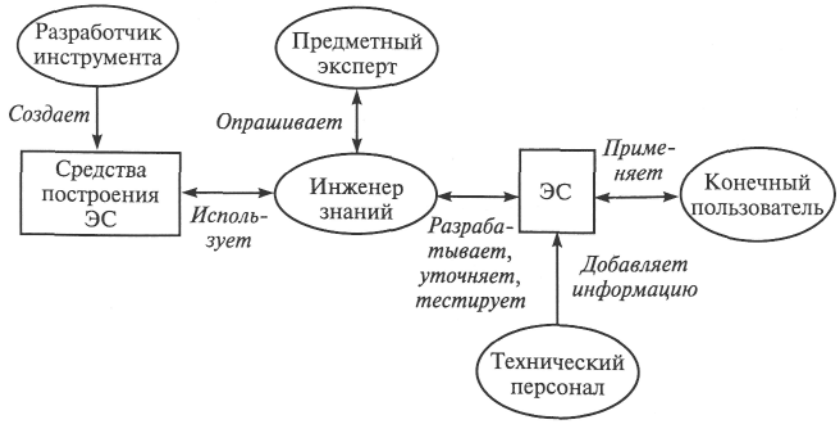


Рис. 14.6. Взаимодействие создателей экспертной системы

Как явствует из вышеизложенного, разработка ЭС — сложный, дорогостоящий и длительный процесс. Последнее обстоятельство иллюстрируется рис. 14.7, на котором приведены условные затраты времени на создание систем для решения проблем различной сложности [38].

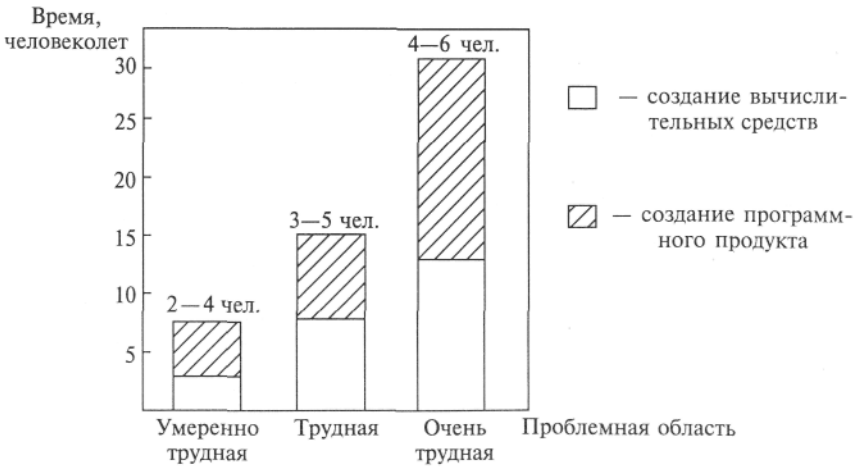


Рис. 14.7. Затраты времени на создание экспертной системы

Существует ряд подходов к оценке того, когда же разработка ЭС является рациональной [21, 22, 26]. На наш взгляд, наиболее конструктивен подход Д.Уотермена, который основан на проверке возможности, оправданности и разумности построения системы.

При этом предлагается считать, что разработка ЭС возможна при совместном выполнении следующих основных условий:

- задача не требует общедоступных знаний;
- решение задачи требует только интеллектуальных действий;
- существуют подлинные (компетентные) эксперты;
- эксперты способны описать свои методы (приемы, уловки и т.п.) решения задачи;
- эксперты единодушны в своих решениях (или, по крайней мере, их мнения «хорошо» согласованы);
- задача понятна и «не слишком» трудна.

Разработка ЭС оправдана, если выполняется хотя бы одно из следующих основных условий:

- получение решения задачи высокорентабельно;
- человеческий опыт решения задачи по различным причинам утрачивается;
- число экспертов в рассматриваемой предметной области мало;
- опыт решения задачи востребован во многих местах;
- опыт нужно применять во враждебных человеку условиях.

Наконец, разработка ЭС разумна, если совместно выполняются следующие основные условия:

- задача требует эвристических решений;

- задача требует оперирования символами;
- задача «не слишком» проста;
- задача представляет практический интерес;
- задача имеет размерность, допускающую реализацию.

При всей условности и субъективности проверки наличия перечисленных обстоятельств можно по-новому взглянуть на причины столь широкой представительности перечня областей применения ЭС.

В заключение напомним о принципиальной важности совершенства базы знаний для эффективности ЭС. Другим важнейшим составным элементом любой системы, основанной на знаниях, в том числе и ЭС, является механизм логического вывода. Обсуждению основ его функционирования для различных моделей представления знаний посвящена следующая глава.

Глава 15. ОСНОВЫ ПОСТРОЕНИЯ И ИСПОЛЬЗОВАНИЯ МЕХАНИЗМОВ ЛОГИЧЕСКОГО ВЫВОДА

15.1. Механизм логического вывода в продукционных системах

Механизм логического вывода — неотъемлемая часть системы, основанной на знаниях (ЭС), реализующая функции вывода (формирования) умозаключений (новых суждений) на основе информации из базы знаний и рабочей памяти.

Как следует из определения, для работы механизма логического вывода необходима как «долговременная» информация, содержащаяся в базе знаний в выбранном при разработке ЭС виде, так и «текущая» оперативная информация, поступающая в рабочую память после обработки в лингвистическом процессоре запроса пользователя. Таким образом, база знаний отражает основные (долговременные) закономерности, присущие предметной области. Запрос пользователя, как правило, связан с появлением каких-либо новых фактов и/или с возникновением потребности в их толковании.

Перед рассмотрением конкретных механизмов логического вывода подчеркнем несколько важных обстоятельств:

- единого механизма логического вывода для произвольных систем, основанных на знаниях (ЭС), не существует;
- механизм логического вывода полностью определяется моделью представления знаний, принятой в данной системе;
- существующие механизмы логического вывода не являются строго фиксированными («узаконенными») для каждого типа систем, основанных на знаниях (ЭС).

Из всех известных механизмов вывода механизм логического вывода является наиболее формализованным (предопределенным). Различают два типа логического вывода:

- прямой вывод (прямая цепочка рассуждений);
- обратный вывод (обратная цепочка рассуждений).

Сущность *прямого логического вывода* в продукционных ЭС состоит в построении цепочки выводов (продукций или правил), связывающих начальные факты с результатом вывода.

В терминах «факты — правила» формирование цепочки вывода заключается в *многократном повторении элементарных шагов* «сопоставить — выполнить».

Рассмотрим следующий пример [54]. В базе знаний некоторой ЭС содержатся три правила, а в рабочей памяти до начала вывода — пять фактов: *B, C, H, G, E* (рис. 15.1). Пусть на вход системы (в рабочей памяти) поступил факт *A*. Механизм вывода «просматривает» левые части правил с целью нахождения таких из них, которые позволяют извлечь новые факты (процедура «сопоставить»). В нашем примере на основе третьего правила выводится факт *D*. Происходит элементарный шаг «выполнить» — данный факт заносится в рабочую память. Процедура «сопоставить» по фактам *C* и *D* выявляет факт *F*. После шага «выполнить» этот факт попадает в рабочую область. По фактам *F* и *B* выводится факт Z, и дальнейший «просмотр» правил БД новой информации не дает.

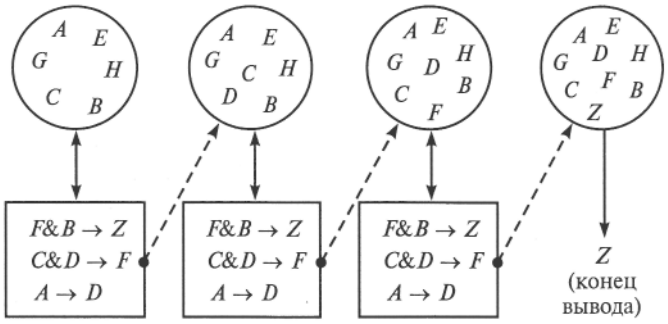


Рис. 15.1. Прямая цепочка рассуждений:
⟷ «сопоставить»; ●- - ► «выполнить»

Таким образом, прямая цепочка рассуждений состоит из следующих фактов: *A — D — F — Z.* Иными словами, из факта *A* на основе имеющихся в базе знаний правил «получен» факт *Z*

Отметим, что, несмотря на очевидную простоту прямого вывода для пользователя ЭС, от которого требуется лишь сообщить системе о вновь поступивших или интересующих его фактах, для базы знаний со значительным числом правил могут возникнуть две проблемы: когда завершить вывод; как обеспечить непротиворечивость правил.

Последнее обстоятельство требует формирования и хранения в ЭС так называемых *метаправил* — правил «работы с правилами». Кроме того, прямая цепочка рассуждений иногда требует значительного времени.

Механизм *обратного вывода* имеет совершенно иной алгоритм. Его идея заключается в проверке справедливости некоторой гипотезы (некоторого суждения, факта), которая выдвигается пользователем и проверяется ЭС. При этом осуществляется проверка истинности не левых, а правых частей продукций, а вопрос формулируется так: «Что нужно, чтобы правая часть данного правила была справедлива и есть ли необходимые суждения в рабочей памяти?». На рис. 15.2 показана работа механизма обратного вывода для того же примера, что для прямой цепочки рассуждений (в предположении, что факт *A* занесен в рабочую память).

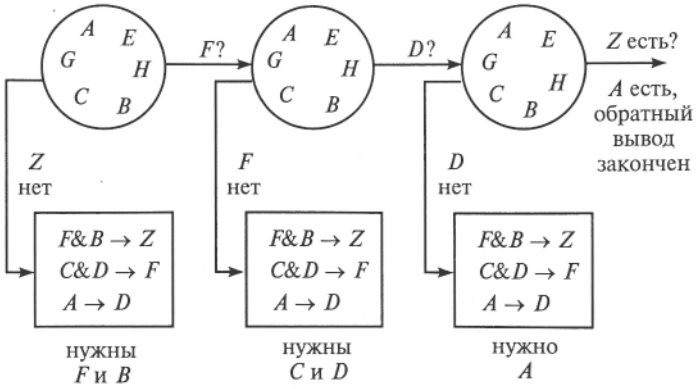


Рис. 15.2. Обратная цепочка рассуждений

При реализации данного механизма пополнения рабочей памяти новыми (выведенными) фактами не производится, а лишь проверяется наличие необходимых суждений на очередном шаге алгоритма. Поскольку непосредственно факта *Z* в рабочей памяти нет, производится анализ правых частей правил до поиска такого правила, которое обосновывает справедливость суждения *Z*. Чтобы факт *Z* был истинным, необходимы факты *F* и *B*. Факт *B* есть, факта *F* нет. Чтобы факт *B* был истинным, необходимы факты *C* и *D*. Факт *C* есть, факта *D* нет. Наконец, чтобы был справедлив факт *D* нужно наличие факта *A,* и так как он в рабочей памяти имеется, обратный вывод закончен. Окончательный результат — на основании имеющихся в ЭС правил и фактов гипотеза о справедливости факта Z подтверждается.

Очевидно, что обратная цепочка рассуждений предъявляет к квалификации пользователя ЭС определенные требования — он должен уметь формулировать «правдоподобные» гипотезы. В

противном случае легко представить весьма непродуктивную работу ЭС, проверяющей и отвергающей одну гипотезу за другой (в качестве примера аналогичной ситуации представим себе врача, ставящего один диагноз за другим и прописывающего пациенту лекарства от самых разных болезней). Платой за выполнение данного требования служит, как правило, сокращение времени реакции ЭС на запрос пользователя.

Для обеспечения уверенности пользователя в получаемых ЭС суждениях после обратного вывода часто прибегают к прямой цепочке рассуждений. Совпадение результатов работы обоих механизмов служит гарантией получения истинного вывода.

Ниже представлен фрагмент блока «Контроль» ЭС, решающей задачи обучения, позволяющий оценивать ответы студентов на зачетах и экзаменах по следующей схеме: обучаемый получает три основных вопроса и отвечает на них.

1. ЕСЛИ по одному из вопросов получена оценка 2, ТО итоговая оценка не может быть выше 3.
2. ЕСЛИ по двум и более вопросам получены оценки 2, ТО итоговая оценка — 2.
3. ЕСЛИ по итогам ответов на основные вопросы обучаемый набрал 14 баллов, ТО необходимо задать дополнительный вопрос.

ЕСЛИ ответ на дополнительный вопрос оценивается 5, ТО итоговая оценка — 5.
ЕСЛИ ответ на дополнительный вопрос оценивается 4, ТО итоговая оценка — 4.
ЕСЛИ ответ на дополнительный вопрос оценивается 3, ТО итоговая оценка — 4.

4. ЕСЛИ по итогам ответов на основные вопросы обучаемый набрал 8 баллов, ТО необходимо задать дополнительный вопрос.

ЕСЛИ ответ на дополнительный вопрос оценивается 5, ТО итоговая оценка — 3.
ЕСЛИ ответ на дополнительный вопрос оценивается 4, ТО итоговая оценка — 3.
ЕСЛИ ответ на дополнительный вопрос оценивается 3, ТО итоговая оценка — 3.

5. ЕСЛИ по одному из вопросов получена оценка 3, ТО итоговая оценка не может быть выше 4.
6. ЕСЛИ по итогам ответов на основные вопросы набрано 9—10 баллов, ТО итоговая оценка — 3.
7. ЕСЛИ по итогам ответов на основные вопросы набрано 11—13 баллов, ТО итоговая оценка — 4.

Преподаватель оценивает каждый ответ по четырехбалльной шкале. ЭС либо сразу рекомендует выставить итоговую оценку, либо «советует» задать дополнительный вопрос и уже по результату ответа на него рекомендует итоговую оценку. В ЭС хранятся знания о существующих в институте правилах оценивания уровня подготовленности обучаемых. Так, например, если на экзамене некий студент на «отлично» ответит два вопроса билета, а третий будет оценен на «хорошо», рекомендуется задать ему дополнительный вопрос. Если на него дается отличный ответ, общая оценка — «отлично»; хороший или удовлетворительный ответ — «хорошо»; неудовлетворительный ответ — «удовлетворительно».

15.2. Понятие о механизме логического вывода в сетевых системах

Механизм логического вывода в сетевых системах основан на использовании двух ведущих принципов: наследования свойств; сопоставления по совпадению.

Первый принцип, в свою очередь, базируется на учете важнейших связей, отражаемых в семантической сети. К таким связям относятся:

- связь «есть», «является» (англ. IS-A);
- связи «имеет часть», «является частью» (англ. HAS-PART, PART-OF).

Последовательно переходя от одного узла сети к другому по направлению соответствующих связей, можно выявить (извлечь) новую информацию, характеризующую тот или иной узел. На рис. 15.3, *а* показан малый фрагмент некоторой семантической сети и обозначена так называемая ветвь наследования свойств. Из этого фрагмента можно вывести заключения типа «Иван — человек», «у Ивана есть голова», «мужчина имеет голову» и т.п.

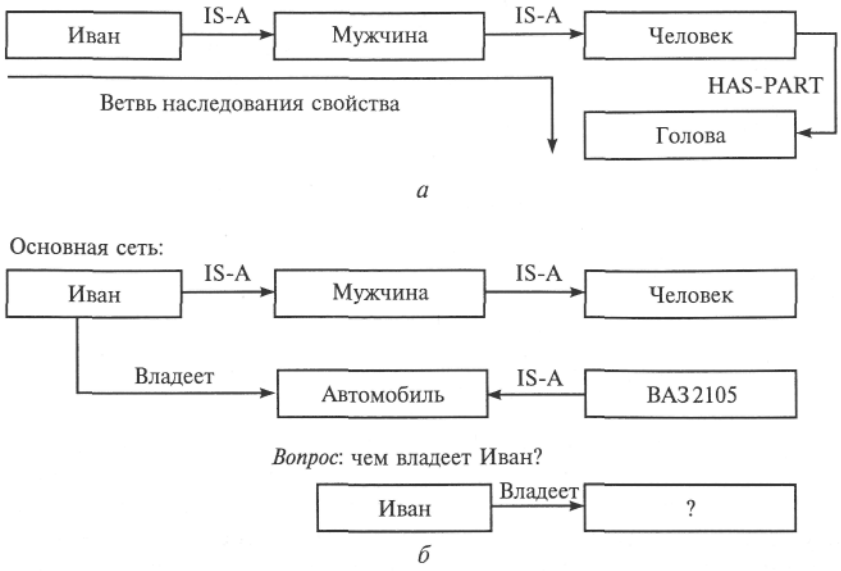


Рис. 15.3. Механизм логического вывода в семантической сети:
а — фрагмент семантической сети; *б* — отношение владения

Принцип сопоставления по совпадению основан на представлении вопроса к системе в виде фрагмента семантической сети с использованием тех же названий сущностей (узлов) и связей, что и в основной сети, и реализации процедуры «наложения» вопроса на сеть и поиска такого его положения, которое соответствует ответу на вопрос. На рис. 15.3, *б* помимо уже известной связи «есть» представлено отношение владения (связь «владеет»). Вопрос: «Чем владеет Иван?» — формализуется с помощью узла «Иван» и отношения «владеет». Далее в простейшем случае осуществляется перебор узлов сети, имеющих имя «Иван» (если они имеются), и поиск такого из них, который имеет связь «владеет». Далее может быть задействован принцип наследования свойств. Ответами на поставленный в примере вопрос будут суждения «Иван владеет автомобилем» и «Иван владеет (автомобилем) ВАЗ 2105». Понятно, что в практике использования ЭС такого типа приходится реализовывать значительно более сложную процедуру поиска, включающую элементы семантического анализа.

15.3. Понятие о механизме логического вывода во фреймовых системах

Как уже отмечалось в подразд. 15.2, обычно фреймовая модель знаний имеет сложную иерархическую структуру, отражающую реальные объекты (понятия) и отношения (связи) некоторой предметной области. Механизм логического вывода в таких ЭС основан на обмене значениями между одноименными слотами различных фреймов и выполнении присоединенных процедур «если — добавлено», «если — удалено» и «если — нужно». Условная схема таких действий для простейшего варианта представлена на рис. 15.4.

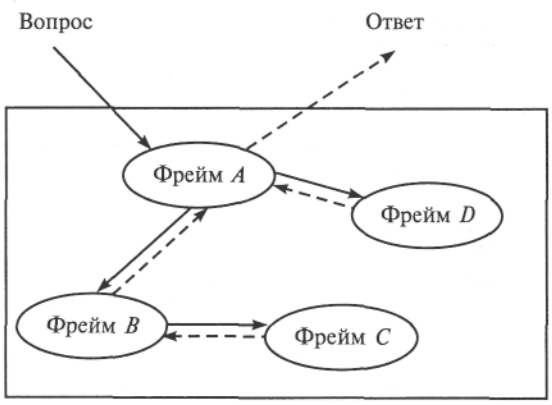


Рис. 15.4. Механизм вывода во фреймовой модели:
——→ сообщение; - - - → значение

Запрос к ЭС в виде сообщения поступает в старший по иерархии фрейм (на рисунке — фрейм *A)*. Если ответа на запрос нет ни в одном из слотов этого фрейма или их совокупности, соответствующие сообщения (запросы) передаются во все фреймы, где имеются слот (слоты), имена которых содержатся в запросе или необходимы для поиска ответа на него (фреймы *B* и *D)*. Если в них содержится искомый ответ, значение соответствующего слота передается в старший по иерархии фрейм (из фрейма *D* во фрейм *A)*. Если для этого нужна дополнительная информация, предварительно передается сообщение (из фрейма *B* во фрейм С) и получается значение (из фрейма С во фрейм *B)*. Значения, передаваемые в ответ на сообщения, либо непосредственно содержатся в соответствующих слотах фреймов, либо определяются как результат выполнения присоединенных процедур.

В современных фреймовых системах, как правило, для пользователя реализована возможность формулировать запросы на языке, близком к реальному. Интерфейсная программа (лингвистический процессор) должна «уметь» по результатам анализа запроса определять, в какой (какие) слот (слоты) необходимо поместить значение (значения) для инициализации автоматической процедуры поиска ответа.

Рассмотрим более конкретный пример, иллюстрирующий работу фреймовой ЭС, используемой в подразделении, организующем научно-исследовательскую работу в некотором учреждении. На рис. 15.5 представлена иерархия справочной информации об отчете по научно-исследовательской работе (о понятии, узле «отчет по научно-исследовательской работе»).

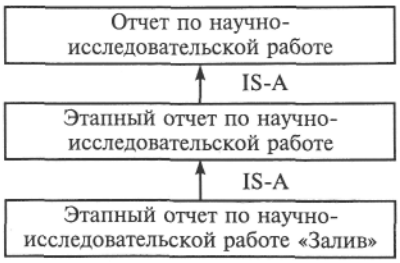


Рис. 15.5. Иерархия понятия «Отчет по научно-исследовательской работе»

Рис. 15.6 содержит структуры понятий «Отчет по научно-исследовательской работе» и «Этапный отчет по научно-исследовательской работе», а рис. 15.7 — структуру понятия «Этапный отчет по научно-исследовательской работе «Залив» со значениями некоторых слотов и присоединенными процедурами.

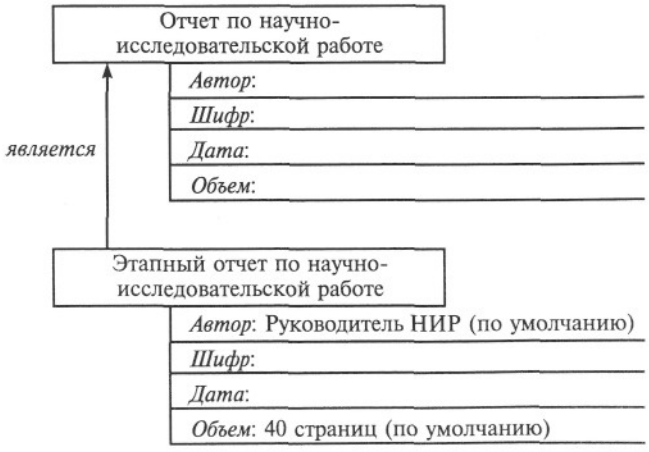


Рис. 15.6. Структура понятий «Отчет по научно-исследовательской работе» и «Этапный отчет по научно-исследовательской работе»

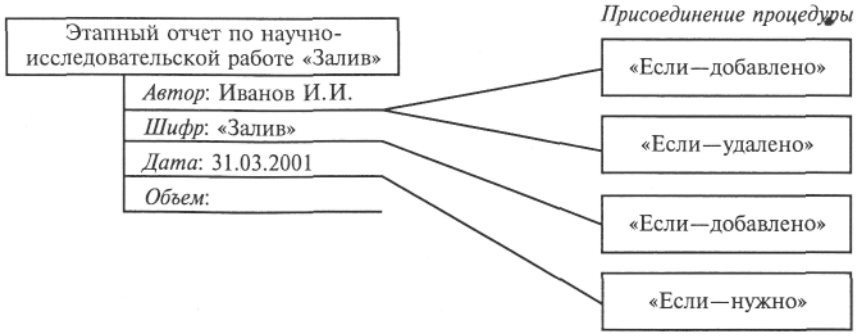


Рис. 15.7. Структура понятия «Этапный отчет по научно-исследовательской работе «Залив»

Фреймовая система функционирует следующим образом. Пусть в ЭС поступил запрос от полномочного пользователя: «Необходима информация о ходе выполнения научно-исследовательской работы «Залив» (напомним, что, как правило, язык исходного запроса близок к естественному). Информация проходит через лингвистический процессор, анализируется и в виде значения «Залив» вносится в слот *Шифр* узла «Этапный отчет по научно-исследовательской работе «Залив». Далее начинают работать присоединенные процедуры:

- процедура «Если—добавлено», связанная со слотом *Шифр,* выполняется, поскольку в слот было введено некоторое значение. Эта процедура осуществляет поиск сведений о руководителе научно-исследовательской работы «Залив» (в нашем примере — И.И.Иванов) и вписывает это имя в слот *Автор* узла «Этапный отчет по научно-исследовательской работе «Залив»;
- процедура «Если—добавлено», связанная со слотом *Автор,* выполняется, так как в слот было вписано значение. Эта процедура начинает составлять сообщение, чтобы отправить его И. И. Иванову, но обнаруживает, что отсутствует значение слота *Дата;*
- процедура «Если—добавлено», просматривая *слот Дата* и найдя его пустым, активизирует процедуру «Если—нужно», связанную с этим слотом. Процедура найдет текущую дату, используя календарь ЭС, выберет ближайшую к ней (но бльшую) дату представления отчета (в нашем примере — 31.03.2003) и впишет ее в слот *Дата;*
- процедура «Если—добавлено», связанная со слотом *Автор,* найдет, что отсутствует еще одно значение, необходимое для формирования выходного сообщения, а именно значение слота *Объем.* Данный слот (узла «Этапный отчет по научно-исследовательской работе «Залив») не имеет присоединенных процедур, поэтому приходится брать значение по умолчанию из одноименного слота общей концепции «Этапного отчета по научно-исследовательской работе» (в нашем примере — 40 с.).

Теперь ЭС может сформировать выходное сообщение типа: «Этапный отчет по научно-исследовательской работе «Залив» должен быть представлен И. И. Ивановым к 31 марта 2003 г. Предполагаемый объем отчета — 40 с.» и/или «И.И.Иванов! Представьте этапный отчет по научно-исследовательской работе «Залив» объемом не более 40 с. к 31 марта 2003 г.».

Если в какой-либо момент значение слота *Автор* (в нашем примере — И. И. Иванов) будет удалено, то сработает процедура «Если — удалено» и система автоматически отправит И.И.Иванову уведомление о том, что отчет не требуется.

15.4. Механизм логического вывода в диагностических системах байесовского типа

Диагностические ЭС широко применяются в различных областях человеческой деятельности (медицине, технике, экономике и др.). Как правило, в них используются продукционные модели знаний о предметной области. Однако, если имеется возможность использования в правилах статистических данных о понятиях и связях между ними, весьма целесообразно применить известную теорему Байеса для пересчета апостериорных вероятностей по результатам проверки наличия тех или иных симптомов.

Применительно к техническим диагностическим системам используется следующая схема формализации:

- объект имеет множество возможных неисправностей

$$S_1,\ S_2,\ \ldots,\ S_m;$$

- каждой неисправности приписывается априорная вероятность

$$P(S_1), P(S_2),\ \ldots,\ P(S_m);\ \sum_{i=1}^{m} P(S_i) = 1;$$

- каждая неисправность проявляется через симптомы

$$C_1,\ C_2,\ \ldots,\ C_n,$$

причем каждая неисправность характеризуется «своими» симптомами из «общего» списка:

$$P(C_j/S_i),\ i = 1,\ 2,\ \ldots,\ m;\ j = 1,\ 2,\ \ldots,\ n;$$

- известны условные вероятности проявления симптомов при каждой неисправности.

Тогда можно определить апостериорные вероятности наличия неисправности при данном симптоме

$$P(S_i / C_j) = \frac{P(C_j / S_i)P(S_i)}{P(C_j)} = \frac{P(C_j / S_i)P(S_i)}{\sum_{i=1}^{m} P(C_j / S_i)P(S_i)},$$

причем при расчете апостериорной вероятности учитывается, наблюдался при испытании данный симптом или нет.

Зная перечисленные вероятности, легко реализовать процедуру проверки наиболее вероятных симптомов, причем проверка очередного симптома должна сопровождаться пересчетом значений всех апостериорных вероятностей. Для получения априорных и условных вероятностей необходимо обработать статистические» данные (при их наличии) или получить и обработать экспертную информацию.

На рис. 15.8 представлена иллюстрация описанного подхода. На рис. 15.8, *а* показаны исходные априорные вероятности наличия неисправностей. Как правило, задается некоторый уровень вероятности P_{mp}, превышение которого свидетельствует о необходимости проверки именно тех неисправностей, для которых и наблюдается превышение (в нашем примере — S_i). Далее проверяется наличие того симптома, для которого вероятность его проявления при *i*-й неисправности наибольшая (например, симптома C_1 на рис. 15.8, *б*).

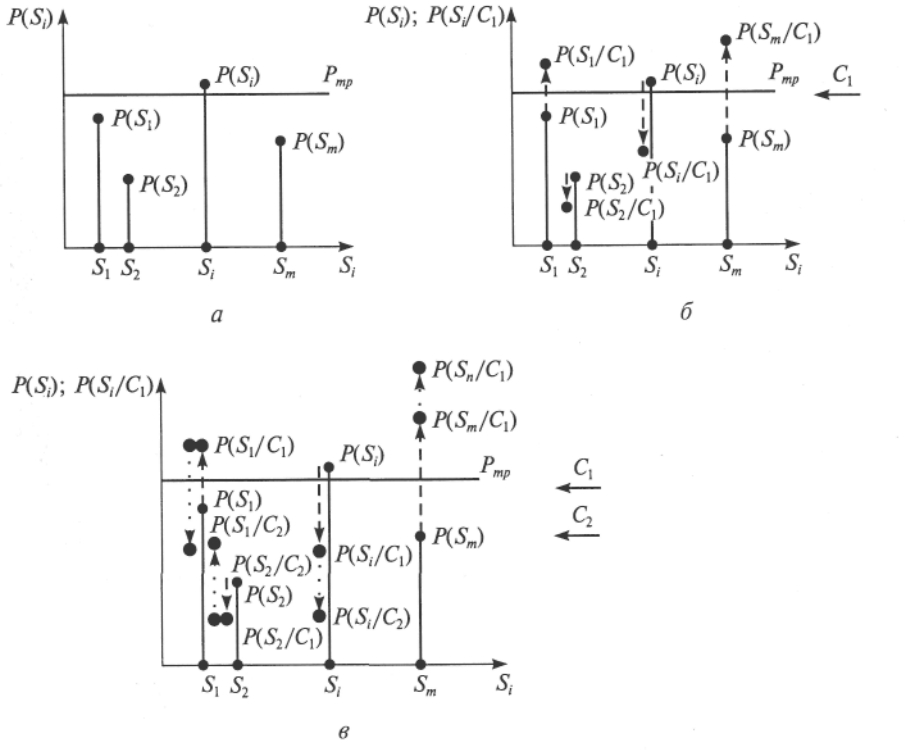


Рис. 15.8. Алгоритм работы диагностической ЭС:
а — исходные вероятности наличия неисправностей; *б* — проверка симптома C_1;
в — проверка симптома C_2; ⟶ вероятность; - - ➤ условная вероятность;
· · · ➤ переходный уровень

По результатам проверки пересчитываются все апостериорные вероятности и выявляются те из них, которые превышают заданный уровень. По ним определяется очередной проверяемый симптом (на рис. 15.8, *в* — симптом C_2) и т.д. Заметим, что в результате пересчета апостериорная вероятность той или иной неисправности может как увеличиться, так и уменьшиться. После нескольких шагов данный алгоритм приводит к тому, что ЭС некоторые неисправности, апостериорные вероятности которых стали очень малыми, отбрасывает (перестает учитывать), а другие предлагает исправить.

Рассмотрим конкретный пример — фрагмент ЭС диагностического типа, предназначенной для поиска неисправности в автомобиле при следующих исходных данных:

- автомобиль может иметь четыре неисправности:

S_1 — неисправна аккумуляторная батарея;
S_2 — отсутствует топливо;
S_3 — «отсырело» зажигание;
S_4 — замаслены свечи;

- симптомами неисправностей являются:

C_1 — фары не горят;
C_2 — указатель топлива на нуле;
C_3 — автомобиль не заводится;
C_4 — стартер проворачивается;
C_5 — двигатель работает неустойчиво, «чихает»;

- значения априорных вероятностей:

$$P(S_1) = 0{,}4;\ P(S_2) = 0{,}2;\ P(S_3) = 0{,}3;\ P(S_4) = 0{,}1;$$

- значения условных вероятностей проявлений симптомов при наличии неисправностей приведены в табл. 15.1. Знаком «+» обозначены вероятности $P(C_j/S_i)$, знаком «-» — вероятности $P(_j/S_i)$.

Таблица 15.1

Значения условных вероятностей проявления симптомов при наличии неисправностей

S_i \ C_j	C_1		C_2		C_3		C_4		C_5	
	+	–	+	–	+	–	+	–	+	–
S_1	1,0	0,0	0,9	0,1	1,0	0,0	0,0	1,0	0,1	0,9
S_2	0,1	0,9	1,0	0,0	1,0	0,0	0,9	0,1	0,1	0,9
S_3	0,1	0,9	0,1	0,9	0,7	0,3	0,9	0,1	0,1	0,9
S_4	0,1	0,9	0,1	0,9	0,8	0,2	0,9	0,1	1,0	0,0

Реализация описанного выше алгоритма для последовательности симптомов «фары горят» — «указатель топлива не на нуле» — «стартер проворачивается» — «автомобиль заводится» — «двигатель «чихает» приведет к следующему заключению ЭС: «Просушите зажигание, проверьте свечи».

Другая последовательность симптомов «фары не горят» — «автомобиль не заводится» — «стартер не проворачивается» — «указатель топлива не на нуле» — «двигатель не чихает» приведет ЭС к рекомендации: «Замените аккумуляторную батарею». Если при проверке симптомов окажется, что «фары горят», «указатель топлива на нуле», «автомобиль не заводится», «стартер проворачивается», «двигатель не чихает», рекомендация ЭС, естественно, такова: «Залейте бензин».

Широкое распространение диагностических ЭС в различных областях деятельности определяется рядом обстоятельств.

Во-первых, возможностью обеспечения близости априорных и условных вероятностей, которые используются в алгоритме, к «истинным» значениям. Как правило, при грамотном учете опыта работы специалистов по устранению соответствующих неисправностей хорошие оценки названных вероятностей могут быть получены по результатам обработки статистических данных.

Во-вторых, сравнительной простотой обеспечения диалога пользователя с системой на языке, близком к естественному, поскольку промежуточные и итоговые заключения ЭС, основанные на формальных шагах алгоритма работы, легко интерпретируются в понятные всем рекомендации.

В-третьих, возможностью выдачи пользователю (как правило, по запросу) промежуточных результатов диагностики неисправности, т. е. пояснения рекомендаций ЭС, что в подавляющем большинстве случаев облегчает их восприятие.

Наконец, возможностью постоянного учета текущего опыта пользователей и простотой корректировки (при необходимости) модели знаний о предметной области.

В заключение раздела отметим, что в учебнике рассмотрены лишь методологические основы построения и использования СИИ. Практика совершенствования информационных технологий представляет все новые направления применения интеллектуальных средств.

Так, например, наряду с ИППП появились так называемые интеллектуальные БД [27], в которых используются достижения теории искусственного интеллекта как для организации хранения информации о предметной области, так и для удовлетворения информационных потребностей пользователей.

Другим примером может служить разработанная специалистами Института человеческого и машинного познания при университете Западной Флориды (США) технология хранения и представления пользователям информации, получившая название *C Map* (англ. *concept map* — карта понятий), являющаяся одним из вариантов применения семантических сетей. С помощью этой технологии можно осваивать большие объемы сложно структурированного материала, решая различные задачи (в том числе и задачи обучения специалистов).

Еще одним примером является известная концепция «интеллектуального дома (жилища)», призванная рационально использовать средства искусственного интеллекта при всестороннем обеспечении управления бытовыми системами (начиная от регулирования подачи электроэнергии и воды и заканчивая включением/ выключением микроволновой печи или телевизора в заданное время). Существует множество подобных примеров, подтверждающих главный вывод: магистральным путем современной автоматизации профессиональной деятельности людей является ее интеллектуализация.

СПИСОК ЛИТЕРАТУРЫ

1. Автоматизированные информационные технологии в экономике / Под ред. Г.А.Титоренко. — М.: ЮНИТИ, 2002.

2. *Айвазян С.А., Мхитарян В. С.* Прикладная статистика. Основы эконометрики. — М.: ЮНИТИ, 2001.

3. *Арсеньев Ю.И., Шелобаев С. И., Давыдкова Т.Ю.* Интегрированные интеллектуальные системы принятия решений. — М.: ЮНИТИ, 2002.

4. *Бабешко Л. О.* Коллокационные модели прогнозирования в финансовой среде. — М.: Экзамен, 2001.

5. *Балдин К. В., Быстров О. Ф., Мальцев А. В.* Теоретические основы моделирования сложных систем. Компьютерные методы и средства обучения моделированию. — М.: РДЛ, 1995.

6. *Балдин К. В.* Моделирование жизненного цикла сложных систем: В 2 ч. — М.: РДЛ, 2000.

7. *Балдин К. В., Уткин В. Б.* Теоретические основы автоматизации управленческой деятельности в экономике. — Воронеж: МОДЭК, 2003.

8. *Балдин К. В., Быстров О. Ф., Соколов М. М.* Экономические и информационно-аналитические основы управления инвестиционными проектами: — Воронеж: МОДЭК, 2002.

9. *Благотатских В. А., Енгибарян М.А., Ковалевская Е.В.* Экономика, разработка и использование программного обеспечения. — М.: Финансы и статистика, 1998.

10. *Гейн К., Сарсон Т.* Системный структурный анализ: средства и методы. — М.: Эйтекс, 1992.

11. *Гилберт С.* Самоучитель Visual C++ 6 в примерах. — М.: Диасофт, 2002.

12. *Горчаков А.А., Орлова Н.В.* Компьютерные экономико-математические модели. — М.: Компьютер, ЮНИТИ, 1995.

13. ГОСТ 19.101-77, 19.002-80, 19.003-80: Сб. — М: Изд-во стандартов, 1984.

14. ГОСТ 24.003-84, 24.101-80, 24.103-84, 24.104-85, 24.101 — 85, 24.205-80, 24.301-80, 24.302-80, 24.303-80, 24.304-82, 34.003-90, 34.602 — 89. АСУ. Основные положения. Термины и определения. Общие требования. Техническое задание на АСУ. Условные обозначения. — М.: Изд-во стандартов, 1988.

15. *Дейт К.* Введение в системы баз данных. — М.: Диалектика, 1998.

16. *Додж М., Стинсон К.* Эффективная работа с Microsoft Excel 2000. — СПб.: Питер, 2002.

17. *Дуброва Т.А.* Статистические методы прогнозирования. — М.: ЮНИТИ, 2002.

18. *Жак С. В.* Математические модели менеджмента и маркетинга. — Ростов н/Д: ЛаПО, 1997.

19. *Завгородний В.Н.* Комплексная защита информации в компьютерных системах. — М.: Финансы и статистика, 2001.

20. *Зегжда Д. П., Ивашко А. М.* Основы безопасности информационных систем. — М.: Горячая линия — Телеком, 2000.

21. Интеллектуальные САПР технологических процессов в радиоэлектронике / Под ред. В.Н.Ильина. — М.: Радио и связь, 1991.

22. Искусственный интеллект: В 3 кн. Кн.1. Системы общения и экспертные системы / Под ред. Э.В.Попова. — М.: Радио и связь, 1990.

23. *Калянов Г.* Российский рынок CASE-средств // НС WEEK/RE. — 1998. — № 23.

24. *Калянов Г.* CASE-структурный системный анализ (автоматизация и применение). — М.: Лори, 1996.

25. *Касперский Е.* Antiviral Toolkit Pro: Энциклопедия компьютерных вирусов. — М.: Лаборатория Касперского, 1999.

26. Компьютерные информационные системы управленческой деятельности / Под ред. Г.А.Титоренко. — М.: Экономическое оборудование, 1993.

27. *Корнеев В. В., Гарев А. Ф., Васютин С. В.* Базы данных. Интеллектуальная обработка информации. — М.: Нолидж, 2000.

28. *Котов С. А.* Нормирование жизненного цикла программной продукции. — М.: ЮНИТИ, 2002.

29. *Краснощеков П. С., Петров А. А.* Принципы построения моделей. — М.: ФАЗИС: ВЦ РАН, 2000.

30. *Кремер Н. Ш., Пудко Б.А., Тришин И. М.* Исследование операций в экономике. — М.: ЮНИТИ, 1997.

31. *Ловцов Д. А., Сергеев Н.А.* Управление безопасностью эргосистем. — М.: РДЛ, 2000.

32. Локальные вычислительные сети: принципы построения, архитектура, коммутационные средства / Под ред. С.В.Назарова. — М.: Финансы и статистика, 1994.

33. *Лукацкий А.В.* Обнаружение атак. — СПб.: БХВ — Санкт-Петербург, 2001.

34. *Медведовский И. Д., Семьянов П. В., Леонов Д. Г.* Атака на *Internet.* — М.: ДМК, 1999.

35. Моделирование производственно-инвестиционной деятельности фирмы / Под ред. Г.В.Виноградова. — М.: ЮНИТИ, 2002.

36. *Мыльник В. В., Китаренко Б. П., Воложенко В. А.* Системы управления. — М.: Экономика и

финансы, 2002.

37. *Нейлор К.* Как построить свою экспертную систему: Пер. с англ. — М.: Энергоатомиздат, 1991.

38. *Олифер В. Г., Олифер Н.А.* Компьютерные сети: Принципы, технологии, протоколы. — СПб.: Питер, 1999.

39. *Першиков В. И., Савинков В. М.* Толковый словарь по информатике. — М.: Финансы и статистика, 1991.

40. *Петров А. А., Поспелов И. Г., Шананин А.А.* Опыт математического моделирования экономики. — М.: Энергоатомиздат, 1996.

41. *Попов Э.В., Федоров С. К., Казаков Е. Ф.* Статистические и динамические экспертные системы. — М.: Финансы и статистика, 2000.

42. *Райордан Р.* Основы реляционных баз данных. — М.: Русская редакция, 2001.

43. *Саймино Д.* Сети Интранет: Внутреннее движение: Пер. с англ. — М.: ООО «Бук Медиа Паблишер», 1997.

44. *Саукап Р.* Проектирование реляционных систем баз данных. — М.: Русская редакция, 1998.

45. Системы управления базами данных и знаниями / Под ред. А. Н. Наумова. — М.: Финансы и статистика, 1998.

46. *Спортак М., Паппас Ф.* Компьютерные сети и сетевые технологии. — Киев: ООО «ТИД «ДС», 2002.

47. *Спортак М.* Компьютерные сети: В 2 кн. — М.: Диасофт, 2002.

48. Теория и практика обеспечения информационной безопасности / Под ред. П.Д. Зегжды. — М.: Яхтсмен, 1996.

49. Толковый словарь по искусственному интеллекту. — М.: Радио и связь, 1996.

50. *Трояновский В. М.* Элементы математического моделирования в макроэкономике. — М.: РДЛ, 2001.

51. *Тыугу Э.Х.* Концептуальное программирование. — М.: Наука, 1994.

52. *Уотшем Т.Дж., Паррамоу К.* Количественные методы в финансах: Пер. с англ. — М.: ЮНИТИ, 1999.

53. *Уткин В. Б.* Основы автоматизации профессиональной деятельности. — М.: РДЛ, 2001.

54. *Уткин В. Б., Сазанович А.Н., Мдичарадзе В. Г.* Информатика. — М.: РДЛ, 1995.

55. *Федосеев В.В.* Курс лекций по экономико-математическому моделированию. — М.: Экономическое образование, 1996.

56. *Федосеев В.В.* Экономико-математические методы и модели в маркетинге. — М.: Финстатинформ, 1996.

57. *Фигурнов В. Э.* IBM PC для пользователя. — М.: Финансы и статистика, 1999.

58. *Фомин Г. П.* Системы и модели массового обслуживания в коммерческой деятельности. — М.: Финансы и статистика, 2000.

59. *Харитонова И. А., Михеева В. Д.* Microsoft Access 2000. — СПб.: БХВ — Санкт-Петербург, 2000.

60. Экономико-математические методы и прикладные модели / Под ред. В.В. Федосеева. — М.: ЮНИТИ, 2001.

61. Экономическая информатика и вычислительная техника / Под ред. В.П.Косарева, А.Ю.Королева. — М.: Финансы и статистика, 1996.

62. MegaPlus: Электроника, компьютеры, связь. — 1999. — № 5.

63. Oracle8: Энциклопедия пользователя: Пер. с англ. — Киев: Диа-Софт, 1998.

Содержание