В.А. Лобанова
О.А. Воронина

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ЭЛЕКТРОННЫХ СРЕДСТВ

ОРЛОВСКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
имени И.С. Тургенева

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

В.А. Лобанова, О.А. Воронина

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ЭЛЕКТРОННЫХ СРЕДСТВ

Орёл
ОГУ имени И.С. Тургенева
2020

УДК 621.396.6 (075.8)
ББК 32я73
Л68

Рецензенты:

доктор технических наук, профессор
кафедры электроники, радиотехники и систем связи
федерального государственного бюджетного
образовательного учреждения высшего образования
«Орловский государственный университет имени И.С. Тургенева»
С.Л. Косчинский,

доктор технических наук, профессор, академик РАЕН
заместитель начальника Управления по Орловской области филиала ФГУП
«Радиочастотный центр ЦФО» в Центральном федеральном округе
А.П. Фисун

Лобанова, В.А.

Л68 Информационные технологии проектирования электронных средств: учебное пособие / В.А. Лобанова, О.А. Воронина. – Орел: ОГУ имени И.С. Тургенева, 2020. – 238 с.

ISBN 978-5-9929-0867-1

В учебном пособии рассмотрены современные подходы к проектированию электронных средств, CASE-средства, теоретические основы, математические методы и модели автоматизированного проектирования электронных средств, а также техническое, программное и интеллектуальное обеспечение САПР электронных средств.

Предназначено студентам, обучающимся по укрупненной группе направлений подготовки 11.00.00 «Электроника, радиотехника и системы связи», при изучении дисциплин «Информационные технологии проектирования электронных средств», «Современные технологии проектирования электронных средств».

УДК 621.396.6 (075.8)
ББК 32я73

ISBN 978-5-9929-0867-1

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня Россия стоит перед исторической необходимостью перехода от индустриального общества на принципиально новый уровень общественного и экономического развития, определяемого жесткими требованиями современной научно-технической революции. Речь идет о формировании информационного общества и информационной экономики, которые в передовых странах уже получили определенное развитие. Предстоит переосмыслить многие устоявшиеся представления и выработать концепцию построения информационного общества с учетом сложившейся в стране ситуации и принимая во внимание, что стратегия развития экономики и общества неотделима от информатизации.

В информационном обществе, материальной базой которого является информационная экономика, акцент значимости смещается на информационный ресурс, представляющий собой знания, накопленные людьми для социального использования в обществе. Эти знания зафиксированы и материализованы в виде документов, баз данных, баз знаний, алгоритмов, компьютерных программ, произведений литературы, науки, искусства. Информационные ресурсы рассматриваются как стратегические ресурсы страны, региона, организации.

Для каждой страны переход в новую эпоху экономического развития, в основе которой лежит использование многообразных информационных ресурсов, определяется степенью информатизации ее экономики и общества в целом.

Ключевая роль в современной инфраструктуре информатизации принадлежит системам коммуникаций и вычислительным сетям, в которых сосредоточены новейшие средства вычислительной техники, информатики, связи, а также самые прогрессивные информационные технологии. Именно они обеспечивают пользователям широкий набор информационно-вычислительных услуг с доступом к локальным и удаленным машинным ресурсам, технологиям и базам данных. Решение этих задач немыслимо без специалистов, владеющих современными информационными технологиями проектирования технических и программных средств.

Это диктовалось и диктуется постоянным ростом сложности и трудоемкости задач, решение которых возлагается на ЭВМ в различных сферах применения.

Объем и сложность современных технических систем решающим образом зависит от умения проектировщиков предварить их соз-

дание описанием всего комплекса проблем, связанных с их дальнейшей эксплуатацией. Особое внимание уделяется начальному этапу проектирования, который опирается на методологию структурного анализа (SADT-методологию).

Наряду с техническим совершенствованием средств электронной техники развиваются методы и приемы программирования вычислений, высшей ступенью которых является автоматическое программирование, требующее минимальных затрат труда математиков-программистов. Большое развитие и применение получили алгоритмические языки, существенно упрощающие процесс подготовки задач к решению на ЭВМ. С их появлением резко сократились штаты «чистых» программистов, поскольку составление программ на этих языках стало под силу самим пользователям.

Системное проектирование – это процесс, который включает в себя формулировку требований к системе и определение ограничений, влияющих на ее функционирование, разложение системы на подсистемы, выделение на каждом уровне разложения системных компонент и описание связей между ними.

При строгом математическом описании или при определённых установленных зависимостях между входами и выходами системы проектирование электронных средств не вызывает принципиальных трудностей.

Новые компьютерные технологии базируются на современных информационных, языковых и программных средствах, которые в свою очередь основываются на различных математических моделях и алгоритмах.

Информационные средства – это прежде всего базы данных (БД), системы управления базами данных (СУБД) и пользовательские интерфейсы.

Процесс создания новых электронных средств и модернизации существующих многогранен. В течение последних десятилетий наблюдается резкое увеличение производства электронной аппаратуры, повышается её «интеллектуальность». В этих условиях использование современных информационных технологий при создании новых радиоэлектронных и электронно-вычислительных средств служит гарантией качества, надежности, экономичности.

Кроме методологий проектирования, современному специалисту – разработчику радиоэлектронных или электронных средств – необходимо знание математического аппарата, позволяющего формализо-

вать проектные процедуры с тем, чтобы создать или использовать имеющееся программное обеспечение, автоматизирующее процесс проектирования.

Специалист должен в достаточной степени знать и уметь работать с современными пакетами прикладных программ как офисного направления, так и специального (типа P-CAD, MathCAD, Matlab, Workbench, LabView, OrCAD, P-Space и др.).

Настоящее пособие состоит из четырех разделов, в каждом из которых представлены теоретические сведения, примеры, контрольные вопросы и задания для самостоятельной работы, способствующие получению базовых знаний в области информационных технологий проектирования электронных средств.

1. СОВРЕМЕННЫЕ ПОДХОДЫ К ПРОЕКТИРОВАНИЮ ЭЛЕКТРОННЫХ СРЕДСТВ

Возможности улучшения технико-эксплуатационных показателей электронных вычислительных средств значительной степени зависят от элементов, используемых для построения их электронных схем, т.е. требуются как высокограмотные разработчики элементной базы, так и современные программные средства автоматизированного проектирования.

Основными достоинствами систем автоматизированного проектирования (САПР) являются:

1. *Более быстрое выполнение чертежей.* Конструктор, использующий САПР, выполняет чертежи в 3 раза быстрее, чем традиционным способом.

2. *Повышение точности выполнения чертежей.* Точность чертежа, выполненного вручную, определяется остротой зрения конструктора и толщиной грифеля карандаша. На чертеже, выполненном с помощью САПР, любое место точки определено точно, а для более детального просмотра его элементов любая часть чертежа может быть увеличена.

3. *Повышение выполнения качества чертежей.* Качество изображения на обычном чертеже полностью зависит от мастерства конструктора, тогда как плоттер САПР рисует линии и тексты независимо от индивидуальных способностей человека.

4. *Возможность многократного использования чертежа.* Память ЭВМ хранит библиотеку символов, стандартов, геометрических форм. В состав чертежа входит ряд компонентов, имеющих одинаковую форму. Запоминание этих компонентов и многократное их использование позволяют повысить эффективность проектирования.

5. *Специальные чертежные средства.* Исследование объекта в трехмерном пространстве, в динамике.

6. *Ускорение расчетов и анализа при проектировании.* Разнообразное математическое обеспечение позволяет произвести расчет заранее.

7. *Высокий уровень проектирования.* Средства ЭВМ позволяют оптимизировать объект проектирования, перебрать варианты.

8. *Сокращение затрат на усовершенствование.* Средства имитации и анализа в САПР позволяют усовершенствовать прототип.

9. *Интеграция проектирования* с другими видами деятельности: автоматизированная система научных исследований (АСНИ), автоматизированная система технологической подготовки производства (АСТПП), гибкие автоматизированные производства (ГАП). АСТПП – это станки с ЧПУ, изготовление и сборка радиоэлектронных устройств с помощью роботов, гибкие производственные системы (ГПС), средства автоматизированного тестирования.

Технические средства, информационное обеспечение САПР позволяют создавать качественно новые вычислительные системы. Усложнение функций проектирования потребовало разработки интеллектуальных САПР: моделирующих и синтезирующих. Организация САПР с элементами искусственного интеллекта позволяет по-новому подойти к проблеме создания вычислительных устройств с переменной структурой, использовать методы структурного и параметрического синтеза, применять количественные и качественные характеристики.

1.1. Общая характеристика проблемы

Основные идеи и принципы проектирования сложных систем выражены в системном подходе. Для специалиста в области системотехники они являются очевидными и естественными, тем не менее их соблюдение и реализация зачастую сопряжены с определенными трудностями, обусловливаемыми особенностями проектирования. Однако интуитивный подход без применения правил системного анализа может оказаться недостаточным для решения все более усложняющихся задач инженерной деятельности. Общий принцип системного подхода заключается в рассмотрении частей явления или сложной системы с учетом их взаимодействия.

Системный подход включает в себя выявление структуры системы, типизацию связей, определение атрибутов, анализ влияния внешней среды. Системный подход рассматривают как направление научного познания и социальной политики.

Теория систем — дисциплина, в которой конкретизируются положения системного подхода; она посвящена исследованию и проектированию сложных экономических, социальных, технических систем, чаще всего слабоструктурированных. Характерными примерами таких систем являются производственные системы. При проектировании систем цели достигаются в многошаговых про-

цессах принятия решений. Методы принятия решений часто выделяют в самостоятельную дисциплину, называемую «Теория принятия решений».

В технике дисциплину, аналогичную теории систем, в которой исследуются технические системы, их проектирование, чаще называют системотехникой. Предметом системотехники являются, во-первых, организация процесса создания, использования и развития технических систем, во-вторых, методы, принципы их проектирования и исследования. В системотехнике важно уметь сформулировать цели системы и организовать ее рассмотрение с позиций поставленных целей. Тогда можно отбросить лишние и малозначимые части при проектировании и моделировании, перейти к постановке оптимизационных задач.

Системы автоматизированного проектирования и управления относятся к числу сложных современных искусственных систем. Их проектирование и сопровождение невозможны без системного подхода. Поэтому идеи и положения системотехники входят составной частью в дисциплины, посвященные изучению современных автоматизированных систем и технологий их применения. Интерпретация и конкретизация системного подхода имеют место в ряде известных подходов с другими названиями, которые также можно рассматривать как компоненты системотехники. Таковы структурный, блочно-иерархический, объектно-ориентированный подходы.

Одним из методов реализации начального этапа при создании электронных средств (ЭС), в том числе автоматизированных систем управления технологическими процессами (АСУ ТП), является моделирование. В настоящее время моделирование осуществляется в основном программными средствами и реализуется на ЭВМ.

Моделирование имеет своей целью описать поведение системы, построить теорию и гипотезы, обеспечивающие наблюдаемое поведение системы, использовать эти теории для прогнозирования поведения системы под действием возможных изменений в самой системе или способов её функционирования. Начальный этап создания ЭС с использованием ЭВМ – *машинный анализ* – формируется следующим образом: при заданных входных воздействиях на ЭС и описании последних требуется найти набор функций и чисел, описывающих поведение, характеристики и свойства системы.

Модели представляют собой средства для визуализации, описания, проектирования и документирования архитектуры систе-

мы. Модели строятся для того, чтобы понять и осмыслить структуру и поведение будущей системы, облегчить управление процессом ее создания и уменьшить возможный риск, а также документировать принимаемые проектные решения. Поскольку сложность систем возрастает, важно располагать эффективными методами моделирования. Наличие строгого стандарта языка моделирования является весьма существенным.

Язык моделирования должен включать:

- элементы модели – фундаментальные концепции моделирования и их семантику;
- нотацию – визуальное представление элементов моделирования;
- руководство по использованию – правила применения элементов в рамках построения тех или иных типов моделей ЭС.

Конечная разработка модели ЭС – это не моделирование, а получение работающих приложений (кода). Диаграммы в конечном счете – всего лишь наглядные изображения, поэтому использование графических языков моделирования целесообразно в следующих случаях:

- при изучении методов проектирования;
- при общении с экспертами организации;
- при получении общего представления о системе.

Графические модели показывают, какой уровень абстракции существует в системе и какие ее части нуждаются в дальнейшем уточнении.

Решения перечисленных проблем в последние годы находят в применении программно-технологических средств специального класса – CASE-средств, реализующих CASE-технологию создания программных средств и информационных систем.

CASE-технология представляет собой совокупность методов проектирования, набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область на всех стадиях разработки и сопровождения моделируемой системы, анализировать эту модель на всех стадиях разработки и подготовить приложения в соответствии с информационными потребностями пользователя. Большинство существующих CASE-средств основано на методах структурного или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы.

1.2. Структурный подход к проектированию электронных средств

Сущность структурного подхода к разработке информационной системы (ИС), описывающей работу технической системы, заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и т.д. Процесс разбиения 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 наиболее распространенные методологии структурного подхода [23] базируются на ряде общих принципов [13]. В качестве базовых используются два принципа:

- «разделяй и властвуй» – принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;
- иерархического упорядочивания – принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Выделение двух базовых принципов не означает, что остальные принципы являются второстепенными, поскольку игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к провалу всего проекта). К основным из них относятся принципы:

- абстрагирования – заключается в выделении существенных аспектов системы и отвлечении от несущественных;
- формализации – состоит в необходимости строгого методического подхода к решению проблемы;
- непротиворечивости – заключается в обоснованности и согласованности элементов;
- структурирования данных – данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой, и отношения между данными. Каждой группе средств соответствуют опре-

деленные 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».

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного обеспечения: архитектуру программного обеспечения, структурные схемы программ и диаграммы экранных форм. Перечисленные модели в совокупности дают полное описание ИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

Методология функционального моделирования SADT. Разработанная Дугласом Россом, эта методология получила дальнейшее развитие [4]. На ее основе разработана, в частности, известная методология IDEF0 (Icam DEFinition), которая является основной частью программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой по инициативе ВВС 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 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 (правило трех – шести 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 данные функции. Для уже существующих систем SADT может быть использована для анализа функций, выполняемых системой, а также для указания механизмов, посредством которых они осуществляются.

Состав функциональной модели. 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 – с правой стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рис. 1.1).

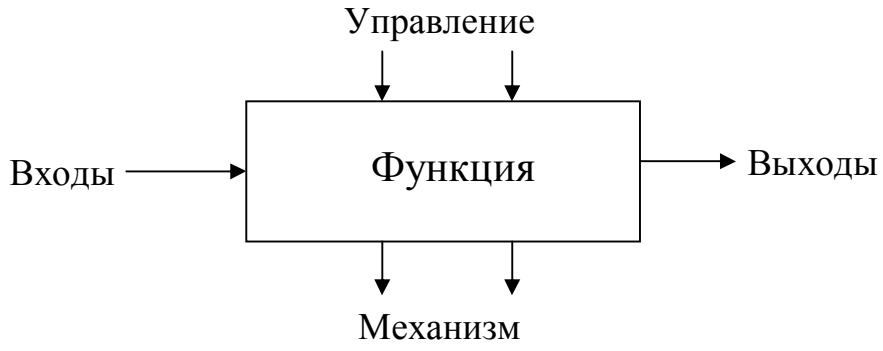


Рис. 1.1. Функциональный блок и интерфейсные дуги

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

На рис. 1.2 показана структура SADT-модели, где приведены четыре диаграммы и их взаимосвязи. Каждый компонент модели может быть декомпозирован на другой диаграмме. Каждая диаграмма иллюстрирует внутреннее строение блока на родительской диаграмме.

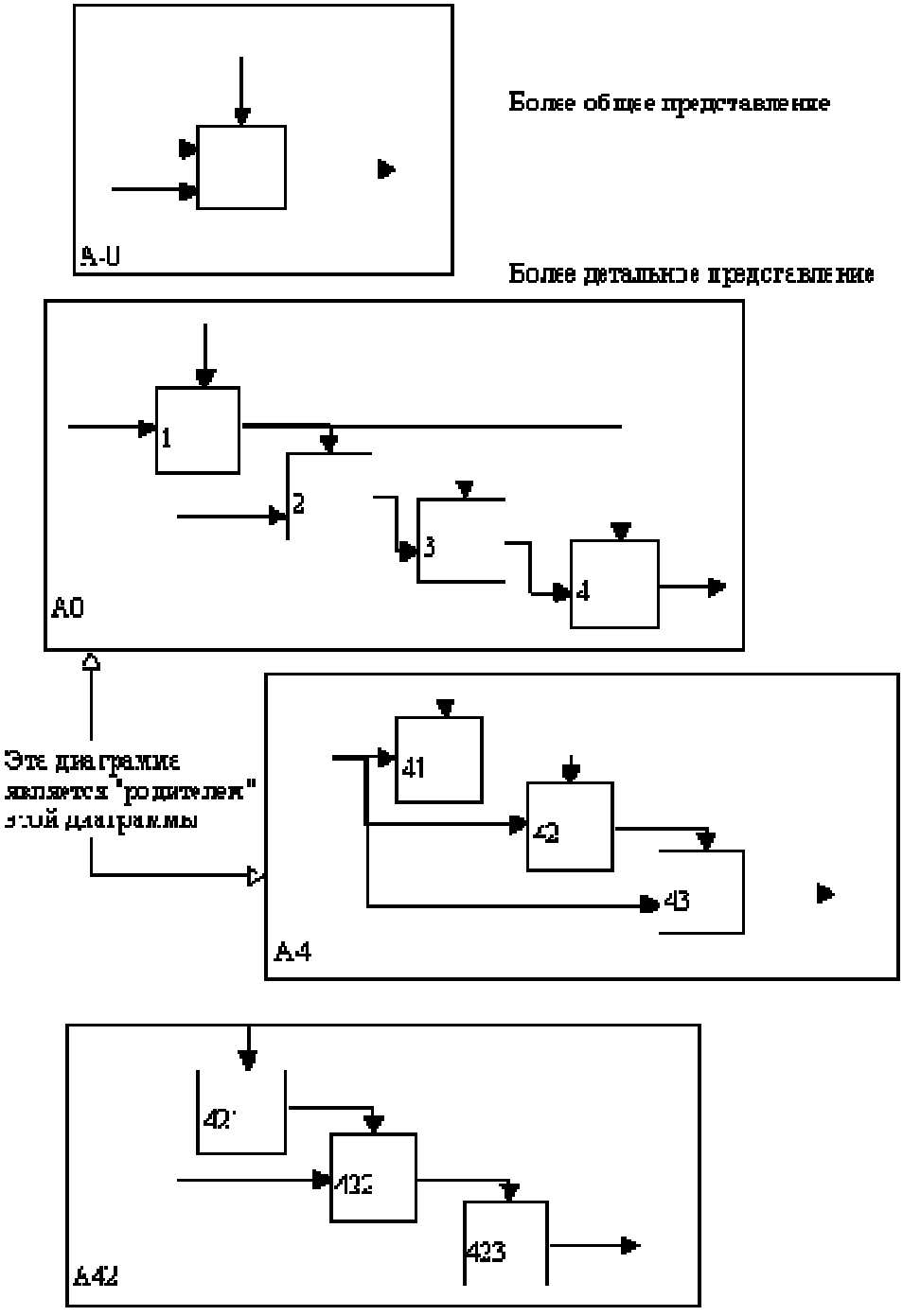


Рис. 1.2. Структура SADT-модели. Декомпозиция диаграмм

Иерархия диаграмм. 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 изображают основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых – блок, границы которого определены интерфейсными дугами. Любая 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. 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. Каждая детальная диаграмма является декомпозицией блока из более общей диаграммы. На каждом шаге декомпозиции более общая диаграмма называется родительской для более детальной диаграммы.

Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, – те же самые, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

На рис. 1.3 – 1.5 приведены различные варианты выполнения функций и соединения дуг с блоками.

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.

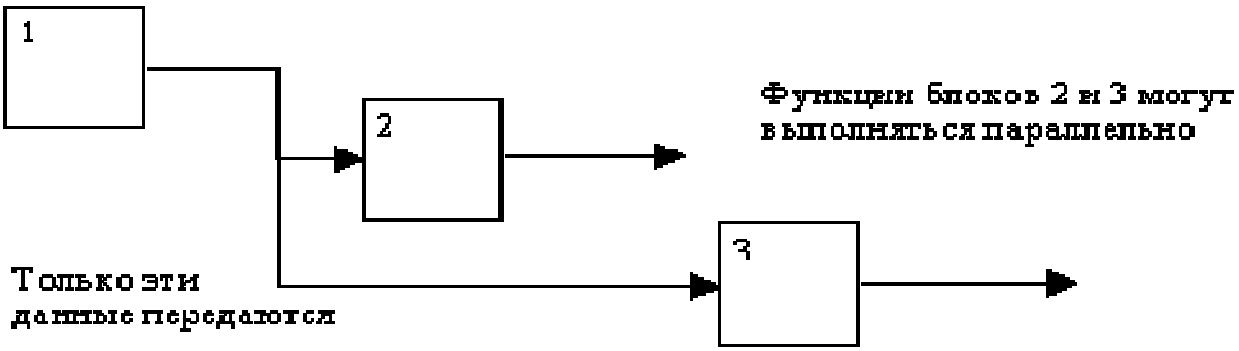


Рис. 1.3. Одновременное выполнение

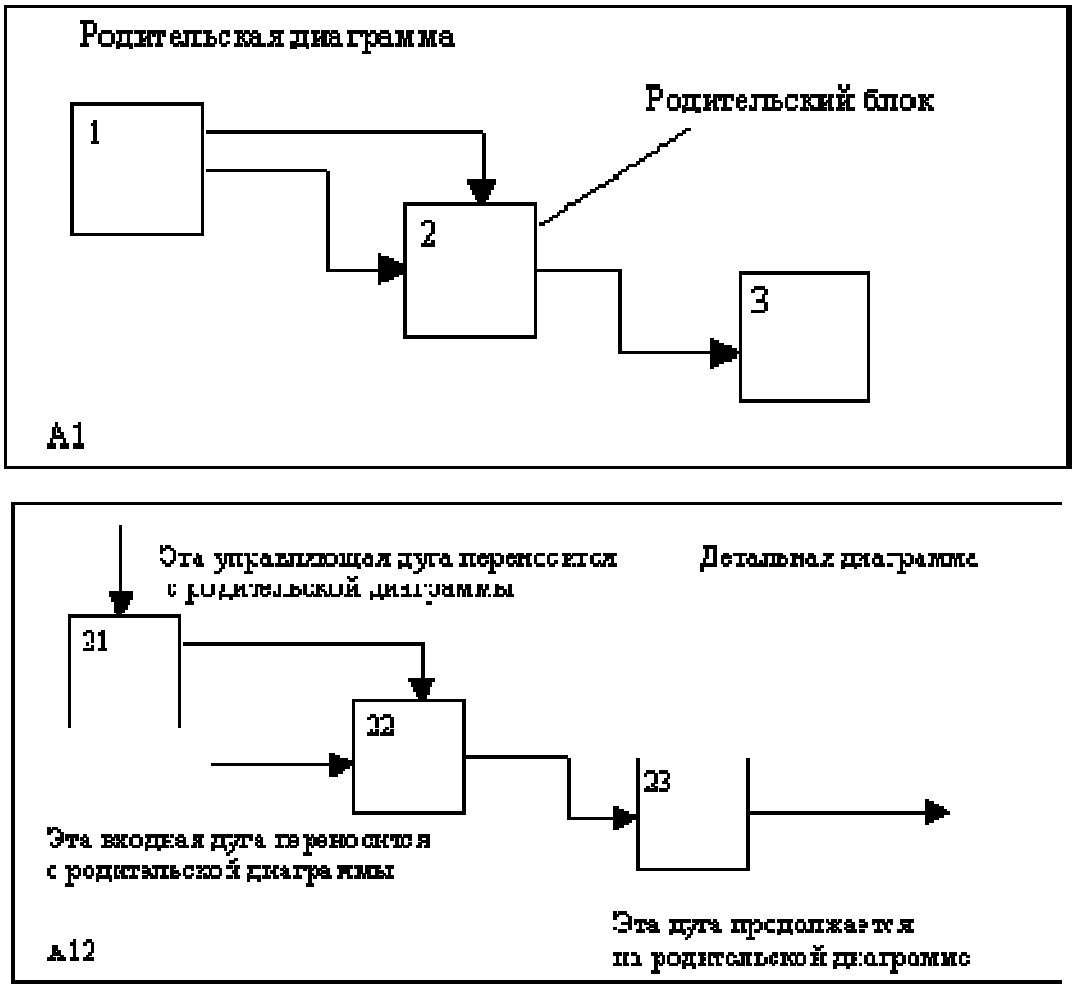


Рис. 1.4. Пример полного и непротиворечивого соответствия

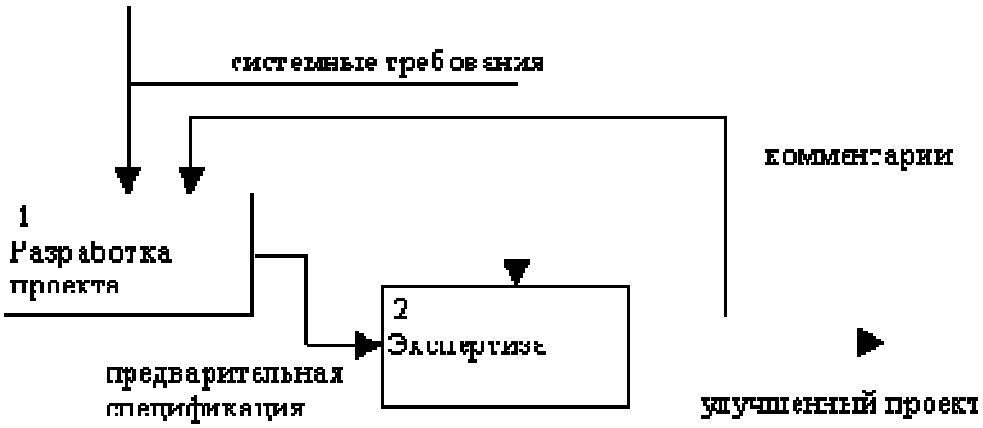


Рис. 1.5. Пример обратной связи

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. (см. рис. 1.5).

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 (рис. 1.6).

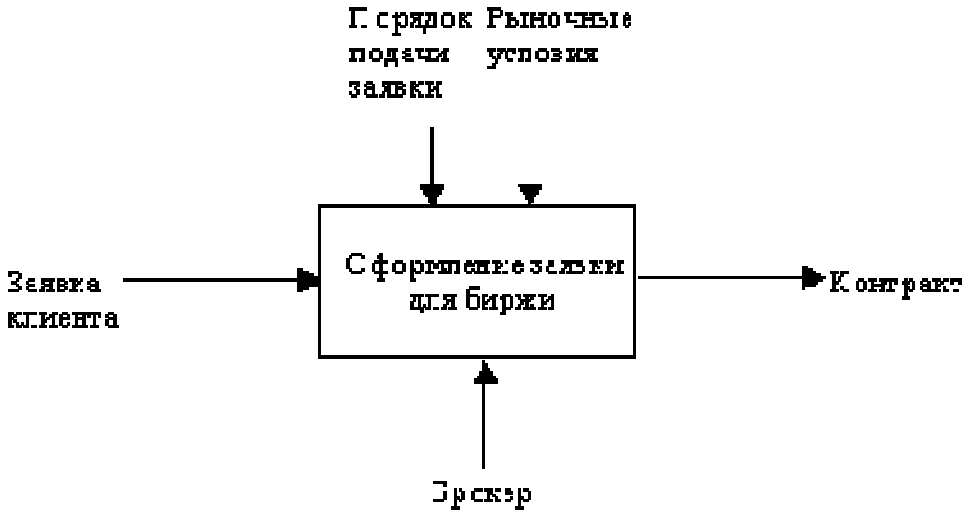


Рис. 1.6. Пример механизма

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть описан диаграммой нижнего уровня, которая,

в свою очередь, может быть 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 диаграмма A2 детализирует блок 2 на диаграмме A0, которая является самой верхней диаграммой модели. На рис. 1.7 показано типичное дерево диаграмм.

Рис. 1.7. Иерархия диаграмм

Типы связей между функциями. Одним из важных моментов при проектировании ИС с помощью методологии SADT является точная согласованность типов связей между функциями. Различают, по крайней мере, семь типов связывания (табл. 1.1):

(0) *Тип 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. Крайний вариант этого случая показан на рис. 1.8.

(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.

Таблица 1.1

Типы связей

Тип связи	Относительная значимость
Случайная	0
Логическая	1
Временная	2
Процедурная	3
Коммуникационная	4
Последовательная	5
Функциональная	6

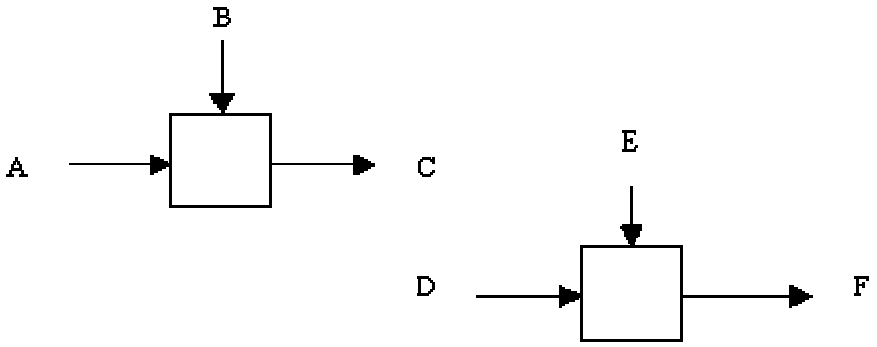


Рис. 1.8. Случайная связность

(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. Пример процедурно-связанной диаграммы приведен на рис. 1.9.

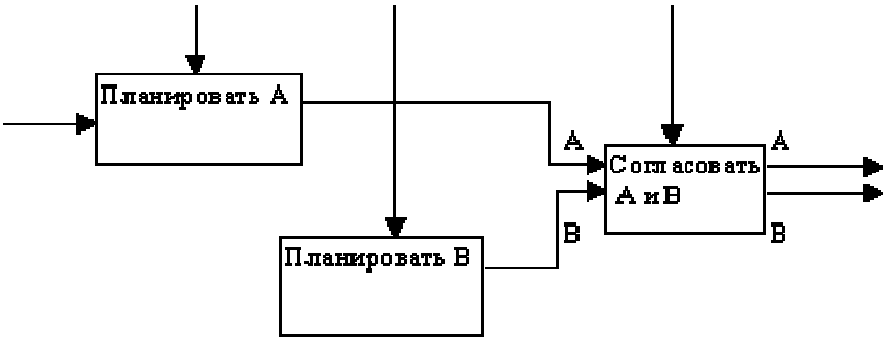


Рис. 1.9. Процедурная связность

(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 (рис. 1.10).

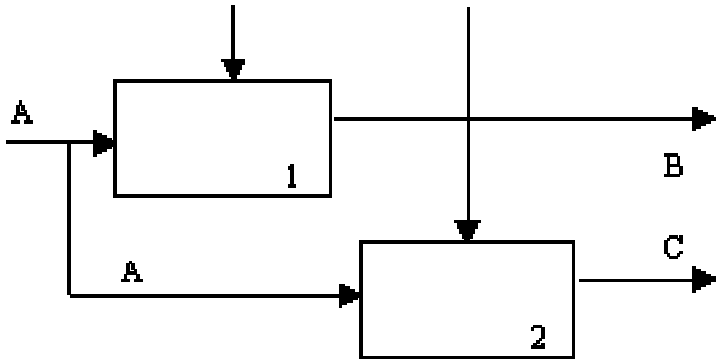


Рис. 1.10. Коммуникационная связность

(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 (рис. 1.11).

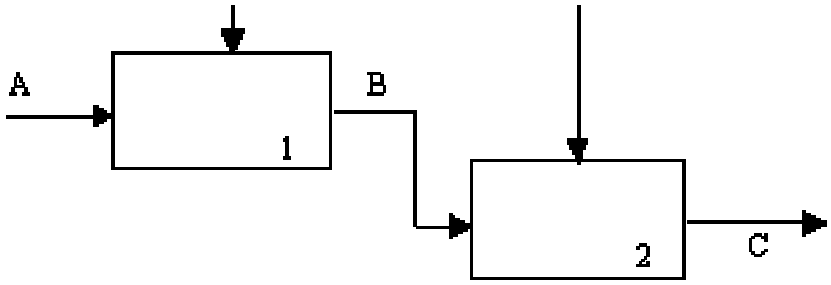


Рис. 1.11. Последовательная связность

(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 (рис. 1.12).

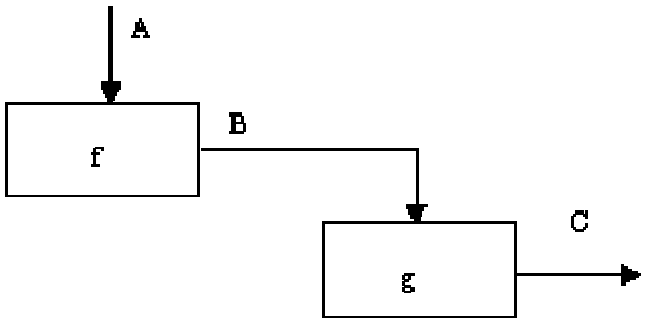


Рис. 1.12. Функциональная связность

В математических терминах необходимое условие для простейшего типа функциональной связности, показанной на рис. 1.12, имеет следующий вид: $C = g(B) = g(f(A))$. Важно отметить, что уровни 4–6 (табл. 1.2) устанавливают типы связностей, которые разработчики считают важнейшими для получения диаграмм хорошего качества.

Таблица 1.2

Значимость и типы функциональной связности

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

Контрольные вопросы

1. Определите цели имитационного моделирования.
2. Какой принцип лежит в основе структурного подхода?
3. Перечислите состав функциональной модели.
4. Сколько функциональных блоков может быть в модели?
5. Перечислите типы связей между функциями.

Задание для самостоятельной работы

Постройте иерархию диаграмм для задания «Разработать курсовой проект».

2. CASE-СРЕДСТВА

2.1. Общая характеристика и классификация

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 программного обеспечения (ПО).

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.

К CASE-средствам относят, как правило, любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла ПО и обладающее такими характерными особенностями, как:

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

Интегрированное CASE-средство (или комплекс средств, поддерживающих полный жизненный цикл ПО) 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 жизненного цикла ИС (toolkit) и полностью интегрированные средства, поддерживающие весь ЖЦ ИС и связанные общим репозиторием.

Помимо этого CASE-средства можно классифицировать по следующим признакам:

- 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 основные типы:

- 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), Silverrun (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-Designor (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, такие как: 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-Designor. 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)].

К вспомогательным типам относятся:

- 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)].

На сегодняшний день российский рынок программного обеспечения располагает такими наиболее развитыми CASE-средствами, как:

- Vantage Team Builder (Westmount I-CASE);
- Designer/2000;

- Silverrun;
- ERwin+BPwin;
- S-Designor;
- CASE.Аналитик.

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

2.2. Технология внедрения CASE-средств

Технология внедрения CASE-средств базируется в основном на стандартах IEEE (Institute of Electrical and Electronics Engineers – Институт инженеров по электротехнике и электронике) [7, 23]. Термин «внедрение» используется в широком смысле и включает все действия – от оценки первоначальных потребностей до полномасштабного использования CASE-средств в различных подразделениях организации-пользователя. Процесс внедрения CASE-средств состоит из следующих этапов [7]:

– определение потребностей в CASE-средствах;
– оценка и выбор CASE-средств;
– выполнение пилотного проекта;
– практическое внедрение CASE-средств.

Процесс успешного внедрения CASE-средств не ограничивается их использованием. На самом деле он охватывает планирование и реализацию множества технических, организационных, структурных процессов, изменений в общей культуре организации и основан на четком понимании возможностей CASE-средств.

На способ внедрения CASE-средств может повлиять специфика конкретной ситуации. Например, если заказчик предпочитает конкретное средство или оно оговаривается требованиями контракта, этапы внедрения должны соответствовать такому предопределенному выбору. В иных ситуациях относительная простота или сложность средства, степень согласованности или конфликтности с существующими в организации процессами, требуемая степень интеграции с другими средствами, опыт и квалификация пользователей могут привести к внесению соответствующих корректив в процесс внедрения.

Определение потребностей в CASE-средствах. Данный этап включает достижение понимания потребностей организации и технологии последующего процесса внедрения CASE-средств (рис. 2.1). Он должен привести к выделению тех областей деятельности организации, в которых применение CASE-средств может принести реальную пользу. Результатом данного этапа является документ, определяющий стратегию внедрения CASE-средств.

Рис. 2.1. Определение потребностей в CASE-средствах

Оценка и выбор CASE-средств. Процессы оценки и выбора тесно взаимосвязаны. По результатам оценки цели выбора и/или критерии выбора и их веса могут потребовать модификации. В таких случаях может понадобиться повторная оценка. Когда анализируются окончательные результаты оценки и к ним применяются критерии выбора, может быть рекомендовано приобретение, разработка, модификация CASE-средства или набора CASE-средств или отказ от внедрения.

Процесс выбора состоит из следующих действий:

– формулировка задач выбора, включая цели, предположения и ограничения;

– выполнение всех необходимых действий по выбору, в том числе определение и ранжирование критериев, определение средств-кандидатов, сбор необходимых данных и применение ранжированных критериев к результатам оценки для определения средств с наилучшими показателями. Для многих пользователей важным критерием выбора является интегрируемость CASE-средства с существующей средой;

– выполнение необходимого количества итераций с тем, чтобы выбрать (или отвергнуть) средства, имеющие сходные показатели;

– подготовка отчета по результатам выбора.

В процессе выбора возможно получение двух результатов:

– рекомендаций по выбору конкретного CASE-средства;

– запроса на получение дополнительной информации, необходимой для оценки.

Масштаб выбора должен устанавливать требуемый уровень детализации, необходимые ресурсы, график и ожидаемые результаты. Существуют параметры, которые могут быть применены для определения масштаба:

– предварительный отбор (например, отбор только средств, работающих на конкретной платформе);

– использование ранее полученных результатов оценки, результатов оценки из внешних источников или комбинации того и другого.

В случае если предыдущие оценки выполнялись с использованием различных наборов критериев или конкретных критериев, но разными способами, результаты оценок должны быть представлены в согласованной форме. После завершения данного шага оценка каждого CASE-средства должна быть представлена в рамках единого набора критериев и сопоставима с другими оценками.

Алгоритмы, используемые для выбора, могут быть основаны на масштабе или ранге. Алгоритмы, основанные на масштабе, вычисляют единственное значение для каждого CASE-средства путем умножения веса каждого критерия на его значение (с учетом масштаба) и сложения всех произведений. CASE-средство с наивысшим результатом получает первый ранг. Алгоритмы, основанные на ранге, используют ранжирование CASE-средств–кандидатов по отдельным критериям или группам критериев в соответствии со значениями критериев в заданном масштабе. Затем ранги сводятся вместе, и вычисляются общие значения рангов.

При анализе результатов выбора предполагается, что процесс выбора завершен, CASE-средство выбрано и рекомендовано к использованию. Тем не менее может потребоваться более точный анализ для определения степени зависимости значений ключевых критериев от различий в значениях характеристик CASE-средств – кандидатов. Такой анализ позволит установить, насколько результат ранжирования CASE-средств зависит от оптимальности выбора весовых коэффициентов критериев. Он также может использоваться для определения существенных различий между CASE-средствами с очень близкими значениями критериев, или рангами.

Если ни одно CASE-средство не удовлетворяет минимальным критериям, выбор (возможно, вместе с оценкой) может быть повторен для других CASE-средств – кандидатов.

Если различия между самыми предпочтительными кандидатами несущественны, дополнительная информация может быть получена путем повторного выбора (возможно, вместе с оценкой) с использованием дополнительных или других критериев. Рекомендации по выбору должны быть строго обоснованы. В случае отсутствия адекватных CASE-средств рекомендуется разработать новое CASE-средство, модифицировать существующее или отказаться от внедрения.

Модель процесса оценки и выбора (рис. 2.2) описывает наиболее общую ситуацию оценки и выбора, а также показывает зависимость между ними [17]. Оценка и выбор могут выполняться независимо друг от друга или вместе, каждый из этих процессов требует применения определенных критериев.

Процесс оценки и выбора может преследовать несколько целей, включая одну или более из следующих:

- оценку нескольких CASE-средств и выбор одного или более из них;
- оценку одного или более CASE-средств и сохранение результатов для последующего использования;
- выбор одного или более CASE-средств с использованием результатов предыдущих оценок.

Входной информацией для процесса оценки является (см. рис. 2.2.):

– определение пользовательских потребностей;

– цели и ограничения проекта;

– данные о доступных CASE-средствах;

– список критериев, используемых в процессе оценки.

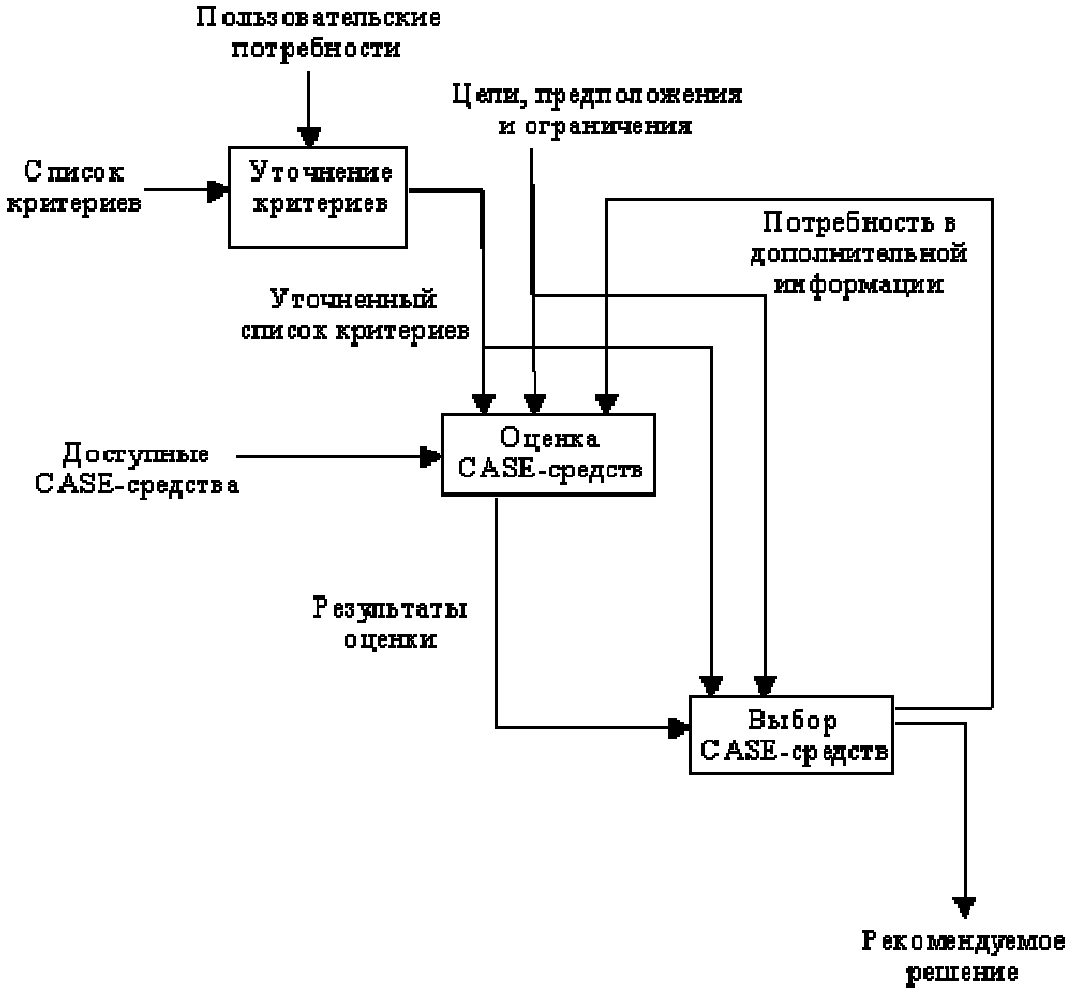


Рис. 2.2. Модель процесса оценки и выбора

Результаты оценки могут включать результаты предыдущих оценок. При этом не следует забывать, что набор критериев, использовавшихся при предыдущей оценке, должен быть совместимым с текущим набором. Конкретный вариант реализации процесса (оценка и выбор, оценка для будущего выбора или выбор, основанный на предыдущих оценках) определяется перечисленными выше целями.

Элементы процесса включают:

- цели, предположения и ограничения, которые могут уточняться в ходе процесса;
- потребности пользователей, отражающие их количественные и качественные требования к CASE-средствам;
- критерии, определяющие набор параметров, в соответствии с которыми производится оценка и принятие решения о выборе;
- формализованные результаты оценок одного или более средств;
- рекомендуемое решение (либо решение о выборе, либо дальнейшая оценка).

Процесс оценки и/или выбора может быть начат только тогда, когда лицо, группа или организация полностью определила для себя конкретные потребности и формализовала их в виде количественных и качественных требований в заданной предметной области. Термин «пользовательские требования» означает именно такие формализованные требования.

Пользователь должен определить конкретный порядок действий и принятия решений с любыми необходимыми итерациями. Например, процесс может быть представлен в виде дерева решений с его последовательным обходом и выбором подмножеств кандидатов для более детальной оценки. Описание последовательности действий должно определять поток данных между ними.

Определение списка критериев основано на пользовательских требованиях и включает:

- выбор критериев для использования из приведенного далее перечня;
- определение дополнительных критериев;
- установление области использования каждого критерия (оценка, выбор или оба процесса);
- определение одной или более метрик для каждого критерия оценки;
- назначение веса каждому критерию при выборе.

CASE-технология представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных систем ПО, поддержанную комплексом взаимосвязанных средств автоматизации. CASE предоставляет системным аналитикам, проектировщикам и программистам инструментарий для автоматизации проектирования и разработки ПО.

CASE позволяет не только получать корректные программные продукты, но и обеспечивает технологически правильный процесс их создания. Главная 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 ПО. Основной акцент в процессе создания ПО приходится на этапы анализа и проектирования, в отличие от кодирования.

CASE-технологии широко применяются для многих типов систем ПО, но чаще всего – в следующих областях:

1. Разработка 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 коммерческим структурам решать задачи стратегического планирования, управления финансами, определения политики фирм, обучения персонала (это направление получило название *бизнес-анализ*).

2. Создание системного и управляющего ПО. Использование CASE-технологии в этой отрасли вызвано высокой сложностью данного вида работ и необходимостью повышения их производительности.

Помимо автоматизации структурных методологий и возможности применения современных методов системной и программной инженерии, CASE-средства имеют ряд преимуществ:

– повышают качество создаваемого ПО благодаря использованию средств автоматического контроля, в частности контроля проекта;

– поддерживают создание прототипа будущей системы, что позволяет на ранних этапах оценить ожидаемый результат;

– ускоряют процесс проектирования и разработки;

– освобождают разработчика от рутинной работы, предоставляя ему возможность сосредоточиться на творческой части разработки;

– поддерживают развитие и сопровождение разработки;

– обеспечивают технологии повторного использования компонентов разработки.

Локальные средства (ERwin, BPwin, S-Designor, CASE.Аналитик)

ERwin – средство концептуального моделирования БД, использующее методологию IDEF1X. ERwin реализует проектирование схемы БД, генерацию ее описания на языке целевой СУБД (ORACLE, Informix, Ingres, Sybase, DB/2, Microsoft SQL Server, Progress и др.) и реинжиниринг существующей БД [24]. ERwin выпускается в нескольких конфигурациях, ориентированных на наиболее распространенные средства разработки приложений 4GL. Версия ERwin/OPEN полностью совместима со средствами разработки приложений PowerBuilder и SQLWindows и позволяет экспортировать описание спроектированной БД непосредственно в репозитории данных средств.

Для ряда средств разработки приложений (PowerBuilder, SQLWindows, Delphi, Visual Basic) выполняется генерация форм и прототипов приложений.

Сетевая версия Erwin ModelMart обеспечивает согласованное проектирование БД и приложений в рамках рабочей группы.

BPwin – средство функционального моделирования, реализующее методологию IDEF0.

Возможные конфигурации CASE-средств и ориентировочная стоимость средств (без технической поддержки) приведены в табл. 2.1.

Таблица 2.1

Стоимость и конфигурация CASE-средств

Конфигурация	Стоимость, $
ERwin/ERX	3,295
Bpwin	2,495
ERwin/ERX for PowerBuilder, Visual Basic, Progress	3,495
ERwin/ERX for Delphi	4,295
ERwin/Desktop for PowerBuilder, Visual Basic	495
ERwin/ERX for SQLWindows / Designer/2000 / Solaris	3,495 / 5,795 / 6,995
ModelMart 5 / 10 user	11,995 / 19,995
Erwin/OPEN for ModelMart	3,995

S-Designor 4.2 представляет собой CASE-средство для проектирования реляционных баз данных [25]. 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. Для существующих систем выполняется реинжиниринг БД.

S-Designor совместим с рядом средств разработки приложений (PowerBuilder, Uniface, TeamWindows и др.) и позволяет экспортировать описание БД в репозитории данных средств. Для PowerBuilder выполняется также прямая генерация шаблонов приложений.

CASE.Аналитик 1.1 является практически единственным в настоящее время конкурентоспособным отечественным CASE-средством функционального моделирования и реализует построение диаграмм потоков данных [3]. Его основные функции:

- построение и редактирование DFD;
- анализ диаграмм и проектных спецификаций на полноту и непротиворечивость;
- получение разнообразных отчетов по проекту;
- генерация макетов документов в соответствии с требованиями ГОСТ 19.XXX и 34.XXX.

Среда функционирования: процессор – Pentium100 и выше, основная память – 4 Мб, дисковая память – 5 Мб, MS Windows 95 и выше. База данных проекта реализована в формате СУБД Paradox и является открытой для доступа. С помощью отдельного программного продукта (Catherine) выполняется обмен данными с CASE-средством ERwin. При этом из проекта, выполненного в CASE.Аналитике, экспортируется описание структур данных и накопителей данных, которое по определенным правилам формирует описание сущностей и их атрибутов.

2.3. Примеры комплексов CASE-средств

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 ЖЦ ПО. Сравниваться могут комплексы методологически и технологически согласованных инструментальных средств, поддерживающие полный ЖЦ ПО и обеспеченные необходимой технической и методической поддержкой со стороны фирм-поставщиков.

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 Silverrun-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.

Примерами других подобных комплексов являются:

- комплекс средств, поставляемых и используемых фирмами «DataX/Florin» и «ЛАНИТ»: Vantage Team Builder for Uniface + + Uniface;

- комплекс средств, поставляемых и используемых фирмой «ФОРС»:

– CASE-средства Designer/2000 (основное), ERwin, Bpwin и Oowin (альтернативные);

– средства разработки приложений Developer/2000, ORACLE Power Objects (основные) и Usoft Developer (альтернативное);

– средство настройки и оптимизации ExplainSQL (Platinum);

– средства администрирования и сопровождения SQLWatch, DBVision, SQL Spy, TSReorg и др. (Platinum);

– средство документирования ORACLE Book;

- комплекс средств на основе продуктов фирмы CENTURA:

– CASE-средства ERwin, Bpwin и Oowin (объектно-ориентированный анализ);

– средства разработки приложений SQLWindows и TeamWindows;

– средство тестирования и оптимизации приложений «клиент–сервер» SQLBench (ARC);

– средства эксплуатации и сопровождения Quest и Crystal Reports.

2.4. Связь процессов проектирования и конструирования. Модели жизненного цикла программного обеспечения

Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО (под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ. Модель ЖЦ зависит от специфики ИС и условий, в которых последняя создается и функционирует). Регламенты стандарта являются общими для любых моделей ЖЦ, методологий и технологий разработки. ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

К настоящему времени наибольшее распространение получили две основные модели ЖЦ:

- каскадная модель;
- спиральная модель.

В изначально существовавших однородных ИС каждое приложение представляло собой единое целое. Для разработки такого типа

приложений применялся каскадный способ. Его основной характеристикой является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем (рис. 2.3). Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой.

Положительные стороны применения каскадного подхода заключаются в следующем [2]:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении ИС, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования, с тем чтобы предоставить разработчикам свободу реализовать их как можно лучше с технической точки зрения. В эту категорию попадают сложные расчетные системы, системы реального времени и другие подобные задачи.

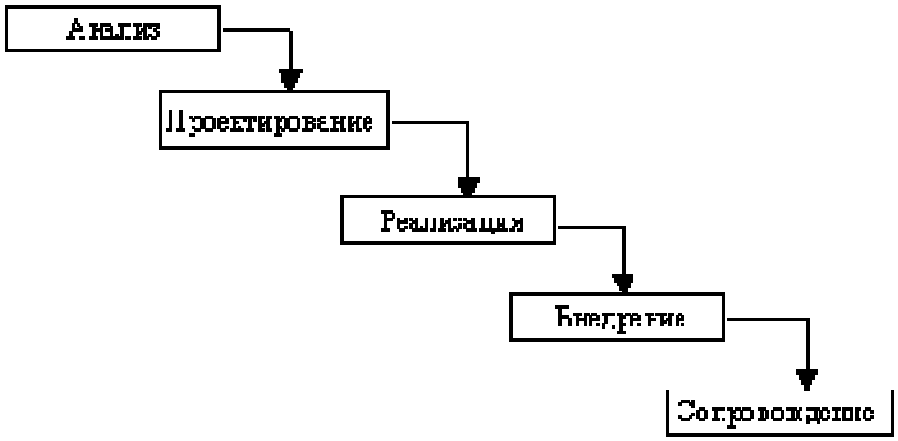


Рис. 2.3. Каскадная схема разработки ПО

Однако в процессе использования каскадного подхода обнаружился ряд его недостатков, вызванных прежде всего тем, что реальный процесс создания ПО никогда полностью не укладывался в такую жесткую схему. В процессе создания ПО постоянно возникала потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ПО принимает вид, показанный на рис. 2.4.

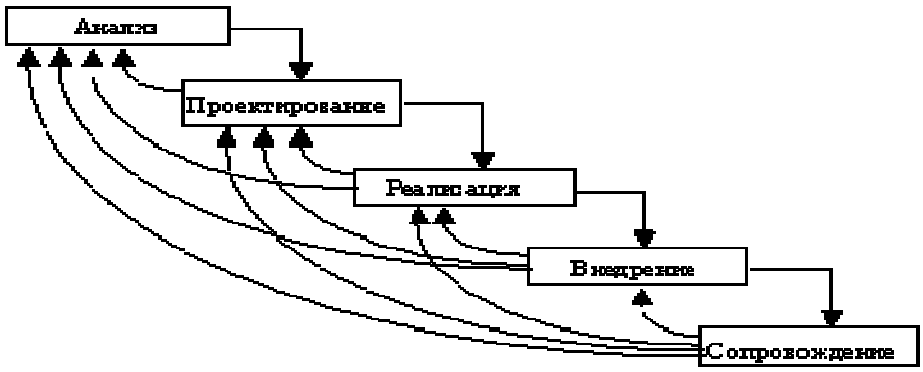


Рис. 2.4. Реальный процесс разработки ПО по каскадной схеме

Основным недостатком каскадного подхода является существенное запаздывание с получением результатов. Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ, требования к ИС «заморожены» в виде технического задания на все время ее создания. Таким образом, пользователи могут внести свои замечания только после того, как работа над системой будет полностью завершена. В случае неточного изложения требований или их изменения в течение длительного периода создания ПО пользователи получают систему, не удовлетворяющую их потребностям. Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением.

Для преодоления перечисленных проблем была предложена спиральная модель ЖЦ, делающая упор на начальные этапы ЖЦ – анализ и проектирование (рис. 2.5) [10]. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации.

Разработка ПО итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача – как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

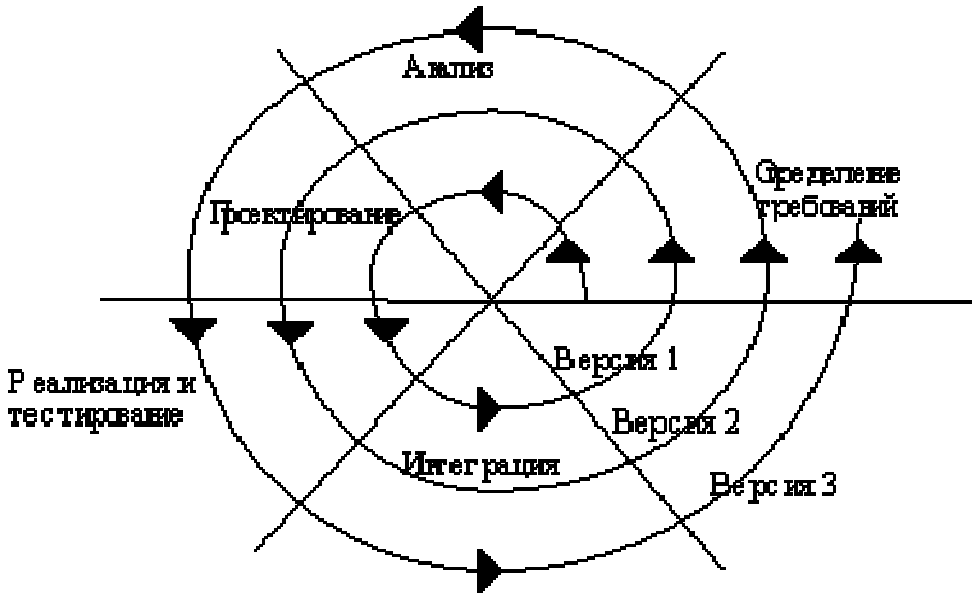


Рис. 2.5. Спиральная модель ЖЦ

Основная проблема спирального цикла – определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Контрольные вопросы

1. Какие программные средства относят к CASE-средствам?
2. Перечислите основные компоненты интегрированного CASE-средства.
3. Назовите основные и вспомогательные типы CASE-средств.
4. Каковы основные этапы процесса внедрения CASE-средств?
5. Охарактеризуйте каждый этап процесса внедрения CASE-средств.
6. Приведите примеры комплексов CASE-средств.
7. Какие модели жизненного цикла программного обеспечения вы знаете? Каковы их достоинства и недостатки?
8. Чем различаются идеальный и реальный процессы разработки ПО по каскадной схеме?
9. Назовите область применения CASE-технологий.

3. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ, МАТЕМАТИЧЕСКИЕ МЕТОДЫ И МОДЕЛИ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ЭЛЕКТРОННЫХ СРЕДСТВ

3.1. Основы построения систем автоматизированного проектирования электронных средств

Необходимость создания САПР. Принципы создания САПР. Виды обеспечения САПР

Система автоматизированного проектирования (САПР) – это комплекс средств автоматизации проектирования, взаимосвязанных с проектными организациями (пользователями системы). САПР включает технические средства, математическое, программное, информационное и лингвистическое обеспечение (специальные языки, проблемно-ориентированные).

Ускорение темпов развития науки и техники привели к таким особенностям при проектировании электронных средств (ЭС), как:

1) непрерывный рост тактико-технических требований (масса, надежность, стоимость, электрические показатели и др.);

2) резкое сокращение сроков морального старения ЭС;

3) увеличение стоимости разработок;

4) сокращение сроков, отводимых на разработку новых изделий.

Эффективно решать эти противоречивые проблемы возможно, лишь применяя в процессе проектирования различные САПР, что, в частности, позволит:

– проанализировать большое количество вариантов различных решений;

– создавать конструкции, оптимально учитывающие предъявляемые к ним требования;

– использовать более точные методы расчета и проектирования, сводящие к минимуму подстроечно-регулировочные операции;

– сократить сроки и снизить стоимость разработки аппаратуры.

При создании САПР учитываются принципы:

• *системного единства,* т.е. целостность системы, взаимосвязь между подсистемами и ее элементами;

• *совместимости,* т.е. обеспечиваются совместное функционирование составных частей САПР и сохранность открытости системы в целом;

• *типизации* (ориентирует на преимущественное создание и использование типовых и унифицированных элементов САПР с последующей их модернизацией);

• *развития* (способствует совершенствованию и обновлению составных частей САПР, а также взаимодействию и расширению взаимосвязи с автоматизированными системами различного уровня и функционального назначения);

• *иерархичности* (проектирование по уровням структуры САПР).

Основными причинами создания автоматизированных систем проектирования являются:

1. Разрозненность отдельных методов автоматизации (подготовка документации, чертежей, моделирование технических систем, оптимизация параметров, организация экспертиз, обработка результатов, принятие решений при многокритериальной постановке задач), отсутствие методологического единства, не позволяющие создать эффективную систему проектирования.

2. Выполнение большого объема работ в сжатые сроки.

3. Повышение требований к качеству проектов (изделий, машин и механизмов).

Удовлетворить эти противоречивые требования с помощью простого увеличения численности проектировщиков нельзя, так как возможность параллельного проведения проектных работ ограничена.

Цель САПР – это повышение качества проектов, снижение материальных затрат, сокращение сроков проектирования и ликвидация тенденции к росту числа проектировщиков, а также повышение производительности их труда. Для САПР характерно системное использование ЭВМ при рациональном распределении функций между человеком и ЭВМ. С помощью ЭВМ решаются задачи, поддающиеся формализации (решение системы уравнений).

Предметом САПР являются формализация проектных процедур, структурирование и типизация процессов проектирования, постановка, модели, методы и алгоритмы решения проектных задач, способы построения технических средств, создания языков, описания программ, банков данных, а также вопросы их объединения в единую проектирующую систему.

Для создания САПР необходимы:

1) совершенствование проектирования на основе применения математических методов и средств вычислительной техники;

2) автоматизация процесса поиска, обработки и выдачи информации;

3) использование методов оптимизации и многовариантного проектирования; применение математических моделей проектируемых объектов;

4) создание банков данных, содержащих сведения справочного характера;

5) повышение качества оформления проектной документации;

6) увеличение творческой доли труда проектировщиков за счет автоматизации нетворческих работ;

7) унификация и стандартизация методов проектирования;

8) подготовка и переподготовка специалистов в области САПР;

9) взаимодействие САПР с автоматизированными системами различного уровня.

Выделяют семь видов обеспечения САПР: математическое, программное, информационное, техническое, лингвистическое, методическое, организационное.

Математическое обеспечение (МО) включает в себя алгоритмы, по которым разрабатывается программное обеспечение; функциональные модели проектируемых объектов; методы численного решения задач; методы поиска экстремума.

МО САПР делится:

• на математические методы и построение на их основе математических моделей объектов проектирования;

• формализованное описание технологии автоматизированного проектирования.

МО должно описывать во взаимосвязи объект, процесс и средства автоматизации проектирования.

Программное обеспечение (ПО) – это совокупность всех программ и эксплутационной документации к ним, необходимых для выполнения автоматизированного проектирования. ПО делится на общесистемное и специальное (прикладное).

Общесистемное ПО создано для организации функционирования технических средств, т. е. планирования и управления вычислительным процессом, распределения имеющихся ресурсов. Общесистемное ПО близко по назначению к операционным системам.

Специальное ПО реализует математическое обеспечение для непосредственного выполнения проектных процедур.

Специальное, или прикладное, ПО имеет форму пакетов прикладных программ (ППП).

Уровни программного обеспечения: машинный код, язык Ассемблера, языки высокого уровня (рис. 3.1).

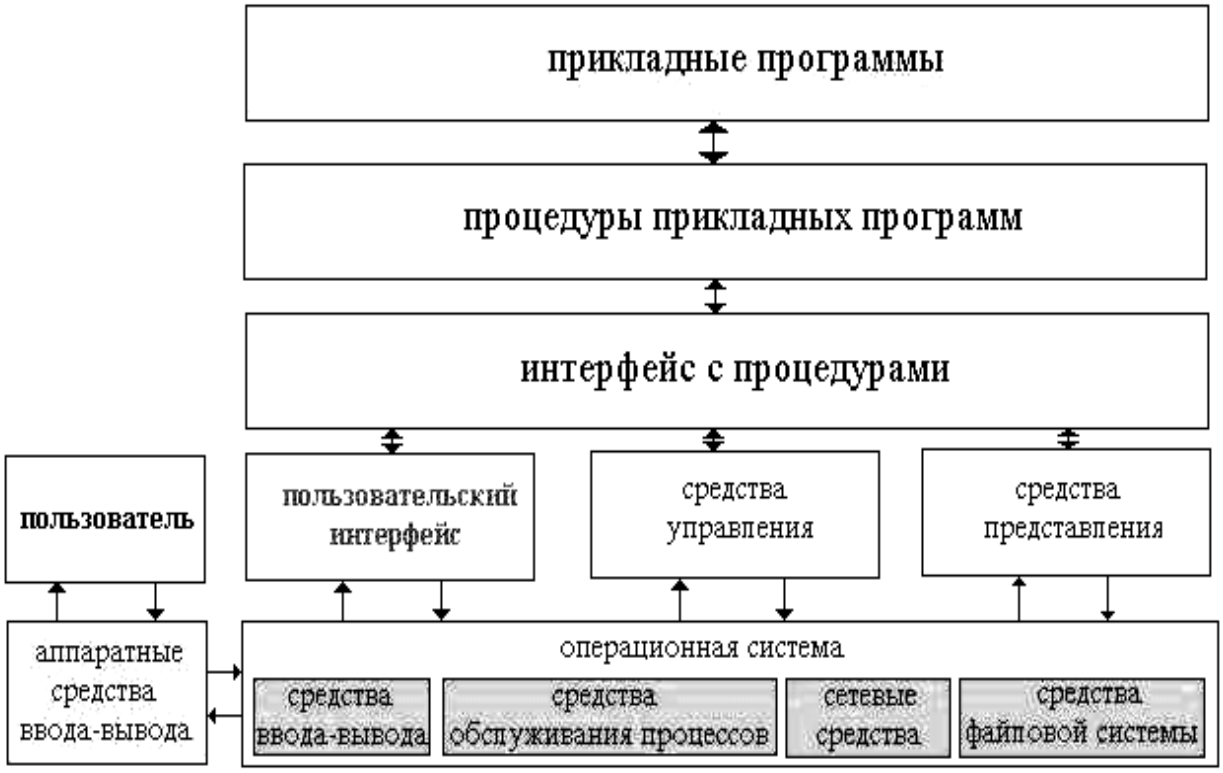


Рис. 3.1. Иерархическая структура программного обеспечения

Информационное обеспечение (ИО) – данные, которыми пользуется проектировщик в процессе проектирования для выработки проектного решения. Это справочные данные о комплектующих изделиях, типовых проектных решениях, параметрах элементов, сведения о состоянии текущих разработок в виде промежуточных и окончательных проектных решений, структур и параметров проектируемых объектов. Совокупность данных, используемых в САПР, составляет информационный фонд. Основная функция ИО – это ведение фонда, обновление, сохранение и организация доступа к данным.

В состав ИО САПР входят:

- программные модули;
- исходные и результирующие данные для программных модулей;
- нормативно-справочная проектная документация, государственные и отраслевые стандарты, руководящие материалы и указания,

типовые проектные решения, текущая проектная документация, отражающая ход и состояние выполнения проекта.

Различают следующие способы ведения ИО САПР:

– использование файловой системы;
– построение библиотек;
– использование баз данных (БД);
– создание информационных программ-адаптеров.

Применение файловой системы и построение библиотек широко распространено, так как поддерживается средствами операционной системы. Эти способы применяют при хранении программных модулей, диалоговых сценариев поддержки процесса проектирования, вводе крупных масштабов исходных данных, хранении текстовых документов. Однако они малопригодны при оперативной обработке справочных данных.

Лингвистические средства системы управления базами данных изменяются от языков программирования до языков, ориентированных на конкретного пользователя.

Основные функции СУБД:

• создание схемы базы данных;
• организация хранения данных;
• защита целостности БД;
• поддержание загрузки БД;
• предоставление пользователям доступа к БД.

Для организации межмодульного интерфейса создают информационные программы-адаптеры. В САПР, программы которых оперируют с большим числом данных (входных, промежуточных, результирующих), области обмена удобно организовать в виде некоторого банка данных. Это позволяет часть функций, выполняемых адаптером, возложить на СУБД, что в итоге сокращает время на разработку информационного и программного обеспечения. Адаптер выполняет совокупность операций по организации информационного взаимодействия между программными модулями.

Техническое обеспечение САПР. Традиционное проектирование занимает 15 % вычислительных операций. Для САПР необходимы специализированные средства, инвариантные к различным видам объектов проектирования и для решения типовых инженерных, конструкторских и технологических задач. В основном это автоматизированное рабочее место (АРМ), которое предназначено для решения сложных проектных задач в автономном режиме (для трех- и двухмерного представления объектов проектирования).

Лингвистическое обеспечение САПР. Его основу составляют специальные языковые средства (языки проектирования), предназначенные для описания процедур автоматизированного проектирования и проектных решений. Это проблемно-ориентированные языки (ПОЯ) Фортран, Си и др. Для решения геометрических задач инженерного типа ПОЯ соединяют в себе средства алгоритмического языка для решения математических задач и специальные языковые средства моделирования геометрических объектов. Создаются ПОЯ по соответствующим областям применения (строительство, электроника и т. д.). Но чрезмерное разнообразие языков затрудняет обмен средствами САПР между предприятиями. Развитие гибких производственных систем требует тщательного решения вопросов по составу лингвистического обеспечения.

Методическое обеспечение САПР — это входящие в состав системы документы, регламентирующие порядок ее эксплуатации, носят характер инструкций.

Организационное обеспечение САПР — положения, приказы, штатное расписание, квалификационные требования, регламентирующие организационную структуру подразделений с комплексом средств автоматизированного проектирования.

Уровни проектирования. При использовании блочно-иерархического подхода к проектированию представления о проектируемой системе разделяют на иерархические уровни. На верхнем уровне используют наименее детализированное представление, отражающее только самые общие черты и особенности проектируемой системы. На следующих уровнях степень подробности описания возрастает, при этом рассматривают уже отдельные блоки системы, но с учетом воздействий на каждый блок его соседей. Такой подход позволяет на каждом иерархическом уровне формулировать задачи приемлемой сложности, поддающиеся решению с помощью имеющихся средств проектирования. Разбиение на уровни должно быть таким, чтобы документация на блок любого уровня была обозрима и воспринимаема одним человеком.

Другими словами, блочно-иерархический подход есть декомпозиционный (или диакоптический) подход, который основан на разбиении сложной задачи большой размерности на последовательно и (или) параллельно решаемые группы задач малой размерности, что существенно сокращает требования к используемым вычислительным ресурсам или время решения задач.

Можно говорить не только об иерархических уровнях спецификаций, но и об иерархических уровнях проектирования, понимая под каждым из них совокупность спецификаций некоторого иерархического уровня совместно с постановками задач, методами получения описаний и решения возникающих проектных задач.

Список иерархических уровней в каждом приложении может быть специфичным, но для большинства приложений характерно следующее наиболее крупное выделение уровней:

- *системный уровень*, на котором решают наиболее общие задачи проектирования систем, машин и процессов; результаты проектирования представляют в виде структурных схем, генеральных планов, схем размещения оборудования, диаграмм потоков данных и т.п.;
- *макроуровень*, на котором проектируют отдельные устройства, узлы машин и приборов; результаты представляют в виде функциональных, принципиальных и кинематических схем, сборочных чертежей и т.п.;
- *микроуровень*, на котором проектируют отдельные детали и элементы машин и приборов.

В каждом приложении число выделяемых уровней и их наименования могут быть различными. Так, в радиоэлектронике микроуровень часто называют *компонентным*, макроуровень – *схемотехническим*. Между схемотехническим и системным уровнями вводят так называемый *функционально-логический уровень*. В вычислительной технике системный уровень подразделяют на уровни проектирования ЭВМ (вычислительных систем) и вычислительных сетей. В машиностроении различают уровни деталей, узлов, машин, комплексов.

В зависимости от последовательности решения задач иерархических уровней различают нисходящее, восходящее и смешанное проектирование (стили проектирования). Последовательность решения задач от нижних уровней к верхним характеризует *восходящее проектирование*, обратная последовательность приводит к *нисходящему проектированию*, в смешанном стиле имеются элементы как восходящего, так и нисходящего проектирования. В большинстве случаев для сложных систем предпочитают нисходящее проектирование. Однако при наличии заранее спроектированных составных блоков (устройств) можно говорить о смешанном проектировании.

Неопределенность и нечеткость исходных данных при нисходящем проектировании (так как еще не спроектированы компоненты) или исходных требований при восходящем проектировании (поскольку ТЗ имеется на всю систему, а не на ее части) обусловливают необходимость прогнозировать недостающие данные с последующим их уточнением, т.е. последовательное приближение к окончательному решению (итерационность проектирования).

Наряду с декомпозицией описаний на иерархические уровни применяют разделение представлений о проектируемых объектах на аспекты.

Аспект описания (страта) — описание системы или ее части с некоторой оговоренной точки зрения, определяемой функциональными, физическими или иного типа отношениями между свойствами и элементами.

Различают аспекты функциональный, информационный, структурный и поведенческий (процессный). Функциональное описание относят к функциям системы и чаще всего представляют его функциональными схемами. Получение функциональных описаний часто называют *функциональным проектированием*.

Информационное описание включает в себя основные понятия предметной области (сущности), словесное пояснение или числовые значения характеристик (атрибутов) используемых объектов, а также описание связей между этими понятиями и характеристиками. Информационные модели можно представлять графически (графы, диаграммы «сущность – отношение»), в виде таблиц или списков. Получение информационных описаний часто называют *информационным проектированием* или применительно к созданию баз данных — *инфологическим проектированием*.

Структурное описание относится к морфологии системы, характеризует составные части системы, их межсоединения и может быть представлено структурными схемами, а также различного рода конструкторской документацией. Получение конструкторской документации, т.е. описание геометрических форм изделий, состава компонентов и их пространственного размещения, называют *конструкторским проектированием*.

Поведенческое описание характеризует процессы функционирования (алгоритмы) системы и (или) технологические процессы ее создания. Разработка алгоритмов и программного обеспечения сис-

тем является предметом *алгоритмического проектирования*, а разработка технологических процессов изготовления изделий — предметом *технологического проектирования*.

Иногда аспекты описаний связывают с подсистемами, функционирование которых основано на различных физических процессах.

В общем случае выделение страт может быть неоднозначным. Так, помимо указанного подхода, очевидна целесообразность выделения таких аспектов, как функциональное (разработка принципов действия, структурных, функциональных, принципиальных схем), конструкторское (определение форм и пространственного расположения компонентов изделий), алгоритмическое (разработка алгоритмов и программного обеспечения) и технологическое (разработка технологических процессов) проектирование систем. Примерами страт в случае САПР могут служить также виды обеспечения автоматизированного проектирования.

Стадии проектирования – наиболее крупные части проектирования как процесса, развивающегося во времени. В общем случае выделяют стадии: *научно-исследовательских работ* (НИР), эскизного проекта или опытно-конструкторских работ (ОКР), технического, рабочего проектов, испытаний опытных образцов или опытных партий. Стадию НИР иногда называют предпроектными исследованиями или стадией технического предложения. По мере перехода от стадии к стадии степень подробности и тщательность проработки проекта возрастают, и рабочий проект уже должен быть достаточным для изготовления опытных или серийных образцов. Близким к определению стадии, но менее четко оговоренным понятием является понятие этапа проектирования. Проектирование на начальных этапах, в процессе которого принимаются принципиальные проектные решения по облику и принципам действия проектируемых устройств и систем, называют *концептуальным проектированием*.

Стадии (этапы) проектирования подразделяют на составные части – *проектные процедуры*. Примерами проектных процедур могут служить подготовка деталировочных чертежей, анализ кинематики, моделирование переходного процесса, оптимизация параметров и другие проектные задачи. В свою очередь, проектные процедуры можно расчленить на более мелкие компоненты – *проектные операции*. Например, при анализе прочности детали сеточными методами операциями могут быть построение сетки, выбор или расчет внешних воздействий, собственно моделирование полей напряжений

и деформаций, представление результатов моделирования в графической и текстовой формах. Проектирование сводится к выполнению некоторых последовательностей проектных процедур — *маршрутов проектирования*.

Стремление сократить временные затраты на проектирование привело к разработке методик *параллельного проектирования* (совмещенного проектирования), при котором параллельно во времени решаются задачи, связанные друг с другом по входным и выходным данным таким образом, что для решения одной из них требуется знание результатов решения другой задачи. Поскольку эти результаты к началу процедуры параллельного проектирования еще не получены, в методике такого проектирования должны быть указаны способы задания еще не определенных значений параметров. Примером параллельного проектирования могут служить параллельная разработка аппаратных и программных средств вычислительных систем или одновременная разработка самолета и средств его аэродромного обслуживания.

Иногда разработку технического задания на проектирование называют *внешним проектированием*, а реализацию технического задания (ТЗ) – *внутренним проектированием*.

В ТЗ на проектирование объекта указывают, по крайней мере, следующие данные:

1) назначение объекта;

2) условия эксплуатации. Наряду с качественными характеристиками (представленными в вербальной форме) имеются числовые, или *внешние,* параметры, для которых указаны области допустимых значений. Примеры внешних параметров: температура окружающей среды, внешние силы, электрические напряжения, нагрузки и т.п.;

3) требования к *выходным* параметрам, т.е. величинам, характеризующим свойства объекта, интересующие потребителя. Эти требования выражены в виде *условий работоспособности*: y_i, R, T_i, где y_i – i-й выходной параметр, $R \in \{=, <, >, \leq, \geq\}$ – вид отношения; T_i — норма i-го выходного параметра. В случае если R — отношение равенства, нужно задать требуемую точность выполнения равенства.

Примеры условий работоспособности:

– расход топлива на 100 км пробега автомобиля < 8 л;

– коэффициент усиления усилителя на средних частотах > 300;

– быстродействие процессора > 40 Мфлопс.

Классификация САПР

В области классификации САПР используется ряд устоявшихся англоязычных терминов, применяемых для классификации программных приложений и средств автоматизации САПР по отраслевому и целевому назначению.

По отраслевому назначению различают САПР:

– MCAD – автоматизированное проектирование механических устройств. Это машиностроительные САПР, применяющиеся в автомобиле- и судостроении, авиакосмической промышленности, производстве товаров народного потребления. Включают в себя разработку деталей и сборок (механизмов) с использованием параметрического проектирования на основе конструктивных элементов, технологий поверхностного и объемного моделирования (SolidWorks, Autodesk Inventor, КОМПАС, CATIA);

– EDA или ECAD – САПР электронных устройств, радиоэлектронных средств, интегральных схем, печатных плат и т. п. (Altium Designer, OrCAD);

– AEC CAD или CAAD – САПР в области архитектуры и строительства. Используются для проектирования зданий, промышленных объектов, дорог, мостов и пр. (Autodesk Architectural Desktop, AutoCAD Revit Architecture Suite, Piranesi, ArchiCAD).

По целевому назначению различают САПР или подсистемы САПР, которые обеспечивают различные аспекты проектирования [16, 17]:

• CAD — средства автоматизированного проектирования. В контексте указанной классификации термин обозначает средства САПР, предназначенные для автоматизации двумерного и/или трехмерного геометрического проектирования, создания конструкторской и/или технологической документации, и САПР общего назначения.

• CADD — проектирование и создание чертежей.

• CAGD — геометрическое моделирование.

• CAE — средства автоматизации инженерных расчётов, анализа и симуляции физических процессов. Осуществляют динамическое моделирование, проверку и оптимизацию изделий.

• CAA — подкласс средств CAE, используемых для компьютерного анализа.

• CAM — средства технологической подготовки производства изделий. Обеспечивают автоматизацию программирования и управления оборудования с ЧПУ или ГАПС (гибкие автоматизированные производственные системы). Русский аналог данного термина –

АСТПП (автоматизированная система технологической подготовки производства).

• CAPP — средства автоматизации планирования технологических процессов, применяемые на стыке систем CAD и CAM.

Многие системы автоматизированного проектирования совмещают в себе решение задач, относящихся к различным аспектам проектирования: CAD/CAM, CAD/CAE, CAD/CAE/CAM. Такие системы называют комплексными или интегрированными.

С помощью CAD-средств создаётся геометрическая модель изделия, которая используется в качестве входных данных в системах CAM и на основе которой в системах CAE формируется требуемая для инженерного анализа модель исследуемого процесса.

Стадии проектирования — это проектные исследования, техническое задание, техническое предложение, эскизный, технический и рабочий проекты, испытания и внедрение.

Стадия научно-исследовательской работы определяет назначение, принципы построения (создания) ЭС и формирует техническое задание на его проектирование.

На стадии эскизного проекта проверяется корректность и реализуемость основных принципов и положений, определяющих функционирование будущей ЭВМ, и создается эскизный проект.

На стадии технического проекта ведется всесторонняя проработка всех частей проекта, конкретизация и детализация технических решений.

На стадиях рабочего проекта, испытаний и внедрений формируется вся необходимая документация для изготовления изделия. Далее создается и испытывается опытный образец или пробная партия изделий, по результатам испытаний вносятся необходимые коррективы в проектную документацию, затем — внедрение в производство.

САПР характеризуют следующие признаки: тип, разновидность, сложность объекта проектирования; уровень, комплексность автоматизации проектирования; характер, число выпускаемых проектных документов; число уровней в структуре технического обеспечения САПР.

Тип объекта проектирования. Стандарт предусматривает деление САПР на группы:

1. САПР изделий машиностроения.

2. САПР изделий приборостроения.

3. САПР технологических процессов в машино- и приборостроении.

4. САПР программных изделий.

5. САПР организационных систем.

Разновидность объектов проектирования. Стандарт требует указания обозначений на объекты проектирования и кодирования в соответствии с системой обозначения документации на объекты, проектируемые САПР.

Сложность объектов проектирования:

1) простые — с числом составных частей до 10^2;

2) средние — до 10^3;

3) сложные — до 10^4;

4) очень сложные — до 10^5;

5) самые сложные — свыше 10^6.

Составной частью объекта проектирования, представляющего технический комплекс, является деталь (микросхема).

Уровень автоматизации проектирования:

1) низкоавтоматизированный — до 25 % проектных процедур;

2) среднеавтоматизированный — до 50 % ;

3) высокоавтоматизированный — свыше 50 %.

Чтобы отнести САПР к третьей группе, должны быть использованы методы многовариантного оптимального проектирования.

Комплексность автоматизации проектирования:

1) одноэтапные;

2) многоэтапные;

3) комплексные, т. е. автоматизация всех этапов проектирования.

Число уровней в структуре технического обеспечения:

1) *Одноуровневые комплексы технических средств (КТС)* – это компьютеры среднего класса: программная обработка данных, хранение их на основе штатного набора периферийных устройств, где единая мониторная система, банк данных и пакет прикладных программ. Терминальные микроЭВМ совместимы с основной (центральной) ЭВМ и служат либо для подготовки задач к решению на основной ЭВМ, либо для решения простых задач с помощью тех же программных и информационных средств.

2) *Двухуровневые* КТС, имеющие радиальную или кольцевую структуру (вычислительная сеть). В такой САПР функции мониторной системы и СУБД распределены по узлам вычислительной сети.

3) *Трехуровневые* САПР. Помимо двухуровневого КТС включают чертежные автоматы, комплексы для контроля программ к станкам с числовым программным управлением.

Способы организации процесса проектирования

Способ организации процесса проектирования заключается в создании модели процесса проектирования, основанной на концепции управления.

Первый вариант модели – схема процесса проектирования, которая включает в себя:

• цель проектирования, которая неизменна;

• знания технологии определенного типа для создания проекта;

• информацию (проект), которая может быть документирована и использована для производства тем или иным способом в процессе проектирования.

Второй вариант – модель процесса производства (рис. 3.2).

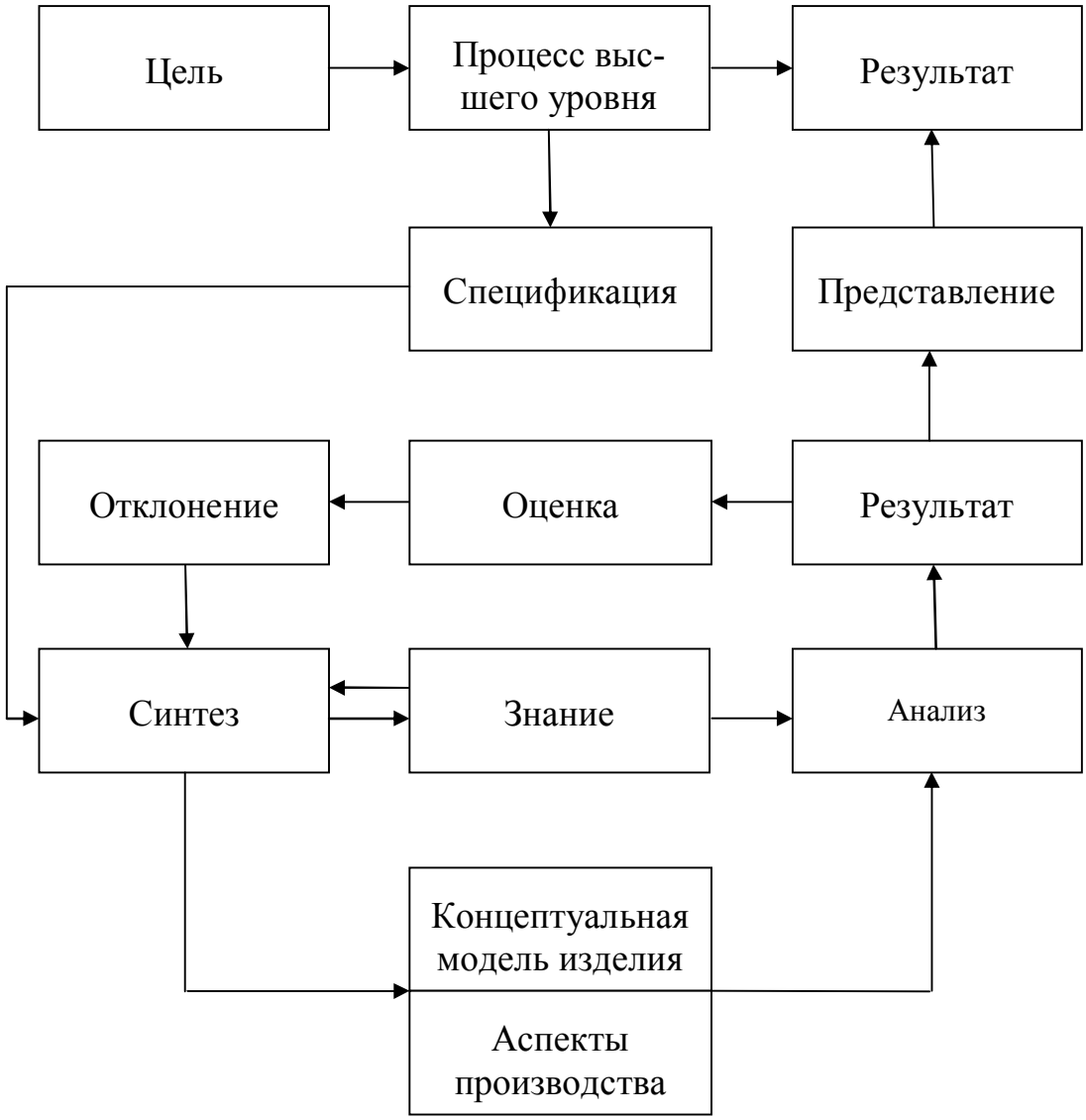


Рис. 3.2. Модель процесса производства

Если цель не достигнута, то проектные решения корректируются. Данные об отклонении предварительного проекта от спецификации передаются к операции синтеза. В среде проектирования находятся вычислительные средства, методическое обеспечение, сам проектировщик.

Проектные процедуры подразделяются на задачи анализа и синтеза (рис. 3.3). Синтез заключается в создании описания вычислительной системы, а анализ — в определении свойств и исследовании работоспособности объекта по его описанию, т. е. при синтезе создаются, а при анализе оцениваются проекты ВС. Процедуры анализа могут быть одно- и многовариантные.

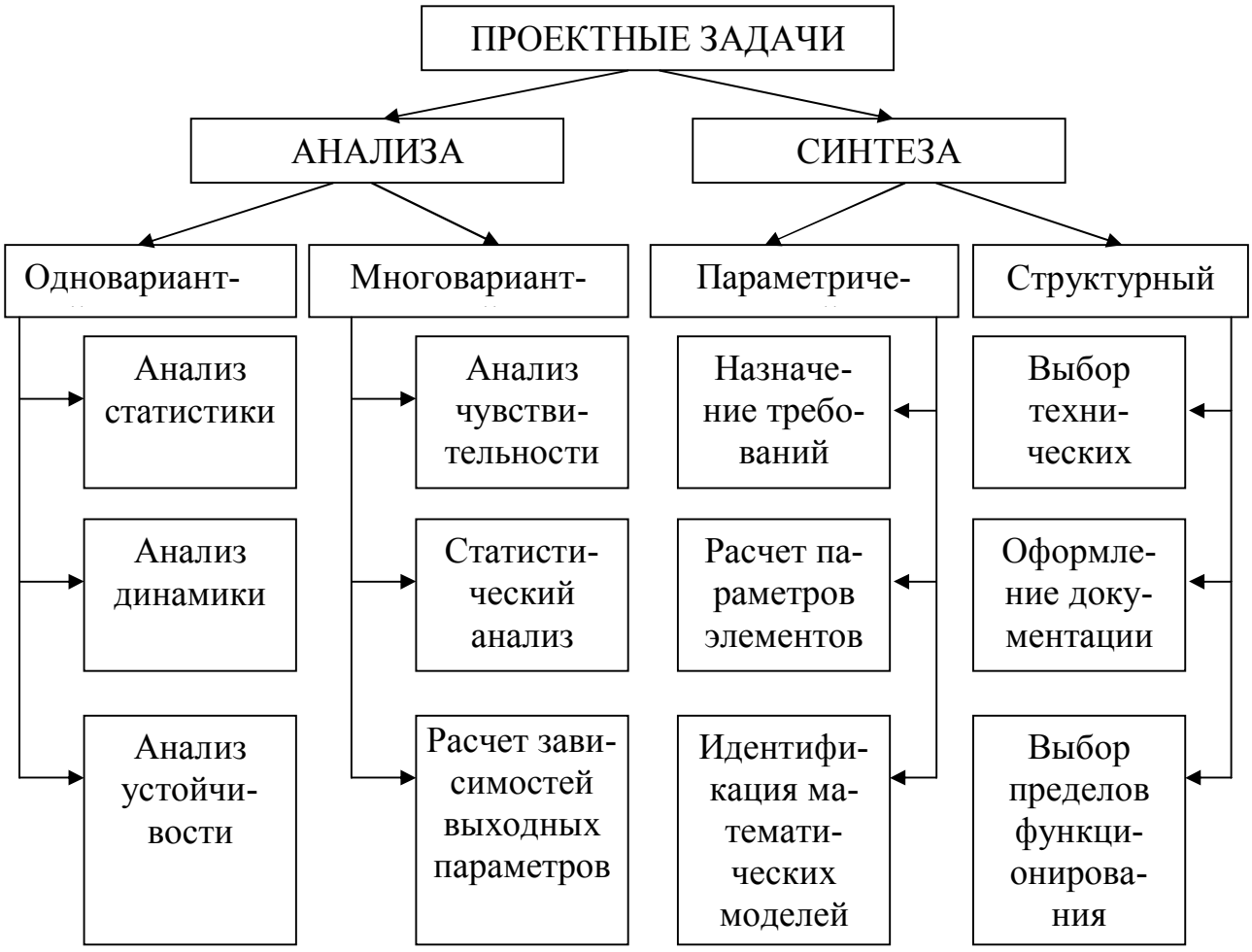


Рис. 3.3. Схема проектных процедур

Одновариантный анализ предполагает задание значений внутренних и внешних параметров и определение значений выходных параметров объекта. Задача анализа с одним вариантом сводится к однократному решению уравнений, составляющих математическую модель. *Многовариантный анализ* заключается в исследовании

свойств ВС в некоторой области пространства внутренних параметров. Такой анализ требует многократного решения систем уравнений.

Процедуры синтеза бывают параметрические и структурные. Целью *структурного синтеза* является определение структуры ВС перечня типов элементов, составляющих ВС, и способа связи элементов (оборудования) между собой в составе ВС.

После выбора исходных значений параметров элементов выполняется анализ вариантов, по результатам которого становится возможной его оценка. Последняя заключается в проверке выполнения работоспособности ВС. Если решение неудовлетворительное, то выбирается один из возможных путей улучшения проекта. Чаще изменяют числовые значения параметров элементов.

Совокупность процедур модификации параметров, анализ и оценка результатов анализа составляют *параметрический синтез.* Если ведется поиск наилучшего значения показателя качества, то процедура параметрического синтеза является процедурой *оптимизации.* Если путем параметрического синтеза не достигаются необходимые условия работоспособности, то модифицируют структуру ВС.

Новый вариант структуры синтезируется, и для него повторяются процедуры формирования модели и параметрического синтеза. Если решения нет, то корректируется ТЗ, что обусловливает итерационный характер проектирования. Взаимосвязь проектных процедур анализа и синтеза имеет характер вложенности (рис. 3.4).

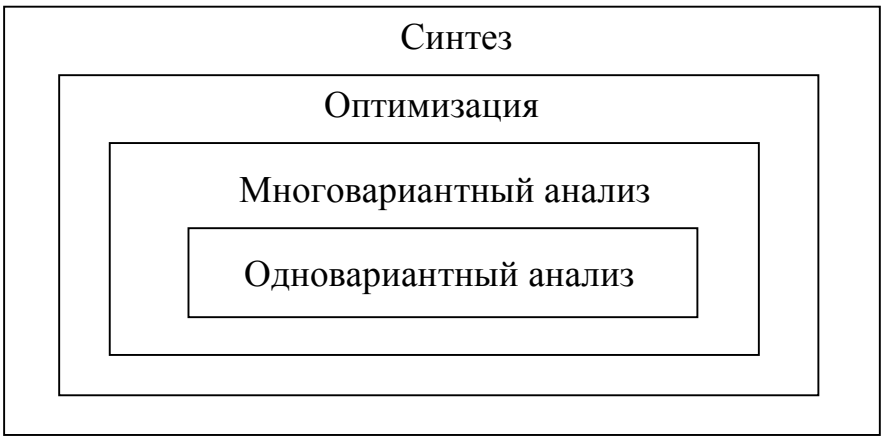


Рис. 3.4. Взаимосвязь проектных процедур

Рассмотрим последовательность этапов проектирования, т.е. маршрут проектирования СБИС (рис. 3.5). На этапе 1 выполняется синтез схемы, ее анализ с учетом предполагаемых задержек рас-

пространения сигналов в элементах. На этапе 2 осуществляется синтез принципиальных схем фрагментов СБИС, считавшихся на этапе 1 элементами.

Рис. 3.5. Маршрут проектирования СБИС

Синтез ведется на основе просмотра нескольких структур и ориентировочной оценки этих вариантов. Параллельно включается этап 7, где проектируются компоненты схемы, т. е. синтезируется физическая и топологическая структура элементов и выбирается технология изготовления СБИС.

На этапе 3 исходными данными являются:

1. Варианты структуры принципиальных схем, отобранных на этапе 2.

2. Характеристика и значения электрических параметров части компонентов, полученных на этапе 7.

Часть параметров элементов варьируется на этапе 3 с целью их оптимизации, проверяется работоспособность схем в условиях воз-

действия различных дестабилизирующих факторов. На этапе 4 синтезируется топология микросхемы, т. е. конфигурация, взаимное расположение компонентов и их соединения в полупроводниковом кристалле. Сведения о ранее спроектированной топологии отдельных компонентов поступают от этапа 7.

На этапе 5 проверяется соответствие топологии исходной принципиальной электрической схеме и соблюдение конструкторско-технологических проектных норм. На этапе 6 создаются фотошаблоны, которые содержат в себе информацию о топологии и будут использоваться в процессе изготовления СБИС. После выполнения этапов 4–6 результаты уточняются, т. е. возможен возврат к этапам 1 и 3.

3.2. Математические модели объектов проектирования электронных средств

Основные требования к математическим моделям. Методика составления математических моделей

Математическое обеспечение включает в себя математические модели (ММ). Основные требования к математическим моделям:

– *универсальность* характеризует полноту отображения в модели свойств реальной ВС (например, ММ резистора в виде уравнений закона Ома характеризует свойство резистора пропускать электрический ток, но не отражает показатели резистора как детали: его цвет, механическую прочность, стоимость и т. п.);

– *точность* ММ оценивается степенью совпадения значений параметров реального объекта и значений тех же параметров, рассчитанных с помощью оцениваемой ММ. Допустим, в ММ свойства оцениваются вектором выходных параметров $Y = (y_1, y_2, \ldots, y_m)$. Тогда, обозначив истинное и рассчитанное с помощью ММ значение j-го выходного параметра через y_{jucm} и y_{jMM}, определим относительную погрешность ε расчета параметра y_j как $\varepsilon = (y_{jMM} - y_{jucm}) / y_{jucm}$. Получена векторная оценка $\varepsilon = (\varepsilon_1 \ldots \varepsilon_m)$. Для скалярной величины используется выражение $\varepsilon = max\ \varepsilon_j$ при j, принадлежащем промежутку $[1{:}m]$;

– *адекватность* ММ — способность отображать заданные свойства объекта с погрешностью не выше заданной. Адекватность модели имеет место в ограниченной области изменения вектора внешних переменных Q, тогда области адекватности (ОА) математической модели определяются по формуле

$$ОА = \{Q/ \varepsilon_м \leq \delta\},$$

где $\delta < 0$ – заданная константа, равная предельно допустимой погрешности модели;

– *экономичность* ММ характеризуется затратами вычислительных ресурсов (машинное время и память; размерность ММ, количество параметров).

Классификация математических моделей:

– *аналитические* – в виде уравнений;

– *алгоритмические* – выражают связи выходных параметров с параметрами внутренними и внешними в форме алгоритма с численным методом решения;

– *имитационные* — алгоритмические модели, отражающие поведение ВС во времени при заданных внешних воздействиях на объект. Это модели массового обслуживания, заданные в алгоритмической форме.

Имитационное моделирование в терминах SADT-технологий: основные понятия и аналитические методы моделирования

SADT (акроним от англ. *Structured Analysis and Design Technique*) — методология структурного анализа и проектирования, интегрирующая процесс моделирования, управление конфигурацией проекта, использование дополнительных языковых средств и руководство проектом со своим графическим языком. Процесс моделирования может быть разделен на несколько этапов: опрос экспертов, создание диаграмм и моделей, распространение документации, оценка адекватности моделей и принятие их для дальнейшего использования. Этот процесс хорошо отлажен, потому что при разработке проекта специалисты выполняют конкретные обязанности, а библиотекарь обеспечивает своевременный обмен информацией. Тестирование – проверка работы системы, а установка – введение системы в действие.

SADT возникла в конце 60-х годов прошлого века в ходе революции, вызванной структурным программированием. Когда большинство специалистов билось над созданием программного обеспечения, некоторые старались разрешить более сложную задачу создания крупномасштабных систем, включающих как людей и машины, так и программное обеспечение, аналогичных системам, применяе-

мым в телефонной связи, промышленности, управлении и контроле за вооружением. В то время специалисты, традиционно занимавшиеся созданием крупномасштабных систем, стали осознавать необходимость большей упорядоченности. Таким образом, разработчики решили формализовать процесс создания системы, разбив его на следующие фазы:

- анализ — определение того, что система будет делать;
- проектирование – определение подсистем и их взаимодействие;
- реализация – разработка подсистем по отдельности, объединение – соединение подсистем в единое целое;
- эксплуатация — использование системы.

***Имитационное моделирование (ситуационное моделирование*)** позволяет строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.

Имитационное моделирование — это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, с которой проводятся эксперименты с целью получения информации об этой системе. Экспериментирование с моделью называют имитацией (имитация – это постижение сути явления, не прибегая к экспериментам на реальном объекте).

Имитационное моделирование – это частный случай математического моделирования. Существует класс объектов, для которых по различным причинам не разработаны аналитические модели либо методы решения полученной модели. В этом случае аналитическая модель заменяется имитатором или имитационной моделью.

Имитационным моделированием иногда называют получение частных численных решений сформулированной задачи на основе аналитических решений или с помощью численных методов.

Имитационная модель – логико-математическое описание объекта, которое может быть использовано для экспериментирования на компьютере в целях проектирования, анализа и оценки функционирования объекта.

К имитационному моделированию прибегают, когда:

- дорого или невозможно экспериментировать на реальном объекте;
- невозможно построить аналитическую модель: в системе есть время, причинные связи, последствие, нелинейности, стохастические (случайные) переменные;
- необходимо сымитировать поведение системы во времени.

Цель имитационного моделирования состоит в воспроизведении поведения исследуемой системы на основе результатов анализа наиболее существенных взаимосвязей между её элементами или, другими словами, разработке симулятора (англ. *simulation modeling*) исследуемой предметной области для проведения различных экспериментов.

Методика составления математической модели

Методика заключается в следующем:

1. Выбор свойств ВС, которые подлежат отображению в ММ. Этот выбор основан на анализе возможных применений модели и определяет степень ее универсальности.

2. Сбор исходной информации о выбранных свойствах объекта (опыт и знания проектировщика, научно-техническая и справочная литература, описание прототипов).

3. Синтез структуры ММ. Структура ММ — общий вид математических соотношений модели без конкретизации числовых значений параметров (в виде графа, схемы, формул).

4. Расчет числовых значений параметров ММ (минимизация погрешностей модели заданной структуры):

$$\min \varepsilon_{\text{м}} (Y) \text{ при } Y = Y_d,$$

где Y – вектор параметров модели;

Y_d – область варьирования параметров;

$\varepsilon_{\text{м}} (Y)$ – погрешность ММ.

5. Оценка адекватности ММ, т. е. сравнение расчетных значений с реальными (экспериментальными) данными.

Классификация моделей структурного синтеза – следующая:

1) перебор вариантов готовых законченных структур;

2) последовательный синтез;

3) трансформация описания разных аспектов.

Перебор вариантов. Такие структуры создаются заранее и хранятся в базе данных. Выбор варианта основан либо на случайной выборке, либо на эвристических способностях человека в диалоговом

режиме, либо на установлении корреляции параметров, характеризующих структуру. Затем выполняется оценка варианта структуры с помощью процедуры параметрического анализа и синтеза, и принимается решение на основе результатов оценки о выборе лучшей структуры из числа рассмотренных. Для такого сравнения вариантов предусматриваются некоторые критерии, объединяющие частные показатели.

Если задачу структурного синтеза удается сформулировать как задачу дискретного математического программирования, то определяется

$$extr\ F(Y) \text{ при } Y, \text{ принадлежащем } Y_d,$$

где Y_d – дискретное множество;

$F(Y)$ – целевая функция;

$Y = \{y_i\}$, y_i – элементы некоторого типа в структуре.

Последовательный синтез характеризуется поэтапным решением задачи синтеза с возможностями оценки получающихся промежуточных структур. Отмечают два способа последовательного синтеза:

– *наращивание* – поочередное добавление элементов к исходной структуре (алгоритмы компоновки и размещения, где оценкой является количество межблочных связей, и предпочтение отдается тем промежуточным вариантам, при которых большое число связей оказывается сконцентрированным в пределах одного блока);

– *выделение* – из некоторой обобщенной структуры постепенно удаляются лишние элементы (обобщенные технологические маршруты обработки деталей, печатных плат, куда включены операции, которые могут встретиться при различных сочетаниях конструктивных особенностей. Сопоставление чертежа конкретной платы и обобщенного маршрута позволяет убрать лишние операции и сформировать конкретный технологический маршрут).

Трансформация описаний разных аспектов. Например, изготовление конструкторской документации, где формализовано преобразование результатов конструкторского проектирования в графическое изображение, выполняемое по правилу проекционного черчения.

Кроме указанных процедур синтеза, существуют комбинированные, т. е. сочетание этих процедур. Например, *диалоговые,* в которых процедуры оценки выполняет ЭВМ, а принятие решения остается за человеком. Назначение ЭВМ — подсказать варианты; назначение человека — модифицировать структуру. Возможно применение

экспертных систем для генерации вариантов структуры и для связи пользователя с САПР в режиме диалога (экспертные системы воспринимают от высококвалифицированных специалистов знания, а затем используют их при решении задач структурного синтеза).

3.3. Особенности проектирования радиоэлектронных средств

Создать проект объекта (изделия или процесса) означает выбрать структуру объекта, определить значения всех его параметров и представить результаты в установленной форме. Результаты (проектная документация) могут быть выражены в виде чертежей, схем, пояснительных записок, программ для программно-управляемого технологического оборудования и других документов на бумаге или на машинных носителях информации.

Разработка (или выбор) структуры объекта есть проектная процедура, называемая *структурным синтезом*, а расчет (или выбор) значений параметров элементов X — процедура *параметрического синтеза*.

Задача структурного синтеза формулируется в системотехнике как задача принятия решений (ЗПР). Ее суть заключается в определении цели, множества возможных решений и ограничивающих условий.

Классифицируют ЗПР по ряду признаков:

- по числу критериев – задачи одно- и многокритериальные;
- по степени неопределенности:

– ЗПР детерминированные;

– ЗПР в условиях риска – при наличии в формулировке задачи случайных параметров;

– ЗПР в условиях неопределенности, т.е. при неполноте или недостоверности исходной информации.

Реальные задачи проектирования, как правило, являются многокритериальными. Одна из основных проблем постановки многокритериальных задач — установление правил предпочтения вариантов. Способы сведения многокритериальных задач к однокритериальным и последующие пути решения изучаются в дисциплинах, посвященных методам оптимизации и математическому программированию.

Наличие случайных факторов усложняет решение ЗПР. Основные подходы к решению ЗПР в условиях риска состоят или в решении «для наихудшего случая», или в учете в целевой функции мате-

матического ожидания и дисперсии выходных параметров. В первом случае задачу решают как детерминированную при завышенных требованиях к качеству решения, что является главным недостатком подхода. Во втором случае достоверность результатов решения намного выше, но возникают трудности с оценкой целевой функции. Применение метода Монте-Карло в случае алгоритмических моделей становится единственной альтернативой, и, следовательно, для решения требуются значительные вычислительные ресурсы.

Существуют две группы ЗПР в условиях неопределенности. Одна из них решается при наличии противодействия разумного противника. Такие задачи изучаются в теории игр, для задач проектирования в технике они не характерны. Во второй группе достижению цели противодействие оказывают силы природы. Для их решения полезно использовать теорию и методы нечетких множеств.

При синтезе структуры автоматизированной системы постановка задачи должна включать в качестве исходных данных следующие сведения:

- множество выполняемых системой функций (другими словами, множество работ, каждая из которых может состоять из одной или более операций); возможно, в этом множестве имеется частичная упорядоченность работ, что может быть представлено в виде ориентированного графа, в котором вершины соответствуют работам, а дуги – отношениям порядка;
- типы допустимых для использования серверов (машин), выполняющих функции системы;
- множество внешних источников и потребителей информации;
- во многих случаях задается также некоторая исходная структура системы в виде взаимосвязанной совокупности серверов определенных типов; эта структура может рассматриваться как обобщенная избыточная или как вариант первого приближения;
- различного рода ограничения, в частности, ограничения на затраты материальных ресурсов и (или) на время выполнения функций системы.

Задача заключается в синтезе (или коррекции) структуры, определении типов серверов (программно-аппаратных средств), распределении функций по серверам таким образом, чтобы достигался экстремум целевой функции при выполнении заданных ограничений.

Конструирование, разработка технологических процессов, оформление проектной документации – частные случаи структурного синтеза.

Задачу параметрического синтеза называют параметрической оптимизацией (или оптимизацией), если ее решают как задачу математического программирования:

$$\text{extr } F(\mathbf{X}),\ \mathbf{X} \in \mathbf{D}_X,$$

где $F(X)$ – целевая функция;

X – вектор управляемых параметров (называемых также проектными или варьируемыми);

$\mathbf{D}_X = \{ \mathbf{X} \mid \boldsymbol{\varphi}(\mathbf{X}) < 0,\ \boldsymbol{\psi}(\mathbf{X}) = 0 \}$ – допустимая область;

$\boldsymbol{\varphi}(\mathbf{X})$ и $\boldsymbol{\psi}(\mathbf{X})$ – функции-ограничения.

Пример 1. Электронный усилитель: управляемые параметры X (параметры резисторов, конденсаторов, транзисторов); выходные параметры Y ($f_в$ и $f_н$ — верхняя и нижняя граничные частоты полосы пропускания; K – коэффициент усиления на средних частотах; $R_{вх}$ – входное сопротивление). В качестве целевой функции $F(X)$ можно выбрать параметр $f_в$, а условия работоспособности остальных выходных параметров отнести к функциям-ограничениям.

Следующая после синтеза группа проектных процедур — процедуры *анализа*. Цель анализа — получение информации о характере функционирования и значениях выходных параметров Y при заданных структуре объекта, сведениях о внешних параметрах Q и параметрах элементов X. Если заданы фиксированные значения параметров X и Q, то имеет место процедура *одновариантного анализа*. Одновариантный анализ часто выполняется с помощью моделирования.

Моделирование состоит из этапов *формирования модели* (modeling) и *исследования модели* (решения) (simulation). В свою очередь, формирование модели включает две процедуры: во-первых, разработку моделей отдельных компонентов; во-вторых, формирование модели системы из моделей компонентов.

Первая из этих процедур выполняется предварительно по отношению к типовым компонентам вне маршрута проектирования конкретных объектов. Как правило, модели компонентов разрабатываются специалистами в прикладных областях, причем знающими требования к моделям и формам их представления в САПР. В помощь разработчику моделей в САПР предлагаются методики и вспомогательные средства, например, в виде программ анализа для экспериментальной отработки моделей. Созданные модели включаются в библиотеки моделей прикладных программ анализа.

На маршруте проектирования каждого нового объекта выполняется вторая процедура (рис. 3.6) — формирование модели системы с использованием библиотечных моделей компонентов.

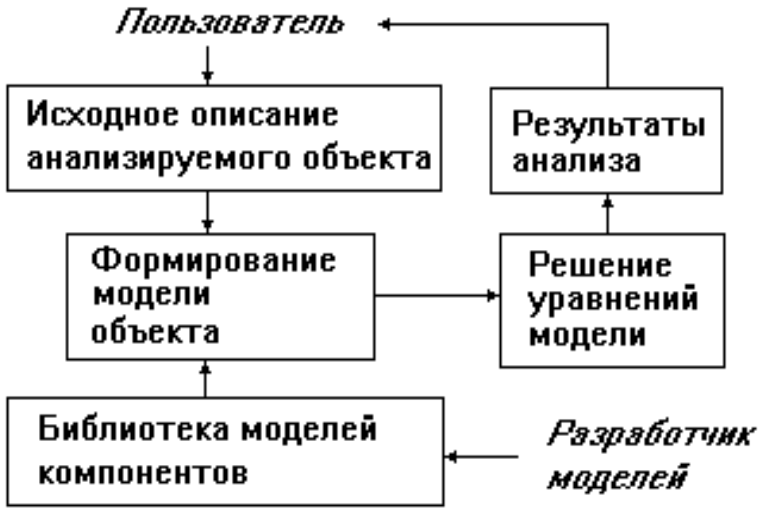


Рис. 3.6. Формирование модели системы

Как правило, эта процедура выполняется автоматически по алгоритмам, включенным в заранее разработанные программы анализа. Примеры таких программ имеются в различных приложениях и прежде всего в отраслях общего машиностроения и радиоэлектроники.

При использовании этих программ пользователь описывает исследуемый объект на входном языке программы анализа в виде не системы уравнений, которая будет получена автоматически, а списка элементов структуры, эквивалентной схемы, эскиза или чертежа конструкции.

Вторая процедура моделирования – *simulation* – сводится к решению уравнений математической модели, например системы дифференциальных уравнений, и вычислению вектора выходных параметров Y.

Если заданы статистические сведения о параметрах X и нужно получить оценки числовых характеристик распределений выходных параметров (например, оценки математических ожиданий и дисперсий), это процедура *статистического анализа*. Если требуется рассчитать матрицы абсолютной A и (или) относительной B чувствительности, имеет место задача *анализа чувствительности*.

Элемент A_{ji} матрицы A называют абсолютным *коэффициентом чувствительности*. Он представляет собой частную произ-

водную j-го выходного параметра y_j по i-му параметру x_i. Другими словами, A_{ji} является элементом вектора градиента j-го выходного параметра. На практике удобнее использовать безразмерные относительные коэффициенты чувствительности B_{ji}, характеризующие степень влияния изменений параметров элементов на изменения выходных параметров:

$$B_{ji} = \frac{A_{ji}\, x_{i\,\text{ном}}}{y_{j\,\text{ном}}},$$

где $x_{i\,\text{ном}}$ и $y_{j\,\text{ном}}$ – номинальные значения параметров x_i и y_j соответственно.

В процедурах *многовариантного анализа* определяется влияние внешних параметров, разброса и нестабильности параметров элементов на выходные параметры. Процедуры статистического анализа и анализа чувствительности — характерные примеры процедур многовариантного анализа.

Выполнение анализа и сопоставление полученных результатов с желаемыми значениями называют процедурой *верификации*.

3.4. Особенности радиоэлектронных средств как объектов автоматизированного проектирования

Радиоэлектронное средство, рассматриваемое как объект проектирования, представляет собой сложную техническую открытую систему, в основе функционирования которой лежат процессы передачи, извлечения и обработки информации, связанные с преобразованием и передачей электромагнитной энергии. Современные РЭС включают большое число элементов с различными связями между ними и обладают такими свойствами, характерными для сложных систем, как [3]:

• ярко выраженное целевое назначение;

• наличие большого числа разнообразных объектов, взаимодействующих с РЭС;

• значительные масштабы зоны действия и возможность размещения в различных средах (земная и водная поверхность, воздушное пространство, космос);

• сложность процессов обработки информации, поступающей от различных объектов, использование средств автоматизации, вычислительной техники и сетевых структур.

Наряду с этими общими свойствами РЭС обладают рядом особенностей, которые выделяют их в отдельный класс проектируемых объектов:

1. Целевое назначение РЭС предполагает обеспечение информационного взаимодействия между пространственно разнесенными объектами с использованием радиосигналов. Поэтому основными частями большинства РЭС являются подсистемы генерирования и приема электромагнитных колебаний.

2. Информационное взаимодействие между пространственно разнесенными объектами осуществляется на основе модуляции параметров радиосигнала (амплитуды, частоты, фазы) полезными сообщениями. Следовательно, важную роль в РЭС играют системы модуляции и демодуляции высокочастотных сигналов.

3. Информационный обмен между радиотехническими системами и объектами с использованием электромагнитных колебаний требует больших энергетических затрат на генерацию и излучение радиосигналов, а также обеспечение концентрации излучаемой мощности в направлении на объект (цель) или группу объектов.

При распространении электромагнитных волн в среде происходит сильное ослабление передаваемых сигналов. Поэтому в приемном тракте требуется значительное усиление принимаемых сигналов без внесения дополнительных искажений.

4. Во всех РЭС приходится решать проблемы, связанные с воздействием естественных и искусственных помех, которые приводят к искажению, а иногда и к полному подавлению полезного сигнала. Для повышения помехозащищенности применяются различные методы, реализуемые специальными устройствами, которые значительно усложняют структуру РЭС.

При проектировании РЭС используют функциональное, конструкторское, технологическое и информационное описание.

Функциональное описание характеризует эксплуатационные функции системы, обеспечивающие выполнение целевых заданий. Оно раскрывает принцип действия, свойства, протекающие в РЭС физические и информационные процессы. На основе функционального описания выполняется декомпозиция системы на элементы с различными функциями.

При разукрупнении системы используется иерархический принцип: сначала указываются обобщенные функции, затем в них выделяются обеспечивающие. В результате такой декомпозиции происходит структурирование системы по уровням иерархии в виде совокупности подсистем, которые состоят из устройств, а последние – из функциональных узлов.

Функциональный узел представляет собой функционально законченную сборочную единицу, не имеющую самостоятельного применения (например, дискриминатор, усилитель и т. п.) и в большинстве случаев выполненную на несущей конструкции. В свою очередь, в функциональном узле также можно выделить отдельные компоненты — печатные платы, микросхемы, резисторы и т.д.

Функциональное описание компонента РЭС должно содержать следующие сведения:

• основные эксплуатационные функции, называемые выходными характеристиками;

• зависимости выходных характеристик от влияющих на них параметров системы и воздействий окружающей среды;

• показатели качества работы компонента и их соответствие целям проектирования;

• разного рода ограничения на функционирование компонента.

Математически функциональное описание компонента может быть записано в виде соотношения [22]:

$$y(t, s) = f(x, u, v, t, s),$$

где y – выходная характеристика;

x – вектор режимных параметров и воздействий других компонентов;

u – вектор проектных и управляющих параметров компонента;

v – воздействия окружающей среды;

t – время;

s – пространственная координата.

Конструкторское описание дает представление о материальной реализации РЭС, отображает взаимное расположение частей проектируемого объекта, их формы, используемые материалы и т. п. При конструкторском описании РЭС выделяют следующие уровни: шкафы (стойки), блоки, модули, узлы печатные, микросборки, элементную базу [8]. На отдельных уровнях может быть дополнительное раз-

укрупнение. Объекты всех уровней обладают свойствами конструктивной и функциональной взаимозаменяемости. Выделяемые уровни конструкторского описания имеют свою несущую конструкцию, обеспечивающую устойчивость и надежность при эксплуатации.

Технологическое описание отображает методы и средства изготовления РЭС.

Особую роль при автоматизированном проектировании играет *информационное описание*, которое содержит все виды информации (документы, сведения, сообщения, сигналы) и отношения между ними.

Все описания разделяются на иерархические уровни, которые различаются степенью детализации отображаемых целей, свойств, функций и структур проектируемых объектов. Наряду с рассмотренными описаниями при проектировании РЭС используется ряд других, в частности [2]:

• первичное (исходное) описание РЭС, которое представляет собой техническое задание на проектирование, т. е. цели и задачи, решаемые системой, тактико-технические требования, условия эксплуатации и т.п.;

• промежуточные описания РЭС и его элементов, основное место в этих описаниях занимают математические и натурные (макеты) модели;

• окончательное описание разработанного объекта в виде полного комплекта технической документации (текстовый материал, схемы, чертежи и т.п.), представляемого на машинных носителях.

В ряде случаев используется термин «морфологическое описание», которое характеризует устройство объекта, его структуру, геометрию и другие сведения и формально записывается кортежем (*B, D, C*), где *B, D, C* — множества элементов (частей), связей между ними и структур системы, расположенных по ступеням иерархии соответственно [23].

В общем случае объекты автоматизированного проектирования следует рассматривать в рамках последовательности (потока) проектов.

Описание РЭС как объекта проектирования непосредственно связано с целями и процессами проектирования, оно развивается по мере выполнения этапа процесса проектирования, при этом последовательно обеспечивается соответствие всех целевых установок техническому заданию. Таким образом, имеет место триада в виде един-

ства цели, объекта и процесса проектирования, рассматриваемых в их развитии от возникновения проблемы (инициации) до окончания проектных работ.

Особенности проектирования конструкции РЭС

Конструкция РЭС, рассматриваемая как структурное образование, составленное из готовых (покупных) и вновь спроектированных частей, во многом определяет качество функционирования и конкурентоспособность разрабатываемой системы. В первую очередь это относится к надежности, технологичности, безопасности, энергетической эффективности.

Конструкции современных РЭС представляют собой сложные системы, т. е. состоят из совокупностей объектов, связей между ними, и предназначены для реализации задаваемых функций. Любая конструкция обладает системными свойствами: композиции и декомпозиции; интегративности (образование при композиции новых качеств, не свойственных исходным частям); иерархичности (деление на структурные уровни (уровни входимости), при этом элемент более низкого уровня входит в спецификацию элемента более высокого уровня).

В процессе конструирования необходимо учитывать большое число факторов, имеющих различную природу. Эти факторы можно объединить в следующие группы требований: общего характера, эксплуатационные, производственные, конструкторско-технологические, а также требования нормативно-технических документов. Кроме того, для разных уровней конструктивных компонентов (микросборки, печатные платы, функциональные модули и т.д.) имеются свои специфические требования.

Разработка конструкции РЭС, которая гарантированно удовлетворяет всем предъявляемым требованиям, представляет собой сложную инженерную задачу. При этом приходится решать проблемы, связанные с тем, что ряд требований являются противоречивыми, а некоторые предусматривают перспективу развития и функциональное наращивание системы в процессе эксплуатации, т. е. нацелены в будущее. В определенной степени противоречия возникают при стремлении повысить надежность за счет структурной избыточности и одновременно уменьшить вес и габаритные размеры или увеличить

мощность передающего устройства и улучшить показатели энергетической эффективности. Нацеленность в будущее предполагает необходимость постоянной модернизации РЭС для использования научно-технических достижений, например расширения диапазонов работы, изменения конструкции антенной системы, увеличения каналов связи и др.

Для решения этих проблем при конструировании используются методы оптимизации, принятия проектных решений в условиях неопределенности, робастного проектирования, модульного конструирования, управления проектами и др. Применение методов автоматизированного проектирования позволяет решать сложные в вычислительном отношении задачи:

• оптимальное геометрическое размещение компонентов нижестоящих уровней в монтажном пространстве вышестоящих (электрорадиоэлементы на печатных платах, печатные платы в функциональных модулях, модули в блоках и т.д.);

• обеспечение оптимального теплового режима в блоках с применением естественного, искусственного или смешанного охлаждения;

• обеспечение оптимальной помехоустойчивости функциональных узлов при воздействии естественных и искусственных помех и др.

Проектирование конструкции РЭС при наличии жестких ограничений (требований) имеет ряд особенностей:

1. Многовариантность.

Основным методом создания качественной конструкции является формирование множества альтернативных вариантов, их анализ и обоснованный выбор наиболее предпочтительного.

2. Модульность и параллельное конструирование компонентов.

Несмотря на то, что в законченном виде конструкция представляет собой единую систему, большое число ее элементов в виде блоков, функциональных модулей, печатных плат может разрабатываться параллельно, в относительной независимости друг от друга [6].

3. Типовые структуры и преемственность.

При формировании альтернативных вариантов конструкции широко используются стандартизация, типизация и унификация электронных модулей и других базовых конструкций. Это обеспечивает конструктивную совместимость, взаимозаменяемость и инвариант-

ность параметров, значительно сокращает сроки проектирования, позволяет использовать информацию о ранее разработанных конструкциях.

4. Показатели конструкции и множество состояний функционирования.

В процессе реальной эксплуатации могут изменяться климатические условия, размещение на объекте, а также другие требования к исполнению РЭС. Поэтому при разработке конструкции необходимо предусмотреть, чтобы созданное изделие не только эффективно работало при соблюдении требований к климатическому исполнению, размещению на объекте, содержащихся в техническом задании, но и обладало робастностью в различных состояниях функционирования. Для этого сопоставление альтернативных вариантов конструкции должно выполняться на множестве состояний функционирования. Данное множество наряду с состоянием нормального функционирования, когда выполняются все задаваемые требования к разрабатываемому изделию, содержит состояния, в которых изделие может находиться вследствие явлений природного характера (наводнение, ураган и т.п.), организационных и других мероприятий.

Таким образом, при анализе конструкции следует учитывать значения основных показателей в различных состояниях функционирования.

Автоматизация технологической подготовки производства РЭС

Технологическая подготовка производства (ТПП) рассматривается как совокупность современных методов организации, управления и решения технологических задач на основе комплексной стандартизации, автоматизации, экономико-математических моделей и средств технологического оснащения. Стандарты Единой системы ТПП устанавливают общие правила организации и управления производством, предусматривают широкое применение прогрессивных технологических процессов (ТП), стандартной технологической оснастки, средств автоматизации производственных процессов и управленческих работ [32].

Основные задачи автоматизации ТПП РЭС: выбор технологий и технических маршрутов для компонентов изделия; разработка принципиальной схемы ТП в виде последовательности этапов укрупненных операций; выбор технологического оборудования, оснастки и инструментов; оптимизация технологических маршрутов и опера-

ций; проектирование систем автоматического контроля и управления технологическим процессом; разработка системы менеджмента качества; оценка экономической эффективности ТП.

Большое внимание при решении этих задач уделяется непосредственному назначению РЭС, условиям эксплуатации изделий и вопросам энергосбережения. Для решения задач автоматизации ТПП широко используются методы математического моделирования, оптимизации и планирования эксперимента.

Построение математических моделей предполагает определение целей моделирования, декомпозицию ТП на отдельные операции, построение математических моделей для каждой операции, верификацию полученных моделей и композицию их в обобщенную модель всего технологического процесса. Разработанные модели используются в САПР ТП и SCADA-системах.

В качестве моделей ТП наиболее часто используются описательные (вербальные), аналитические (математические) и графовые (графические). Описательные модели представляют собой таблицы с информацией о режимах работы, технологических операциях, переходах и т. п. Эти таблицы содержатся в базах данных САПР ТП.

Аналитические модели ТП характеризуют функциональные связи между входными и выходными переменными ТП, отражают физико-химические процессы в технологических установках. Различные виды этих моделей, подробно рассматриваемые в четвертом разделе, необходимы для решения задач оптимизации режимных параметров ТП, анализа чувствительности выходных показателей ТП в случае различного рода отклонений от требуемого регламента, проектирования АСУТП и др.

Графовые модели в виде неориентированных и ориентированных графов, а также различного рода графических зависимостей между переменными ТП применяются для решения задач надежности, планирования загрузки технологического оборудования, управления запасами и т. п. В последнее время начинают использоваться модели в виде когнитивной графики, которые за счет образного представления условий решаемых задач облегчают принятие интеллектуальных решений.

Моделирование ТП во многих случаях тесно связано с моделированием проектируемых и изготовляемых объектов. Например, при

моделировании этапов технологии изготовления СБИС необходимо учитывать характеристики (статические, динамические) и компоненты (активные, паразитные), которые описываются моделями самих СБИС.

Особенностями задач оптимизации применительно к ТПП являются широкое разнообразие постановок задач, сложность их формализации, высокая размерность массивов переменных, участвующих в задаче, и различного рода ограничений, наличие неопределенностей в исходных данных, большое число возможных вариантов построения ТП, отсутствие единого подхода к решению задач.

Если проект связан с незначительной модернизацией РЭС, то большая часть технологических операций считается отработанной, а для отдельных операций требуется уточнение технологических режимов и используемых материалов. В случае серьезной модернизации изделия при формировании вариантов ТП могут использоваться процессы-аналоги с добавлением отдельных операций.

При автоматизации ТПП РЭС наиболее важными принципами являются следующие [15]:

- Принцип иерархичности (многоуровневости), в соответствии с которым работы по проектированию ТП делятся по уровням таким образом, что решения задач, получаемых на одних уровнях, служат исходными данными для задач следующих уровней.
- Принцип неокончательности (альтернативности) решений на этапах ТПП, предполагающий, что после выполнения каждого этапа для последующего проектирования остается не один, а несколько альтернативных вариантов проектных решений. Это повышает обоснованность получения наиболее предпочтительной схемы ТП.
- Принцип итерационности (обратной связи), заключающийся в возможности возврата от каждого этапа проектирования к любым предыдущим. Это позволяет использовать новую информацию, полученную в ходе выполнения работ по ТПП и анализа результатов экспериментальной проверки качества технологического процесса.

Планирование эксперимента тесно связано с методом робастного проектирования [35]. Применение этого метода позволяет разработать ТП, при котором характеристики продукции будут в наименьшей степени подвержены разбросу, вызываемому несовершенством технологического оборудования, неопределенностью сырья и влиянием различного рода внешних воздействий.

Решение задач ТПП для производства нового (инновационного) изделия целесообразно выполнять в соответствии с методологией реинжиниринга бизнес-процессов на предприятии.

3.5. Математические модели функционально-логического этапа проектирования электронных средств

Основы теории множеств

Множество – это совокупность определенных различаемых объектов, таких, что для любого объекта можно установить, принадлежит этот объект данному множеству или нет.

Список употребляемых понятий и обозначений:

$x \in A$ – элемент x принадлежит множеству A;

$x \notin A$ – элемент x не принадлежит множеству A;

$A \subset B$ – A есть подмножество множества B;

$A \subseteq B$ – «$A \subset B$ и $A \neq B$» (строгое подмножество);

$\varnothing$ – пустое множество;

$|A|$ – мощность множества (количество элементов);

$A \cup B$ – объединение множеств A и B;

$A \cap B$ – пересечение множеств;

$A \setminus B$ – разность множеств;

$A \wedge B$ – «A и B» – конъюкция высказываний A и B;

$A \vee B$ – «A или B» – дизъюкция высказываний A и B;

$A \Leftrightarrow B$ – логическая эквивалентность, A равнозначно B.

Простейшие операции над множествами:

Пусть даны множества A и B. *Пересечением* множеств A и B называется множество всех элементов, принадлежащих A и B, и обозначается $A \cap B$:

$A \cap B = \{x: x \in A \text{ и } x \in B\}$.

Объединение A и B обозначается $A \cup B$ и определяется следующим образом:

$A \cup B = \{x: x \in A \text{ или } x \in B\}$.

Разность множеств A и B (или дополнение) определяется соотношением $A \setminus B = \{x: x \in A \text{ и } x \notin B\}$ и записывается в виде $A \setminus B$.

Пустое множество есть множество, обладающее свойством $x \notin \varnothing$ при любом x.

Второе множество, определение которого зависит от задачи, называют универсальным множеством.

Универсальное множество E есть множество всех рассматриваемых в данной задаче элементов.

Два множества *не пересекаются*, если $A \cap B = \varnothing$.

В каждом случае, когда E задано, определим дополнение множества A (обозначается A') или $A' = E \setminus A = \{x: x \notin A\}$.

Из определений $\varnothing$, E, А следует справедливость тождеств:

$A \cup A' = E$, $A \cap A' = \varnothing$.

Диаграмма Венна

Пусть $E = \{b,c,d,e\}$, $A = \{b,c,d\}$, $B = \{c,e\}$. Соответствующая диаграмма изображена на рис. 3.7.

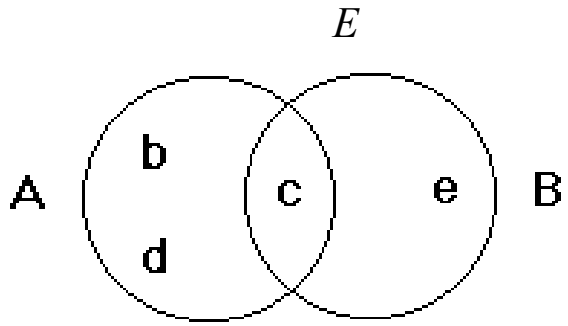


Рис. 3.7. Диаграмма Венна

Нечеткие множества

Пожалуй, наиболее поразительным свойством человеческого интеллекта является способность принимать правильные решения в обстановке неполной и нечеткой информации. Построение моделей приближенных рассуждений человека и использование их в компьютерных системах будущих поколений представляет сегодня одну из важнейших проблем науки. Значительное продвижение в этом направлении было сделано более 35 лет тому назад профессором Калифорнийского университета (Беркли) Лотфи А. Заде. Его работа «Fuzzy Sets» (1965 г.) заложила основы моделирования интеллектуальной деятельности человека и явилась начальным толчком к развитию новой математической теории.

Основанные на этой теории методы построения компьютерных нечетких систем существенно расширяют области применения компьютеров. В последнее время нечеткое управление является одной из

самых активных и результативных областей исследований применения теории нечетких множеств. Нечеткое управление особенно полезно, когда технологические процессы являются слишком сложными для анализа с помощью общепринятых количественных методов или когда доступные источники информации интерпретируются качественно, неточно или неопределенно. Экспериментально показано, что нечеткое управление дает лучшие результаты по сравнению с получаемыми при общепринятых алгоритмах управления.

Основные понятия

Пусть U – универсальное множество, x – элемент U, а R – некоторое свойство. Обычное (четкое) подмножество A универсального множества U, элементы которого удовлетворяют свойству R, определяется как множество упорядоченных пар $A = \{\mu_A(x)/x\}$, где $\mu_A(x)$ – *характеристическая функция*, принимающая значение 1, если x удовлетворяет свойству R, и 0 – в противном случае.

Нечеткое подмножество отличается от обычного тем, что для элементов x из U нет однозначного ответа «*да – нет*» относительно свойства R. В связи с этим нечеткое подмножество A универсального множества U определяется как множество упорядоченных пар $A = \{\mu_A(x)/x\}$, где $\mu_A(x)$ – *характеристическая функция принадлежности* (или просто функция принадлежности), принимающая значения в некотором вполне упорядоченном множестве M (например, $M = [0,1]$). Функция принадлежности указывает *степень* (или уровень) принадлежности элемента x подмножеству A. Множество M называют *множеством принадлежностей*. Если $M = \{0,1\}$, то нечеткое подмножество A может рассматриваться как обычное или четкое множество.

Примеры записи нечеткого множества:

Пусть $U = \{x_1, x_2, x_3, x_4, x_5\}$, $M = [0,1]$; A – нечеткое множество, для которого $\mu_A(x_1) = 0,3$; $\mu_A(x_2) = 0$; $\mu_A(x_3) = 1$; $\mu_A(x_4) = 0,5$; $\mu_A(x_5) = 0,9$.

Тогда A можно представить в виде: $A = \{0,3/x_1; 0/x_2; 1/x_3; 0,5/x_4; 0,9/x_5\}$ или

$A =$	x_1	x_2	x_3	x_4	x_5
	0,3	0	1	0,5	0,9

Основные характеристики нечетких множеств

Пусть $M = [0,1]$ и A – нечеткое множество с элементами из универсального множества U и множеством принадлежностей M.

- Величина $\sup\limits_{x\in U}\mu_A(x)$ называется *высотой* нечеткого множества *A*. Нечеткое множество *A нормально*, если его высота равна единице, т.е. верхняя граница его функции принадлежности равна единице ($\sup\limits_{x\in U}\mu_A(x)=1$). При $\sup\limits_{x\in U}\mu_A(x)<1$ нечеткое множество называется *субнормальным*.

- Нечеткое множество *пусто*, если $\forall x\in U\ \mu_A(x)=0$. Непустое субнормальное множество можно нормализовать по формуле $\mu_A(x)=\dfrac{\mu_A(x)}{\sup\limits_{x\in U}\mu_A(x)}$.

- Нечеткое множество *унимодально*, $\mu_A(x)=1$ только на одном *x* из *U*.

- *Носителем* нечеткого множества *A* является обычное подмножество со свойством $\mu_A(x)>0$, т.е. *носитель* $A=\{x/\mu_A(x)>0\}\ \forall x\in U$.

- Элементы $x\in U$, для которых $\mu_A(x)=0{,}5$, называются *точками перехода* множества *A*.

Примеры:

а) Пусть $U=\{0,1,2,..,10\}$, $M=[0,1]$. Нечеткое множество *«несколько»* можно определить следующим образом:

«*несколько*» = $\{0{,}5/3;0{,}8/4;1/5;1/6;0{,}8/7;0{,}5/8\}$; его характеристики: *высота* = 1, *носитель* = $\{3,4,5,6,7,8\}$, *точки перехода* $\{3, 8\}$.

б) Пусть $U=\{0,1,2,3,...,n,...\}$. Нечеткое множество «*малый*» можно определить:

$$\text{«}малый\text{»}=\left\{\mu_{\text{"}малый\text{"}}(n)=\frac{1}{1+\left(\frac{n}{10}\right)^2}/n\right\}.$$

в) Пусть $U=\{1,2,3,...,100\}$ и соответствует понятию «*возраст*», тогда нечеткое множество «*молодой*» может быть определено с помощью

$$\mu_{\text{"}молодой\text{"}}(x)=\begin{cases}1, & x\in[1,25]\\ \dfrac{1}{1+\left(\dfrac{x-25}{5}\right)^2}, & x\ge 25.\end{cases}$$

г) Пусть $U=\{$*Запорожец, Жигули, Мерседес*,..$\}$ – множество марок автомобилей, а $U'=[0,\infty)$ – универсальное множество «*стоимость*», тогда на U' можно определить нечеткие множества типа:

«*для бедных*», «*для среднего класса*», «*престижные*» с функциями принадлежности (рис. 3.8).

Рис. 3.8. Графическое представление множества марок автомобилей

Имея эти функции и зная стоимость автомобилей из U в данный момент времени, можно определить на U' нечеткие множества с этими же названиями. Так, например, нечеткое множество «*для бедных*», заданное на универсальном множестве $U = \{$*Запорожец, Жигули, Мерседес*,..$\}$, показано на рис. 3.9.

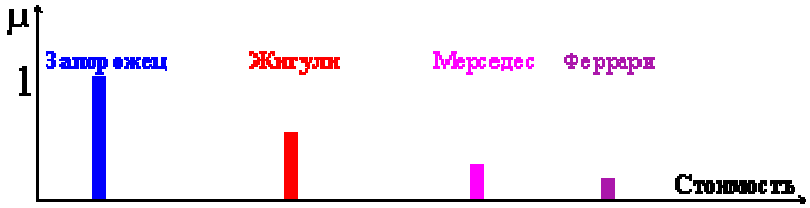


Рис. 3.9. Нечеткое множество «автомобиль для бедных»

Аналогично можно определить нечеткое множество «*скоростные*», «*средние*», «*тихоходные*» и т. д.

Методы построения функций принадлежности нечетких множеств

В приведенных выше примерах использованы *прямые* методы, когда эксперт либо просто задает для каждого $x \in U$ значение $\mu_A(x)$, либо определяет функцию совместимости. Как правило, прямые методы задания функции принадлежности используются для измеримых понятий, таких как скорость, время, расстояние, давление, температура и т. д., или когда выделяются полярные значения.

Во многих задачах при характеристике объекта можно выделить набор признаков и для каждого из них определить полярные значения, соответствующие значениям функции принадлежности, – 0 или 1.

Например, в задаче распознавания лиц можно выделить следующие шкалы:

Обозначение	Признак	0	1
x_1	высота лба	низкий	широкий
x_2	профиль носа	курносый	горбатый
x_3	длина носа	короткий	длинный
x_4	разрез глаз	узкие	широкие
x_5	цвет глаз	светлые	темные
x_6	форма подбородка	остроконечный	квадратный
x_7	толщина губ	тонкие	толстые
x_8	цвет лица	темный	светлый
x_9	очертание лица	овальное	квадратное

Для конкретного лица A эксперт, исходя из приведенной шкалы, задает $\mu_A(x) \in [0,1]$, формируя векторную функцию принадлежности $\{\mu_A(x_1), \mu_A(x_2),... \mu_A(x_9)\}$.

При *прямых* методах используются также групповые прямые методы, когда, например, группе экспертов предъявляют конкретное лицо и каждый должен дать один из двух ответов: «*этот человек лысый*» *или* «*этот человек не лысый*». Тогда количество утвердительных ответов, деленное на общее число экспертов, дает значение $\mu_{\text{«лысый»}}$ (данного лица).

Косвенные методы определения значений функции принадлежности используются в случаях, когда отсутствуют элементарные измеримые свойства, через которые определяется интересующее нас нечеткое множество. Как правило, это методы попарных сравнений. Если бы значения функций принадлежности были известны, например: $\mu_A(x_i) = w_i$, $i = 1,2,...,n$, то попарные сравнения можно представить матрицей отношений $A = \{a_{ij}\}$, где $a_{ij} = w_i/w_j$ (операция деления).

На практике эксперт сам формирует матрицу A, при этом предполагается, что диагональные элементы равны единице, а для элементов симметричных относительно диагонали $a_{ij} = 1/a_{ij}$, т.е. если один элемент оценивается в α раз сильнее, чем другой, то этот последний должен быть в $1/\alpha$ раз сильнее, чем первый. В общем случае задача сводится к поиску вектора w, удовлетворяющего уравнению вида $A_{\boldsymbol{w}} = \lambda_{\max} w$, где λ_{max} – наибольшее собственное значение матрицы A. Поскольку матрица A положительна по построению, решение данной задачи существует и является положительным.

Операции над нечеткими множествами:

1) *Включение.* Пусть A и B – нечеткие множества на универсальном множестве U. Говорят, что A содержится в B, если $\forall x \in U \ \mu_A(x) \le \mu_B(x)$.

Обозначение: $A \subset B$. Иногда используют термин «*доминирование*», т.е. в случае, когда $A \subset B$, говорят, что B доминирует A.

2) *Равенство*. A и B равны, если $\forall x \in U \ \mu_A(x) = \mu_B(x)$.

Обозначение: $A = B$.

3) *Дополнение*. Пусть M = [0,1], A и B – нечеткие множества, заданные на U. A и B дополняют друг друга, если $\forall x \in U \ \mu_A(x) = 1 - \mu_B(x)$.

Обозначение: $B = \bar{A}$ или $A = \overline{B}$. Очевидно, что $(\overline{\overline{A}}) = A$. (Дополнение определено для M = [0,1], но очевидно, что его можно определить для любого упорядоченного M).

4) *Пересечение*. $A \cap B$ – наименьшее нечеткое подмножество, содержащееся одновременно в A и B, $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$.

5) *Объединение*. $A \cup B$ – наибольшее нечеткое подмножество, включающее как A, так и B, с функцией принадлежности $\mu_{A \cup B}(x) = \min(\mu_A(x), \mu_B(x))$.

6) *Относительное дополнение*. $A \setminus B = A \cap \overline{B}$ с функцией принадлежности: $\mu_{A \setminus B}(x) = \mu_{A \cap \overline{B}}(x) = \min(\mu_A(x), 1 - \mu_B(x))$.

7) *Дизъюнктивная сумма*. $A \oplus B = (A \setminus B) \cup (B \setminus A) = (A \cup \overline{B}) \cup (\overline{A} \cap B)$

с функцией принадлежности

$$\mu_{A \setminus B}(x) = \max\{[\min(\mu_A(x), 1 - \mu_B(x))], [\min(1 - \mu_A(x), \mu_B(x))]\}$$

Примеры:

Пусть даны нечеткие множества:

$A = \{0{,}4/x_1; 0{,}2/x_2; 0/x_3; 1/x_4\}$;

$B = \{0{,}7/x_1; 0{,}9/x_2; 0{,}1/x_3; 1/x_4\}$;

$C = \{0{,}1/x_1; 1/x_2; 0{,}2/x_3; 0{,}9/x_4\}$.

а) $A \subset B$, т. е. A содержится в B или B доминирует A, C *не сравнимо* ни с A, ни с B, т.е. пары $\{A, C\}$ и $\{B, C\}$ – пары недоминируемых нечетких множеств. $A \neq B \neq C$.

б) $\overline{A} = \{0{,}6/x_1; 0{,}8/x_2; 1/x_3; 0/x_4\}$;

$\overline{B} = \{0{,}3/x_1; 0{,}1/x_2; 0{,}9/x_3; 0/x_4\}$.

в) $A \cap B = \{0{,}4/x_1; 0{,}2/x_2; 0/x_3; 1/x_4\}$.

г) $A \cup B = \{0{,}7/x_1; 0{,}9/x_2; 0{,}1/x_3; 1/x_4\}$.

д) $A - B = A \cap \overline{B} = \{0{,}3/x_1; 0{,}1/x_2; 0/x_3; 0/x_4\}$;

$B - A = \overline{A} \cap B = \{0{,}6/x_1; 0{,}8/x_2; 0{,}1/x_3; 0/x_4\}$.

е) $A \oplus B = \{0{,}6/x_1; 0{,}8/x_2; 0{,}1/x_3; 0/x_4\}$.

8) *Наглядное представление операций над нечеткими множествами.*

Для нечетких множеств можно строить визуальное представление. Рассмотрим прямоугольную систему координат, на оси ординат которой откладываются значения $\mu_A(x)$, на оси абсцисс в произвольном порядке расположены элементы U (такое представление использовано в примерах нечетких множеств). Если U по своей природе упорядочено, то этот порядок желательно сохранить в расположении элементов на оси абсцисс. Такое представление делает наглядными простые операции над нечеткими множествами (рис. 3.10).

На верхней части рисунка заштрихованная часть соответствует нечеткому множеству A и, если говорить точно, изображает область значений A и всех нечетких множеств, содержащихся в A. На нижней части даны $\bar{A}$, $A \cap \bar{A}$, $A \cup \bar{A}$.

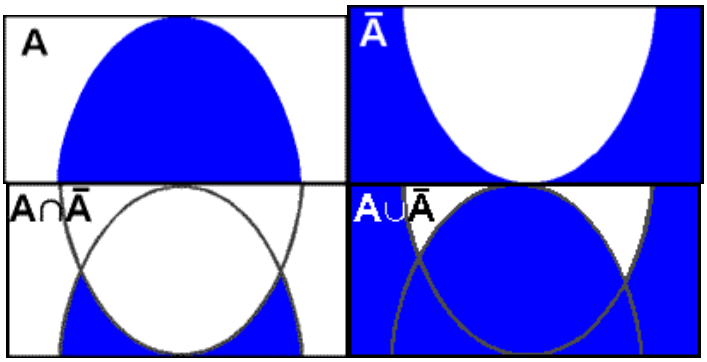


Рис. 3.10. Графическое представление операций над нечеткими множествами

9) *Свойства операций* $\cup$ *и* $\cap$.

Пусть A, B, C – нечеткие множества, тогда выполняются следующие свойства:

- $\left.\begin{array}{l} A \cap B = B \cap A \\ A \cup B = B \cup A \end{array}\right\}$ – коммутативность;
- $\left.\begin{array}{l} (A \cap B) \cap C = A \cap (B \cap A) \\ (A \cup B) \cup C = A \cup (B \cup A) \end{array}\right\}$ – ассоциативность;
- $\left.\begin{array}{l} A \cap A = A \\ A \cup A = A \end{array}\right\}$ – идемпотентность;
- $\left.\begin{array}{l} A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \\ A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \end{array}\right\}$ – дистрибутивность;
- $A \cup \varnothing = A$, где $\varnothing$ – пустое множество, т.е. $\mu_{\varnothing}(x) = 0$; $\forall > x \in U$;

- $A \cap \varnothing = \varnothing$;
- $A \cap U = A$, где U – универсальное множество;
- $A \cup U = U$;
- $\left.\begin{array}{l} \overline{A \cap B} = \overline{A} \cup \overline{B} \\ \overline{A \cup B} = \overline{A} \cap \overline{B} \end{array}\right\}$ – теоремы де Моргана.

В отличие от четких множеств для нечетких множеств в общем случае

$A \cap \bar{A} \neq \varnothing$;

$A \cup \bar{A} \neq U$.

Введенные выше операции над нечеткими множествами основаны на использовании операций *max* и *min*. В теории нечетких множеств разрабатываются вопросы построения обобщенных, параметризованных операторов пересечения, объединения и дополнения, позволяющих учесть разнообразные смысловые оттенки соответствующих им связок «*и*», «*или*», «*не*».

10) Один из подходов к операторам пересечения и объединения заключается в их определении в *классе треугольных норм и конорм*.

а) *Треугольной нормой* (*t-нормой*) называется двуместная действительная функция T, удовлетворяющая следующим условиям:

- $T(0,0) = 0$; $T(\mu_A, 1) = \mu_A$; $T(1, \mu_A) = \mu_A$ – ограниченность;
- $T(\mu_A, \mu_B) \le T(\mu_C, \mu_D)$, если $\mu_A \le \mu_C$, $\mu_B \le \mu_D$ – монотонность;
- $T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$ – коммутативность;
- $T(\mu_A, T(\mu_B, \mu_C)) = T(T(\mu_A, \mu_B), \mu_C)$ – ассоциативность.

Простым случаем треугольных норм являются:

$\min(\mu_A,\mu_B)$;

произведение $\mu_A \cdot \mu_B$;

$\max(0, \mu_A + \mu_B - 1)$.

б) *Треугольной конормой* (*t-конормой*) называется двуместная действительная функция со свойствами:

- $T(1,1) = 1$; $T(\mu_A, 0) = \mu_A$; $T(0, \mu_A) = \mu_A$ – *ограниченность*;
- $T(\mu_A, \mu_B) \ge T(\mu_C, \mu_D)$, если $\mu_A \ge \mu_C$, $\mu_B \ge \mu_D$ – *монотонность*;
- $T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$ – *коммутативность*;
- $T(\mu_A, T(\mu_B, \mu_C)) = T(T(\mu_A, \mu_B), \mu_C)$ – *ассоциативность*.

Примеры *t*-конорм:

$\max(\mu_A,\mu_B)$;

$\mu_A+\mu_B-\mu_A\cdot\mu_B$;

$\min(1, \mu_A+\mu_B)$.

Алгебраические операции над нечеткими множествами

а) *Алгебраическое произведение A* и *B* обозначается $A\cdot B$ и определяется следующим образом:

$\forall x\in U;\ \mu_{A\cdot B}(x) = \mu_A(x)\cdot\mu_B(x)$.

б) *Алгебраическая сумма* этих множеств обозначается $A \hat{+} B$ и определяется как:

$\forall x\in U;\ \mu_{A\hat{+}B} = \mu_A(x) + \mu_B - \mu_A(x)\mu_B(x)$.

Для операций $\{\cdot, \hat{+}\}$ выполняются следующие свойства:

- $\left.\begin{array}{l} A\cdot B = B\cdot A \\ A\hat{+}B = B\hat{+}A \end{array}\right\}$ – коммутативность;
- $\left.\begin{array}{l} (A\cdot B)\cdot C = A\cdot(B\cdot A) \\ (A\hat{+}B)\hat{+}C = A\hat{+}(B\hat{+}A) \end{array}\right\}$ – ассоциативность;
- $A\cdot\varnothing=\varnothing$, $A\hat{+}\varnothing=A$, $A\cdot U=A$, $A\hat{+}U=U$;
- $\left.\begin{array}{l} \overline{A\cdot B} = \overline{A}\hat{+}\overline{B} \\ \overline{A\hat{+}B} = \overline{A}\cdot\overline{B} \end{array}\right\}$ – теоремы де Моргана;
- $\left.\begin{array}{l} A\cdot A\neq A \\ A\hat{+}A\neq A \end{array}\right\}$ – идемпотентность;
- $\left.\begin{array}{l} A\cdot(B\hat{+}C)\neq(A\cdot B)\hat{+}(A\cdot C) \\ A\hat{+}(B\cdot C)\neq(A\hat{+}B)\cdot(A\hat{+}C) \end{array}\right\}$ – дистрибутивность;
- $A\cdot\bar{A}\neq\varnothing,\ A\hat{+}\overline{A}\neq U$.

Для примера докажем закон де Моргана: $\overline{A\cdot B} = \overline{A}\hat{+}\overline{B}$. Обозначим $\mu_A(x)$ через *a*, $\mu_B(x)$ через *b*. Тогда в левой части закона де Моргана имеем: $1-ab$, а в правой:

$$(1-a)+(1-b)-(1-a)(1-b) = 1-a+1-b+a+b-ab = 1-ab\,.$$

Таким образом, левая и правая части совпадают.

Докажем, что свойство дистрибутивности не выполняется, т. е. $A\cdot(B\hat{+}C)\neq(A\cdot B)\hat{+}(A\cdot C)$. Для левой части имеем: $a(b+c-bc) = ab+ac-abc$; для правой: $ab+ac\text{-}(ab)(ac) = ab+ac+a^2bc$. Это означает, что дистрибутивность не выполняется при $a \neq a^2$.

При совместном использовании операций $\{\cup, \cap, +, \cdot\}$ выполняются следующие свойства:

- $A \cdot (B \cup C) = (A \cdot B) \cup (A \cdot C)$;
- $A \cdot (B \cap C) = (A \cdot B) \cap (A \cdot C)$;
- $A \hat{+} (B \cup C) = (A \hat{+} B) \cup (A \hat{+} C)$;
- $A \hat{+} (B \cap C) = (A \hat{+} B) \cap (A \hat{+} C)$.

г) На основе операции алгебраического произведения (по крайней мере, для целых α эта основа очевидна) определяется операция *возведения в степень* α нечеткого множества A, где α – положительное число (рис. 3.11). Нечеткое множество A^{α} устанавливается функцией принадлежности $\mu_A{}^{\alpha}$. Частным случаем возведения в степень являются:

- CON(A) = A^2 – операция *концентрирования*;
- DIL(A) = $A^{0,5}$ – операция *растяжения*,

которые используются при работе с лингвистическими неопределенностями.

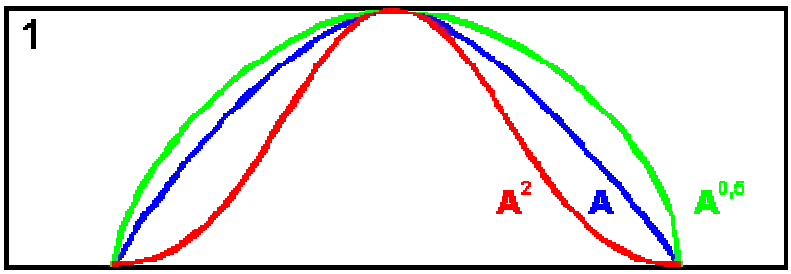


Рис. 3.11. Графическое представление операции возведения в степень

д) *Умножение на число.* Если α – положительное число, то нечеткое множество αA имеет функцию принадлежности

$\mu_{\alpha A}(x) = \alpha \mu_A(x)$.

е) *Выпуклая комбинация нечетких множеств.* Пусть $A_1, A_2, ..., A_n$ – нечеткие множества универсального множества U, а ω_1, ω_2, ..., ω_n – неотрицательные числа, сумма которых равна единице. Выпуклой комбинацией $A_1, A_2, ..., A_n$ называется нечеткое множество A с функцией принадлежности

$\forall x \in U \ \mu_A(x_1, x_2, ..., x_n) = \omega_1 \mu_{A_1}(x) + \omega_2 \mu_{A_2}(x) + ... + \omega_n \mu_{A_n}(x)$.

ж) *Декартово произведение нечетких множеств.* Пусть $A_1, A_2, ..., A_n$ – нечеткие подмножества универсальных множеств $U_1, U_2, ..., U_n$ соответственно. Декартово произведение

$A = A_1 \times A_2 \times ... \times A_n$ является нечетким подмножеством множества $U = U_1 \times U_2 \times ... \times U_n$ с функцией принадлежности

$$\mu_A(x_1, x_2, ..x_n) = \min\{\mu_{A_1}(x_1), \mu_{A_2}(x_2), ..., \mu_{A_n}(x_n)\}.$$

з) *Оператор увеличения нечеткости.* Используется для преобразования четких множеств в нечеткие и для увеличения нечеткости нечеткого множества.

Пусть A – нечеткое множество, U – универсальное множество, и для всех $x \in U$ определены нечеткие множества $K(x)$. Совокупность всех $K(x)$ называется ядром оператора увеличения нечеткости Ф. Результатом действия оператора Ф на нечеткое множество A является нечеткое множество вида

$$\Phi(A, K) = \bigcup_{x \in U} \mu_A(x) K(x),$$

где $\mu_A(x)K(x)$ – произведение числа на нечеткое множество.

Пример:

$U = \{1,2,3,4\};$

$A = \{0,8/1;0,6/2;0/3;0/4\};$

$K(1) = \{1/1;0,4/2\};$

$K(2) = \{1/2;0,4/1;0,4/3\};$

$K(3) = \{1/3;0,5/4\};$

$K(4) = \{1/4\}.$

Тогда

$$\Phi(A,K) = \mu_A(1)K(1) \cup \mu_A(2)K(2)\mu_A(3)K(3)\mu_A(4)K(4) =$$

$$= 0,8\{1/1;0,4/2\} \cup 0,6\{1/2;0,4/1;0,4/3\} = \{0,8/1;0,6/2;0,24/3\}.$$

и) *Четкое множество α-уровня (или уровня α).* Множеством α- уровня нечеткого множества A универсального множества U называется четкое подмножество A_α универсального множества U, определяемое в виде

$A_\alpha = \{x / \mu_A(x) \geq \alpha\}$, где $\alpha \leq 1$.

Пример: $A = \{0,2/x_1;0/x_2;0,5/x_3;1/x_4\}$,

тогда $A_{0,3} = \{x_3, x_4\}$, $A_{0,7} = \{x_4\}$.

Достаточно очевидное свойство: если $\alpha_1 \geq \alpha_2$, то $A_{\alpha_1} \leq A_{\alpha_2}$.

к) *Теорема о декомпозиции.* Всякое нечеткое множество A разложимо по его множествам уровня в виде $A = \bigcup_{x \in M} \alpha A_\alpha$, где αA_α – произведение числа α на множество A, и α «пробегает» область значений M функции принадлежности нечеткого множества A.

Пример: $A=\{0{,}1/x_1;0/x_2;0{,}7/x_3;1/x_4\}$ представимо в виде

$A=0{,}1(1,0,1,1)\cup 0{,}7(0,0,1,1)\cup 1(0,0,0,1)=\{0{,}1/x_1;0/x_2;0{,}1/x_3;0{,}1/x_4\}\cup$

$\cup\{0/x_1;0/x_2;0{,}7/x_3;0{,}7/x_4\}\cup\{0/x_1;0/x_2;0/x_3;1/x_4\}=$

$=\{0{,}1/x_1;0/x_2;0{,}7/x_3;1/x_4\}.$

Если область значений функции принадлежности состоит из n градаций $\alpha_1\le\alpha_2\le\alpha_3\le\ldots$, то A (при фиксированных значениях градаций) представимо в виде

$$A=\bigcup_{i=1}^{n}\alpha_i A_{\alpha_i},$$

т. е. определяется совокупностью обычных множеств $\{A_{\alpha_1},A_{\alpha_2},\ldots,A_{\alpha_n}\}$, где $A_{\alpha_1}\ge A_{\alpha_2}\ge\ldots\ge A_{\alpha_n}$.

Расстояние между нечеткими множествами, индексы нечеткости

Пусть A и B – нечеткие подмножества универсального множества U. Введем понятие *расстояния* $\rho(A,B)$ между нечеткими множествами. При введении расстояния предъявляются следующие требования:

а) $\rho(A,B)\ge 0$ – неотрицательность;

б) $\rho(A,B)=\rho(A,B)$– симметричность;

в) $\rho(A,B)<\rho(A,C)+\rho(C,B)$.

К этим трем требованиям можно добавить четвертое: $\rho(A,A)=0$.

Определим следующие расстояния по формулам:

1) *Расстояние Хемминга* (или *линейное расстояние*):

$$\rho(A,B)=\sum_{i=1}^{n}\left|\mu_A(x_i)-\mu_B(x_i)\right|.$$

Очевидно, что $\rho(A,B)\in[0,n]$.

2) *Евклидово, или квадратичное, расстояние*:

$$\varepsilon(A,B)=\sqrt{\sum_{i=1}^{n}\left(\mu_A(x_i)-\mu_B(x_i)\right)^2},\quad \varepsilon(A,B)\in\left[0,\sqrt{n}\right].$$

3) *Относительное расстояние Хемминга*:

$$\rho(A,B)=\frac{1}{n}\sum_{i=1}^{n}\left|\mu_A(x_i)-\mu_B(x_i)\right|,\quad \rho(A,B)\in[0,1].$$

4) *Относительное евклидово расстояние*:

$$\varepsilon(A,B)=\frac{1}{\sqrt{n}}\sqrt{\sum_{i=1}^{n}\left(\mu_A(x_i)-\mu_B(x_i)\right)^2},\quad \varepsilon(A,B)\in[0,1].$$

Расстояние Хемминга и квадратичное расстояние в случае, когда U бесконечно, определяются аналогично с условием сходимости соответствующих сумм: если U – счетное, то

$$\rho(A,B)=\sum_{i=1}^{\infty}\left|\mu_A(x_i)-\mu_B(x_i)\right|;$$

$$\varepsilon(A,B)=\sqrt{\sum_{i=1}^{\infty}\left(\mu_A(x_i)-\mu_B(x_i)\right)^2};$$

если $U=R$ (числовая ось), то

$$\rho(A,B)=\int_{-\infty}^{\infty}\left|\mu_A(x_i)-\mu_B(x_i)\right|dx;$$

$$\varepsilon(A,B)=\sqrt{\int_{-\infty}^{\infty}\left(\mu_A(x_i)-\mu_B(x_i)\right)^2 dx}.$$

Здесь приведены два наиболее часто встречающихся определения понятия расстояния. Разумеется, для нечетких множеств можно ввести и другие определения этого понятия.

5) Перейдем к *индексам нечеткости,* или *показателям размытости,* нечетких множеств. Если объект x обладает свойством R (порождающим нечеткое множество A) лишь в частной мере, т. е. $0<\mu_A(x)<1$, то внутренняя неопределенность, двусмысленность объекта x в отношении R проявляется в том, что он, хотя и в разной степени, принадлежит сразу двум противоположным классам: классу объектов, «обладающих свойством R», и классу объектов, «не обладающих свойством R». Эта двусмысленность максимальна, когда степени принадлежности объекта обоим классам равны: $\mu_A(x)=\mu_{\bar{A}}(x)=0{,}5$, и минимальна, когда объект принадлежит только одному классу: либо $\mu_A(x)=1$ и $\mu_{\bar{A}}(x)=0$, либо $\mu_A(x)=0$ и $\mu_{\bar{A}}(x)=1$.

В общем случае показатель размытости нечеткого множества можно определить в виде функционала $d(A)$ со значениями в R (положительная полуось), удовлетворяющего условиям:

$d(A)=0$ тогда и только тогда, когда A – обычное множество;

$d(A)$ максимально тогда и только тогда, когда $\mu_A(x)=0{,}5$ для всех $x\in U$.

$\mu_A(x)\le\mu_B(x)$ при $\mu_A(x)<0{,}5$;

$\mu_A(x)\ge\mu_B(x)$ при $\mu_A(x)>0{,}5$;

$\mu_A(x)$ – любое при $\mu_B(x)=0{,}5$.

$d(A)=d(\bar{A})$ – симметричность по отношению к 0,5.

$d(A \cup B) + d(A \cap B) = d(A) + d(B)$.

Рассмотрим *индексы нечеткости* (показатели размытости), которые можно определить, используя понятие расстояния.

6) *Обычное множество, ближайшее к нечеткому*.

Пусть A – нечеткое множество. *Вопрос:* какое обычное множество $A \subset U$ является ближайшим к A, т. е. находится на наименьшем евклидовом расстоянии от нечеткого множества A. Таким подмножеством, обозначаемым $\underline{A}$, является подмножество с характеристической функцией

$$\mu_{\underline{A}}(x_i) = \begin{cases} 0, & \text{если } \mu_A(x_i) < 0{,}5 \\ 1, & \text{если } \mu_A(x_i) > 0{,}5 \\ 0 \text{ или } 1, & \text{если } \mu_A(x_i) = 0{,}5. \end{cases}$$

Как правило, принимают $\mu_A(x_i) = 0$, если $\mu_A(x_i) = 0{,}5$.

Используя понятие обычного множества, ближайшего к нечеткому, введем следующие индексы нечеткости нечеткого множества A.

7) *Линейный индекс нечеткости*: $d(A) = \frac{2}{n}\rho(A, \underline{A})$. Здесь $\rho(A, \underline{A})$ – линейное (хеммингово) расстояние, множитель $\frac{2}{n}$ обеспечивает выполнение условия $0 \le d(A) \le 1$.

8) *Квадратичный индекс нечеткости*: $d(A) = \frac{2}{\sqrt{n}}\varepsilon(A, \underline{A})$, $0 \le d(A) \le 1$. Здесь $\varepsilon(A, \underline{A})$ – квадратичное (евклидово) расстояние.

9) Вводим линейный и квадратичный индексы нечеткости, используя понятие расстояния и понятие обычного множества, ближайшего к нечеткому. Эти же индексы можно определить, используя операцию дополнения, следующим образом:

$d(A) = \frac{2}{n}\sum_{i=1}^{n} \min\left(\mu_A(x_i), \mu_{\bar{A}}(x_i)\right)$ – линейный индекс;

$d(A) = \frac{2}{\sqrt{n}}\sqrt{\sum_{i=1}^{n} \min\left(\mu_A^2(x_i), \mu_{\bar{A}}^2(x_i)\right)}$ – квадратичный индекс.

10) Отметим следующие свойства, связанные с ближайшим обычным множеством:

$\underline{A \cap B} = \underline{A} \cap \underline{B}$;

$\underline{A \cup B} = \underline{A} \cup \underline{B}$,

а также $\forall x \in U : \left|\mu_A(x_i) - \mu_{\underline{A}}(x_i)\right| = \mu_{A \cap \overline{A}}(x_i)$, откуда для линейного индекса нечеткости имеем

$$d(A) = \frac{2}{n}\sum_{i=1}^{n} \mu_{A \cap \overline{A}}(x_i),$$

т. е. в этом представлении становится очевидным, что $d(A) = d(\overline{A})$.

Нечеткое множество с функцией принадлежности $2\mu_{A \cap \overline{A}}(x_i)$ иногда называют векторным индикатором нечеткости.

Принцип обобщения

Принцип обобщения – одна из основных идей теории нечетких множеств – носит эвристический характер и используется для расширения области применения нечетких множеств на отображения.

Будем говорить, что имеется нечеткая функция *f*, определенная на *X* со значением *Y*, если она каждому элементу $x \in X$ ставит в соответствие элемент $y \in Y$ со степенью принадлежности $\mu_f(x, y)$. Нечеткая функция f определяет нечеткое отображение $f : X \xrightarrow{H} Y$.

Принцип обобщения заключается в том, что при заданном четком $f : X \to Y$ или нечетком $f : X \xrightarrow{H} Y$ отображении для любого нечеткого множества *A*, заданного на *X*, определяется нечеткое множество *f*(*A*) на *Y*, являющееся образом *A*.

Пусть $f : X \to Y$ – заданное четкое отображение, а $A = \{\mu_A(x)/x\}$ – нечеткое множество в *X*. Тогда образом *A* при отображении *f* является нечеткое множество *f*(*A*) на *Y* с функцией принадлежности

$$\mu_{f(A)}(y) = \sup_{x \in f^{-1}(y)} \mu_A(x); \quad y \in Y,$$

где $f^{-1}(y) = \{x / f(x) = y\}$.

В случае нечеткого отображения $f : X \to Y$, когда для любых $x \in X$ и $y \in Y$ определена двуместная функция принадлежности $\mu_f(x, y)$ $\mu_f(x,y)$, образом нечеткого множества *A*, заданного на *X*, является нечеткое множество *f*(*A*) на *Y* с функцией принадлежности $\mu_{f(A)}(y) = \sup_{x \in U} \min\left(\mu_A(x), \mu_f(x, y)\right)$.

Нечеткие отношения

Нечёткое множество — понятие, введённое Лотфи Заде, который допустил, что функция принадлежности элемента множеству может принимать любые значения в интервале $[0, 1]$, а не только значения 0 или 1. Является базовым понятием нечёткой логики.

Под *нечётким множеством* A понимается совокупность упорядоченных пар, составленных из элементов x универсального множества X и соответствующих степеней принадлежности $\mu_A(x)$

$$A = \{(x, \mu_A(x)) | x \in X\},$$

причем $\mu_A(x)$ – функция принадлежности (**характеристическая функция**), указывающая, в какой степени (мере) элемент x принадлежит нечёткому множеству A.

Функция $\mu_A(x)$ принимает значения в некотором линейно упорядоченном множестве M на отрезке $[0,1]$. Если $M = \{0,1\}$, т.е. состоит только из двух элементов, нечёткое множество может рассматриваться как обычное, чёткое множество.

Пусть $U = U_1 \times U_2 \times \ldots \times U_n$ – прямое произведение универсальных множеств и M – некоторое множество принадлежностей (например, $M = [0,1]$). В случае $n = 2$ и $M = [0,1]$ нечетким отношением R между множествами $X=U_1$ и $Y=U_2$ будет называться функция $R:(X,Y) \rightarrow [0,1]$, которая ставит в соответствие каждой паре элементов $(x,y) \in X \times Y$ величину $\mu_R(x,y) \in [0,1]$.

Обозначение: нечеткое отношение на $X \times Y$ запишется в виде $x \in X, y \in Y : xRy$. В случае, когда $X = Y$, т.е. X и Y совпадают, нечеткое отношение $R:X \times X \rightarrow [0,1]$ называется нечетким отношением на множестве X.

Примеры:

а) Пусть $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3, y_4\}$, $M = [0,1]$. Нечеткое отношение $R = XRY$ может быть задано, к примеру, таблицей:

R	y_1	y_2	y_3	y_4
x_1	0	0	0,1	0,3
x_2	0	0,8	1	0,7
x_3	1	0,5	0,6	1

б) Пусть $X = Y$ – множество всех действительных чисел. Отношение $x >> y$ (x много больше y) можно задать функцией принадлежности

$$\mu_R(x,y) = \begin{cases} 0, & \text{если } x \le y \\ \dfrac{1}{1 + \dfrac{1}{(x-y)^2}}, & \text{если } y < x. \end{cases}$$

в) Отношение R, для которого $\mu_R(x,y)=e^{-k(x-y)^2}$, при достаточно больших k можно интерпретировать так: «x и y – близкие друг к другу числа».

3.6. Элементы теории алгоритмов. Графовое представление математических моделей конструктивных модулей

История возникновения теории графов

Родоначальником теории графов принято считать математика Леонарда Эйлера (1707–1783). Однако теория графов многократно переоткрывалась разными учеными при решении различных прикладных задач.

1. Задача о Кенигсбергских мостах.

На рис. 3.12 представлен схематический план центральной части города Кенигсберг (ныне Калининград), включающий два берега реки Перголя, два острова на ней и семь соединяющих мостов. Задача состоит в том, чтобы обойти все четыре части суши, пройдя по каждому мосту один раз, и вернуться в исходную точку. Эта задача была решена (доказано, что решение не существует) Л. Эйлером в 1736 году.

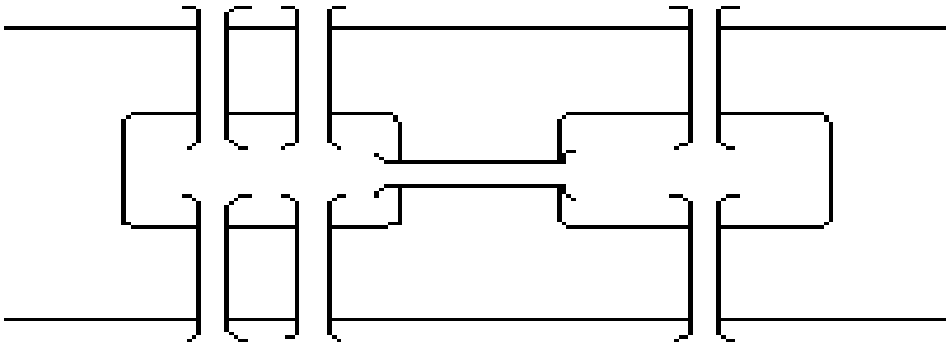


Рис. 3.12. К задаче о семи мостах

2. Задача о трех домах и трех колодцах.

Имеется три дома и три колодца, каким-то образом расположенные на плоскости. Провести от каждого дома к каждому колодцу тропинку так, чтобы тропинки не пересекались (рис. 3.13). Эта задача была решена (доказано, что решение не существует) К.М. Куратовским в 1930 году.

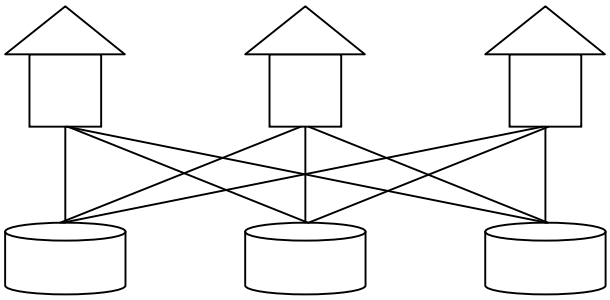


Рис. 3.13. К задаче о трех колодцах

3. Задача о четырех красках.

Разбиение плоскости на непересекающиеся области называется картой. Области на карте называются соседними, если они имеют общую границу. Задача состоит в раскрашивании карты таким образом, чтобы никакие две соседние области не были закрашены одним цветом (рис. 3.14). С конца позапрошлого века известна гипотеза, что для этого достаточно четырех красок.

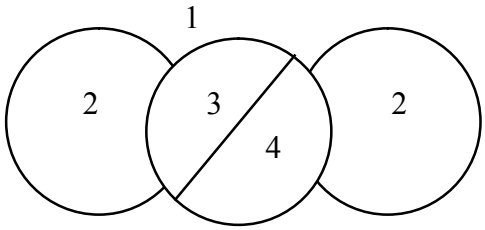


Рис. 3.14. К задаче о четырех красках

В 1976 году К. Аппель и Ф. Хейкен опубликовали решение задачи о четырех красках, которое базировалось на переборе вариантов с помощью компьютера. Решение этой задачи «программным путем» явилось прецедентом, породившим бурную дискуссию, которая отнюдь не закончена. Суть опубликованного решения состоит в том, чтобы перебрать большое, но конечное число (около 2000) типов потенциальных контрпримеров к теореме о четырех красках и показать, что ни один случай контрпримером не является. Этот перебор был выполнен программой примерно за тысячу часов работы суперкомпьютера. Проверить «вручную» полученное решение невозможно – объем перебора выходит далеко за рамки человеческих возможностей.

Многие математики ставят вопрос: можно ли считать такое «программное доказательство» действительным доказательством? Ведь в программе могут быть ошибки… Методы формального доказательства правильности программ не применимы к программам такой сложности, как обсуждаемая. Тестирование не может гарантировать отсутствие ошибок и в данном случае вообще невозможно. Таким образом, остается уповать на программистскую квалификацию авторов и верить, что они сделали все правильно.

Основные понятия теории графов

1) *Графом G(V,E)* называется совокупность двух множеств – непустого множества V (множества вершин) и множества E двухэлементных подмножеств множества V (E – множество ребер).

2) *Ориентированным* называется граф, в котором $E \subseteq V \times \mathrm{V}$ – множество упорядоченных пар вершин вида (x, y), где x – начало, а y – конец дуги. Дугу (x, y) часто записывают как $x \rightarrow y$. Говорят также, что дуга $x \rightarrow y$ ведет от вершины x к вершине y, а вершина y – *смежная* с вершиной x.

3) Если элементом множества E может быть пара *одинаковых* (не различных) элементов V, то такой элемент множества E называется *петлей*, а граф – *графом с петлями* (или *псевдографом*).

4) Если E является не множеством, а *набором*, содержащим несколько одинаковых элементов, то эти элементы называются *кратными ребрами*, а граф – *мультиграфом*.

5) Если элементами множества E являются не обязательно двухэлементные, а любые подмножества множества V, то такие элементы множества E называются *гипердугами*, а граф – *гиперграфом*.

6) Если задана функция $F:V \rightarrow M$ и/или $F:E \rightarrow M$, то множество M называется множеством *пометок*, а граф – *помеченным* (или *нагруженным*). В качестве множества пометок обычно используются буквы или целые числа. Если функция F инъективна, т. е. разные вершины (ребра) имеют разные пометки, то граф называют *нумерованным*.

7) *Подграфом* называется граф $G'(V',E')$, где $V' \subset V$ и/или $E' \subset E$.

а) Если $V' = V$, то G' называется *остовным* подграфом G.

б) Если $V' \subset V \,\&\, E' \subset E \,\&\, (V' \neq \mathrm{V} \vee \mathrm{E}' \neq \mathrm{E})$, то граф G' называется *собственным* подграфом графа G.

в) Подграф $G'(V',E')$ называется *правильным* подграфом графа $G(V,E)$, если G' содержит все возможные рёбра G.

8) *Степень (валентность)* вершины – это количество ребер, инцидентных этой вершине (количество смежных с ней вершин).

9) *Маршрутом* в графе называется чередующаяся последовательность вершин и ребер $v_0, e_1, v_1, e_2, v_2, ..., e_k, v_k$, в которой любые два соседних элемента инцидентны.

а) Если $v_0 = v_k$, то маршрут *замкнут*, иначе – *открыт*.

б) Если все ребра различны, то маршрут называется *цепью.*

в) Если все вершины (а значит, и ребра) различны, то маршрут называется *простой цепью*.

г) Замкнутая цепь называется *циклом*.

д) Замкнутая простая цепь называется *простым циклом*.

е) Граф без циклов называется *ациклическим.*

ж) Для орграфов цепь называется *путем*, а цикл – *контуром*.

Пример

В графе, диаграмма которого приведена на рис. 3.15:

- v_1, v_3, v_1, v_4 – маршрут, но не цепь;
- $v_1, v_3, v_5, v_2, v_3, v_4$ – цепь, но не простая цепь;
- v_1, v_4, v_3, v_2, v_5 – простая цепь;
- $v_1, v_3, v_5, v_2, v_3, v_4, v_1$ – цикл, но не простой цикл;
- v_1, v_3, v_4, v_1 – простой цикл.

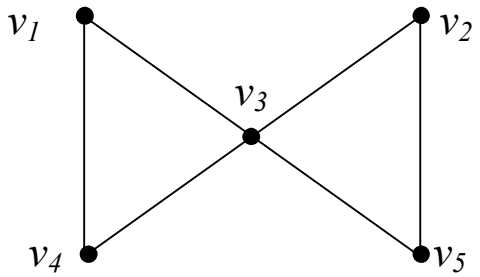


Рис. 3.15. Маршруты, цепи, циклы

10) Если граф имеет цикл (не обязательно простой), содержащий все ребра графа по одному разу, то такой цикл называется *эйлеровым циклом*.

11) Если граф имеет простой цикл, содержащий все вершины графа (по одному разу), то такой цикл называется *гамильтоновым циклом*.

12) *Деревом* называется связный граф без циклов.

13) *Остовом* называется дерево, содержащее все вершины графа.

14) *Паросочетанием* называется множество ребер, в котором никакие два не смежны.

15) Паросочетание называется *максимальным*, если никакое его надмножество не является независимым.

16) Две вершины в графе *связаны*, если существует соединяющая их простая цепь.

17) Граф, в котором все вершины связаны, называется *связным.*

18) Граф, состоящий только из изолированных вершин, называется *вполне несвязным.*

19) *Длиной маршрута* называется количество ребер в нем (с повторениями).

20) *Расстоянием* между вершинами u и v называется длина кратчайшей цепи $\langle u,v\rangle$, а сама кратчайшая цепь называется *геодезической*.

21) *Диаметром* графа G называется длина длиннейшей геодезической цепи.

22) *Эксцентриситетом* вершины v в связном графе $G(V,E)$ называется максимальное расстояние от вершины v до других вершин графа G (рис. 3.16).

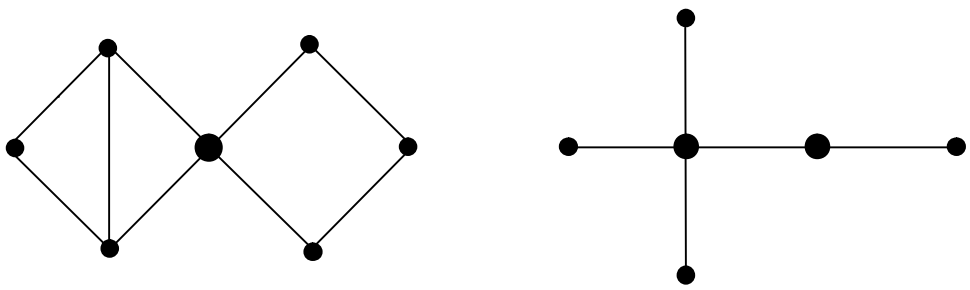


Рис. 3.16. Эксцентриситеты вершин и центры графов (выделены)

23) *Радиусом* графа G называется наименьший из эксцентриситетов вершин.

24) Вершина v называется *центральной*, если ее эксцентриситет совпадает с радиусом графа.

25) Множество центральных вершин называется *центром* графа (см. рис. 3.16).

Основные теоремы теории графов

Опираясь на представленные выше определения теории графов, приведем формулировки и доказательства теорем, которые затем найдут применение при решении задач.

Теорема 1. Удвоенная сумма степеней вершин любого графа равна числу его ребер.

Доказательство. Пусть A_1, A_2, A_3, ..., A_n — вершины данного графа, а $p(A_1)$, $p(A_2)$, ..., $p(A_n)$ – степени этих вершин. Подсчитаем число ребер, сходящихся в каждой вершине, и просуммируем эти числа. Это равносильно нахождению суммы степеней всех вершин. При таком подсчете каждое ребро будет учтено дважды (оно ведь всегда соединяет две вершины).

Отсюда следует: $p(A_1) + p(A_2) + ... + (A_n) = 0{,}5N$, или $2(p(A_1) + p(A_2) + ... + p(A_n)) = N$, где N — число ребер.

Теорема 2. Число нечетных вершин любого графа четно.

Доказательство. Пусть $a_1, a_2, a_3, ..., a_k$ — это степени четных вершин графа, а $b_1, b_2, b_3, ..., b_m$ – степени нечетных вершин графа. Сумма $a_1+a_2+a_3+...+a_k+b_1+b_2+b_3+...+b_m$ ровно в два раза превышает число ребер графа. Сумма $a_1+a_2+a_3+...+a_k$ – четная (как сумма четных чисел), тогда сумма $b_1+b_2+b_3+...+b_m$ должна быть четной. Это возможно лишь в том случае, если m — четное, т. е. четным является и число нечетных вершин графа, что и требовалось доказать.

Эта теорема имеет немало любопытных следствий.

Следствие 1. Нечетное число знакомых в любой компании всегда четно.

Следствие 2. Число вершин многогранника, в которых сходится нечетное число ребер, четно.

Следствие 3. Число всех людей, когда-либо пожавших руку другим людям нечетное число раз, является четным.

Теорема 3. Во всяком графе с n вершинами, где $n \geq 2$, всегда найдутся две или более вершины с одинаковыми степенями.

Доказательство. Если граф имеет n вершин, то каждая из них может иметь степень 0, 1, 2, ..., $(n–1)$. Предположим, что в некотором графе все его вершины имеют различную степень, и докажем, что этого быть не может. Действительно, если $p(A) = 0$, то это значит, что

A — изолированная вершина, и поэтому в графе не найдется вершины X со степенью $p(X) = n–1$. В самом деле, эта вершина должна быть соединена с $(n–1)$ вершиной, в том числе и с A, но ведь A оказалась изолированной. Следовательно, в графе, имеющем n вершин, не могут быть одновременно вершины степени 0 и $(n–1)$. Это значит, что из n вершин найдутся две, имеющие одинаковые степени.

Теорема 4. Если в графе с n вершинами ($n \geq 2$) только одна пара имеет одинаковую степень, то в этом графе всегда найдется либо единственная изолированная вершина, либо единственная вершина, соединенная со всеми другими.

Доказательство данной теоремы опускаем. Остановимся лишь на некотором ее пояснении. Содержание этой теоремы хорошо разъясняется задачей: группа, состоящая из n школьников, обменивается фотографиями. В некоторый момент времени выясняется, что двое совершили одинаковое число обменов. Доказать, что среди школьников есть либо один, еще не начинавший обмена, либо один, уже завершивший его.

Теорема 5. Если у графа все простые циклы четной длины, то он не содержит ни одного цикла четной длины.

Суть теоремы в том, что на этом графе невозможно найти цикл (как простой, так и непростой) нечетной длины, т. е. содержащий нечетное число ребер.

Теорема 6. Для того чтобы граф был эйлеровым, необходимо и достаточно, чтобы он был связным и все его вершины имели четную степень**.**

Теорема 7. Для того чтобы на связном графе можно было бы проложить цепь AB, содержащую все его ребра в точности по одному разу, необходимо и достаточно, чтобы A и B были единственными нечетными вершинами этого графа.

Доказательство этой теоремы очень интересно и характерно для теории графов. Его также следует считать конструктивным. Для доказательства к исходному графу присоединяем ребро (A, B); после этого все вершины графа станут четными. Этот новый граф удовлетворяет всем условиям теоремы, и поэтому в нем можно проложить эйлеров цикл Ψ. И если теперь в этом цикле удалить ребро (A, B), то останется искомая цепь AB**.**

На этом приеме основано доказательство следующей теоремы, которую следует считать обобщением теоремы 7.

Теорема 8. Если данный граф является связным и имеет $2k$ вершин нечетной степени, то в нем можно провести k различных цепей, содержащих все его ребра в совокупности ровно по одному разу.

Теорема 9. Различных деревьев с n перенумерованными вершинами можно построить n^{n-2}.

По поводу доказательства этой теоремы сделаем одно замечание. Эта теорема известна, в основном, как вывод английского математика А. Кэли (1821–1895). Графы-деревья издавна привлекали внимание ученых. Сегодня двоичные деревья используются не только математиками, но и биологами, химиками, физиками и инженерами.

Теорема 10. Полный граф с пятью вершинами не является плоским.

Доказательство. Воспользуемся формулой Эйлера: $B - P + \Gamma = 2$, где B – число вершин плоского графа, P – число его ребер, Γ – число граней. Формула Эйлера справедлива для плоских связных графов, в которых ни один из многоугольников не лежит внутри другого.

Эту формулу можно доказать методом математической индукции. Доказательство опускаем, заметим только, что формула справедлива и для пространственных многогранников. Пусть все пять вершин графа соединены друг с другом. Замечаем, что на графе нет ни одной грани, ограниченной только двумя ребрами. Если через φ_1 обозначить число таких граней, то $\varphi_2 = 0$. Далее рассуждаем от противного, а именно: предположим, что исследуемый граф – плоский. Это значит, что для него верна формула Эйлера. Число вершин в данном графе $B = 5$, число ребер $P = 10$, тогда число граней $\Gamma = 2 - B + P = 2 - 5 + 10 = 7$.

Это число можно представить в виде суммы: $\Gamma = \varphi_1 + \varphi_2 + \varphi_3 + \ldots$, где φ_3 – число граней, ограниченных тремя ребрами, φ_4 — число граней, ограниченных четырьмя ребрами и т. д.

С другой стороны, каждое ребро является границей двух граней, поэтому число граней равно $2P$, в то же время $2P = 20 = 3\varphi_3 + 4\varphi_4 + \ldots$. Умножив равенство $\Gamma = 7 = \varphi_3 + \varphi_4 + \varphi_5 + \ldots$ на 3, получим $3\Gamma = 21 = 3(\varphi_3 + \varphi_4 + \varphi_5 + \ldots)$.

Ясно, что $(3\varphi_3 + 3\varphi_4 + 3\varphi_5 + \ldots) < (3\varphi_3 + 4\varphi_4 + 5\varphi_5 + \ldots)$ или $3\Gamma < 2P$, но по условию $2P = 20$, а $3\Gamma = 21$. Поэтому вывод, полученный при введенном нами предположении (граф – плоский), противоречит условию. Отсюда заключаем, что полный граф с пятью вершинами не является плоским.

Теорема 11. (Теорема Понтрягина – Куратовского). Граф является плоским тогда и только тогда, когда он не имеет в качестве подграфа полного графа с пятью вершинами.

Способы представления графов в компьютере

Конструирование структур данных для представления в программе объектов математической модели – это основа искусства практического программирования. Далее приводится четыре различных базовых представления графов. Выбор наилучшего представления определяется требованиями конкретной задачи. Более того, при решении конкретных задач используются, как правило, некоторые комбинации или модификации указанных представлений, общее число которых необозримо. Но все они так или иначе основаны на базовых идеях, описанных ниже.

Требования к представлению графов

Известны различные способы представления графов в памяти компьютера, которые различаются объемом занимаемой памяти и скоростью выполнения операций над графами. Представление выбирается исходя из потребностей конкретной задачи. Далее приведены четыре наиболее часто используемых представления с указанием характеристики $n(p,q)$ – объема памяти для каждого представления. Здесь p – число вершин, а q – число ребер.

Матрица смежности. Представление графа с помощью квадратной булевой матрицы M, отражающей смежность вершин, называется матрицей смежности, где

$$M[i, j] = \begin{cases} 1, \text{ если вершина } v_i \text{ смежна с вершиной } v_j, \\ 0, \text{ если вершины } v_i \text{ и } v_j \text{ не смежны.} \end{cases}$$

Для матрицы смежности $n(p,q) = O(p^2)$.

Замечание. Матрица смежности неориентированного графа симметрична относительно главной диагонали, поэтому достаточно хранить только верхнюю (или нижнюю) треугольную матрицу.

Матрица инциденций. Представление графа с помощью матрицы H, отражающей инцидентность вершин и ребер, называется матрицей инциденций, где для неориентированного графа

$$H[i, j] = \begin{cases} 1, \text{ если вершина } v_i \text{ инцидентна ребру } e_j, \\ 0 – \text{ в противном случае,} \end{cases}$$

а для орграфа

$$H[i,j] = \begin{cases} 1, \text{ если вершина } v_i \text{ инцидентна ребру } e_j \text{ (его конец)}, \\ 0, \text{ если вершина } v_i \text{ и ребро } e_j \text{ не инцидентны}, \\ -1, \text{ если вершина } v_i \text{ инцидентна ребру } e_j \text{ (его начало)}. \end{cases}$$

Для матрицы инциденций $n(p,q) = O(pq)$.

Списки смежности. Представление графа с помощью списочной структуры, отражающей смежность вершин и состоящей из массива указателей на списки смежных вершин, где элемент списка представлен структурой

N : **record** v : 1..p; n :↑ N **end record**,

называется *списком смежности*. В случае представления неориентированных графов – списками смежности $n(p,q) = O(p + 2q)$, а в случае ориентированных графов – $n(p,q) = O(p + q)$.

Массив дуг. Представление графа с помощью массива структур

E : **array** [1..q] **of record** b,e : 1..p **end record**,

отражающего список пар смежных вершин, называется *массивом ребер* (для орграфов – *массивом дуг*). Для массива ребер (или дуг) $n(p,q) = O(2q)$.

Обзор задач теории графов

Развитие теории графов в основном обязано большому числу всевозможных приложений. По-видимому, из всех математических объектов графы занимают одно из первых мест в качестве формальных моделей реальных систем.

Графы нашли применение практически во всех отраслях научных знаний: физике, биологии, химии, математике, истории, лингвистике, социальных науках, технике и т.п. Наибольшей популярностью теоретико-графовые модели пользуются при исследовании коммуникационных сетей, систем информатики, химических и генетических структур, электрических цепей и других систем сетевой структуры.

Далее перечислены некоторые типовые задачи теории графов и их приложения:

- *Задача о кратчайшей цепи*
- замена оборудования;
- составление расписания движения транспортных средств;
- размещение пунктов скорой помощи;
- размещение телефонных станций.

- *Задача о максимальном потоке*

• анализ пропускной способности коммуникационной сети;

• организация движения в динамической сети;

• оптимальный подбор интенсивностей выполнения работ;

• синтез двухполюсной сети с заданной структурной надежностью;

• задача о распределении работ.

- *Задача об упаковках и покрытиях*

• оптимизация структуры ПЗУ;

• размещение диспетчерских пунктов городской транспортной сети.

- *Раскраска в графах*

• распределение памяти в ЭВМ;

• проектирование сетей телевизионного вещания.

- *Связность графов и сетей*

• проектирование кратчайшей коммуникационной сети;

• синтез структурно-надежной сети циркуляционной связи;

• анализ надежности стохастических сетей связи.

- *Изоморфизм графов и сетей*

• структурный синтез линейных избирательных цепей;

• автоматизация контроля при проектировании БИС.

- *Изоморфное вхождение и пересечение графов*

• локализация неисправности с помощью алгоритмов поиска МИПГ;

• покрытие схемы заданным набором типовых подсхем.

- *Автоморфизм графов*

• конструктивное перечисление структурных изомеров для производных органических соединений;

• синтез тестов цифровых устройств.

Математические модели схем

Основные исходные данные для создания модели — схема электрическая функциональная или принципиальная и параметры типовых конструкций. Постановка задачи и качество решения зависят от математической модели схемы и монтажного пространства. К математической модели схем предъявляют следующие требования [34, 39]:

• информационная полнота (наиболее полное отображение свойств);

• высокая формализация;

• наличие математического аппарата, позволяющего формализовать модель;

• однозначность и простота перехода от объекта к модели и обратно;

• возможность использования модели в существующих алгоритмах или получение модели, где эти алгоритмы работают;

• наглядность представления объекта;

• адекватность модели объекту.

В наибольшей степени изложенным требованиям удовлетворяет граф, являющийся содержательной моделью объекта проектирования. Геометрическое задание графа наглядно представляет отображаемый объект, а матричный и аналитический способы — формально.

Для основных задач конструирования ВС (компоновка, размещение и трассировка) в математической модели отражается следующая информация о схеме [43, 48]:

• связность элементов схемы с точностью до вывода с учетом направления распространения сигнала и фактора неизвестности соединений в пределах одного комплекса (электрической цепи);

• топологические свойства элементов (порядок расположения выводов, возможность перехода соединений между ними и под элементом);

• метрические параметры элементов (их размеры, координаты и размеры полей контактов);

• сведения об инвариантности выводов.

Для различных задач и алгоритмов требуется и различная информация. Например, при разрезании схемы на части и размещении элементов одного типоразмера существенна информация о связности элементов, т. е. электрической связи между ними без учета различия между выходами и входами; при решении задач поиска повторяющихся частей схем и их идентификации необходимо задавать направление связей между элементами с точностью до контактов этих элементов; при решении задач трассировки для определения планарности схем и числа пересечений основными являются топологические свойства элементов [8, 13].

Рассмотрим два способа перехода от схемы к графу [5, 21].

1. Элементам схемы или их выводам ставятся во взаимно однозначное соответствие вершины графа, а связи между ними представ-

ляются ребрами — получаем модель в виде неориентированного или ориентированного обыкновенного графа (мультиграфа).

2. Каждому выводу или элементу схемы ставится во взаимно однозначное соответствие вершина гиперграфа (мультиграфа, если необходимо учитывать направление распространения сигнала), тогда каждое ребро гиперграфа соответствует элементарной цепи, соединяющей эти элементы или их выводы.

Модель схемы в виде неориентированного мультиграфа. Чтобы задать информацию о связности элементов или их выводов, каждая элементарная цепь (комплекс) интерпретируется полным подграфом, что приводит к избыточности ребер, количество вершин подграфа определяется числом элементов или выводов, соединяемых данной цепью. При этом учитывается фактор неизвестности соединения, так как покрывающие деревья, построенные на полном подграфе, соответствуют возможным вариантам соединения элементов данной цепью. Модель схемы получается объединением полных подграфов (рис. 3.18).

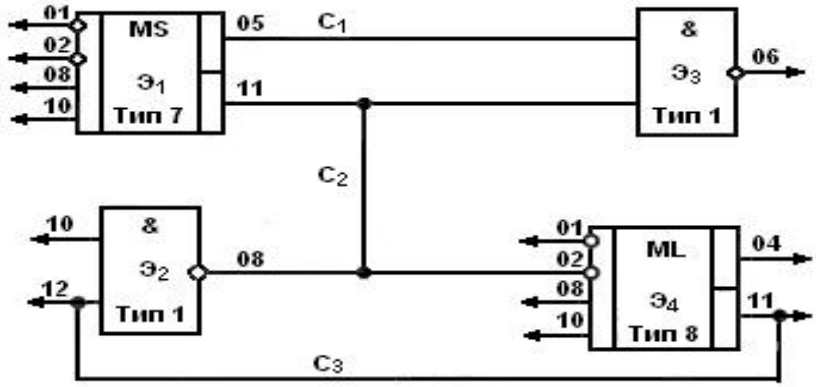


Рис. 3.18. Модель схемы:
& – логическое И; MS – селектор-мультиплексор (выбор канала со стробированием и непрерывная передача информационных посылок в одну линию);
ML – мультиплексор (демультиплексор или аналоговый коммутатор)

При такой интерпретации применяется вероятностный подход – каждому ребру $v_i \in V$ полного подграфа присваивают вес:

$$p_i = 1/(k–1),$$

где k – количество вершин полного подграфа.

При сопоставлении элементов схемы и вершин графа получаем граф (рис. 3.19).

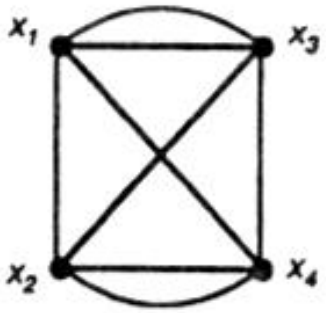


Рис. 3.19. Граф

Введение избыточных ребер может сделать граф непланарным, хотя интерпретируемая схема планарна. По данному графу нельзя получить правильную оценку элементарных связей между частями схемы. Например, количество ребер, попадающих в разрез между G_1 и G_2 графа G ($X_1 = \{x_1, x_3\}$, $X_2 = \{x_2, x_4\}$), равно четырем (для вероятностного графа сумма весов ребер равна 4/3), в то время как в схеме в этом случае разрезается одна цепь. При такой модели схемы существует сильная корреляционная связь между показателями, так что оптимизация одного приводит к оптимизации другого.

При сопоставлении выводов элементов и вершин граф схемы распадается на отдельные компоненты связности (рис. 3.20), количество которых определяется числом электрических цепей схемы. Объединяя эти компоненты связности в соответствии с принадлежностью выводов элементам схемы, получим рассмотренную выше модель.

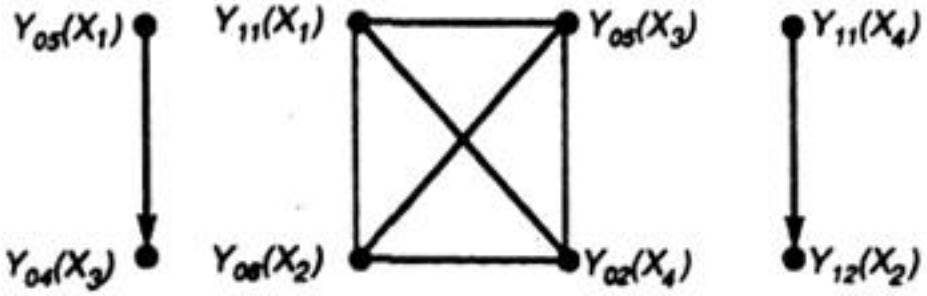


Рис. 3.20. Компоненты связности

Модель схемы, полученную объединением полных подграфов, можно не использовать для решения задач размещения элементов (информацию о метрических параметрах элементов можно учитывать

в весовых характеристиках вершин) и компоновки алгоритмами, в которых определяющим является фактор связности. Модель схемы в виде отдельных компонентов связности несет информацию о соединяемых выводах элементов для задачи трассировки [33].

Электрическую цепь можно представить фиксированным деревом (рис. 3.21). В этом случае не исключаются избыточные ребра, однако не учитывается фактор неизвестности соединений, и неверно отражается связность элементов схемы, так как любые две несмежные вершины дерева не связаны между собой, в то время как в схеме между соответствующими элементами существует электрическая связь. Такую модель можно использовать для решения топологических задач трассировки, если нет ограничений на проведение соединений под элементами и между их контактами.

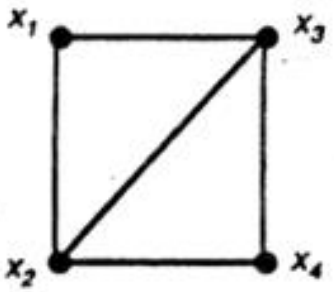


Рис. 3.21. Фиксированное дерево

Модель схемы в виде неориентированного мультиграфа. Такое представление схемы необходимо для задач, где учитывается направление связей между элементами. В этих задачах точная оценка числа связей между элементами или частями схемы несущественна. Чтобы определить, что сигнал с выхода одного элемента поступает на вход другого, используют следующий способ представления электрических цепей дугами ориентированного графа: каждая цепь, соединяющая выходы *n* источников сигнала со входами *m* приемников, т. е. каждая вершина, поставленная в соответствие элементу — источнику сигнала для данной цепи, соединена дугой с каждой вершиной, соответствующей элементу — приемнику сигнала.

При таком способе представления цепей также появляются избыточные ребра. Модель схемы получается объединением двудольных ориентированных графов. Логическую функцию элемента схемы можно задать в качестве весовой характеристики, соответствующей вершине графа. Граф схемы представлен на рис. 3.22.

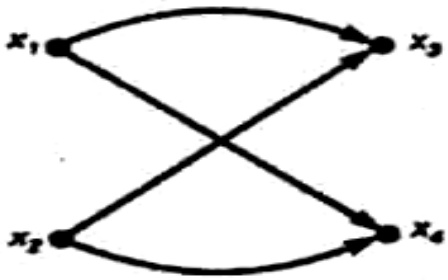


Рис. 3.22. Граф схемы

В этом графе весовая характеристика, например, вершины x_1 равна семи, т. е. определяется типом элемента Э1. Модель не отображает схему с точностью до вывода элемента, поэтому является корректной для схем, реализованных на элементах с одним выходом и равнозначными входами. Корректность модели для схем, построенных на элементах с неравнозначными входами и выходами, может быть обеспечена введением весов ребер.

Вес каждого ребра представляет собой упорядоченную пару, первый элемент которой характеризует выход элемента-источника, а второй – вход элемента-приемника (в простейшем случае пару составляют номера выводов этих элементов). Данная модель предназначена для решения частных задач компоновки (поиск повторяющихся частей схем, установление идентичности схем), сопоставления выводов с вершинами графа.

Идентификацию с точностью до выводов элементов схем можно получить, при этом граф схемы распадается на n компонент связности, где n – число электрических цепей схемы.

Представление схемы гиперграфом и ультраграфом. Рассмотрим модель в виде гиперграфа, когда множество элементов схемы Э соответствует множеству X, а множество электрических цепей C — множеству ребер U ($|X| = n$ — число элементов в схеме; $|U| = m$ — число электрических цепей схемы). Каждое ребро гиперграфа U_k представляется подмножеством тех вершин $X_k \in X$, которым соответствуют элементы, соединенные k-й электрической цепью.

При задании схемы гиперграфом учитывается фактор неизвестности соединения, т. е. для определения связи между i-м и j-м элементами схемы k-электрической цепью достаточно проверить условие $x_i, x_j \in X_k$. Количество связей между некоторым подмножест-

вом X' вершин гиперграфа и его дополнением $X \mid X'$ подсчитывают как число ребер $U_k \equiv X_k$, для которых выполняется условие

$$\exists x_i, x_j \in X_k (x_i \in X' \,\&\, x_j \in X \mid X' \vee x_i \in X \mid X' \,\&\, x_j \in X'); \quad (3.1)$$

$$i, j \in J = \overline{1, n}; \quad k \in K = \overline{1, m}.$$

Отсюда видно, что по гиперграфу можно точно оценить число электрических соединений между частями или элементами схемы. Например, схема, представленная на рис. 3.18, интерпретируется гиперграфом (рис. 3.23).

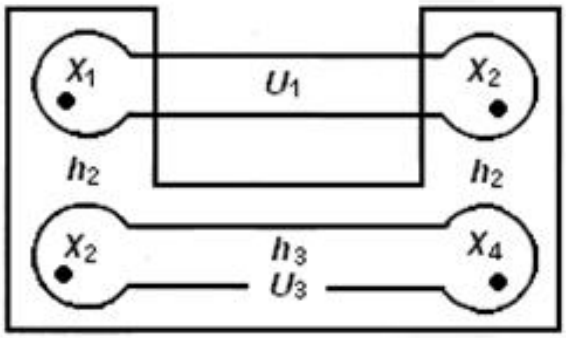


Рис. 3.23. Гиперграф схемы

Число электрических цепей, соединяющих элементы 1 и 3 с остальными, будет равно единице. Подсчет числа ребер гиперграфа, для которых выполняется условие (3.1) при $X' = \{X_1, X_3\}$, дает такое же значение. При матричном представлении модели схемы в виде гиперграфа принадлежность *i*-го элемента схемы *j*-й электрической цепи с точностью до вывода элемента можно задать.

Идентификацию элементов с точностью до вывода при аналитическом представлении гиперграфа можно обеспечить присваиванием весов, характеризующих эти выводы, вершинам, входящим в ребра. Из гиперграфа с помощью соответствующих преобразований можно получить модель схемы в виде неориентированного мультиграфа [20, 37].

При представлении схемы ультраграфом множеству элементов схемы ставится во взаимно однозначное соответствие множество вершин *X,* а множеству электрических цепей – множество ребер *U.* Направление передачи сигналов в модели схемы задается следующим образом: пусть *i*-й элемент схемы принадлежит *j*-й цепи, тогда бинар-

ное отношение инцидентности задано на паре (X_i, U_j), если X_i сопоставлено с элементом как источником сигнала, и (X_i, U_j) – если X_i интерпретирует элемент как приемник сигнала. Кенигово представление ультраграфа схемы приведено на рис. 3.24.

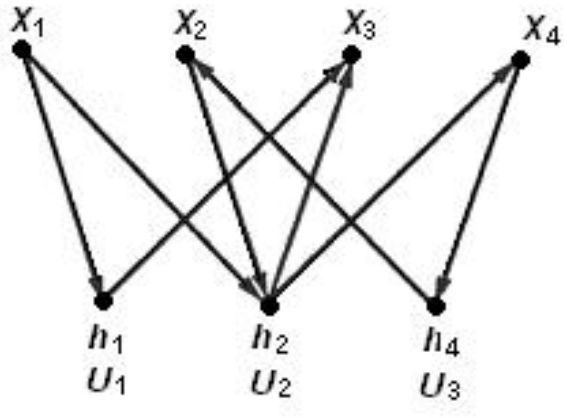


Рис. 3.24. Ультраграф схемы

Отображение схемы с точностью до выводов элементов обеспечивается введением весов, характеризующих эти выводы. При задании ультраграфа в виде множеств X, U и отображения U в X весами вершин, входящих в граф, установлены номера контактов элементов, сопоставленных с этими вершинами.

Ультраграф, как и гиперграф, учитывает фактор неизвестности соединений и позволяет точно оценить число электрических соединений [26, 49]. Для всех рассмотренных моделей не выполняется требование информационной полноты. В наибольшей степени оно удовлетворяется, когда схема представляется ультраграфом, при наличии дополнительных сведений о конструктивно-технологических характеристиках элементов и их логических функциях.

При интерпретации элементов схемы вершинами графа эти сведения для всех моделей могут быть заданы в виде весовых характеристик вершин. Топологические свойства элементов схемы не отображены ни в одной из рассмотренных моделей.

При представлении схемы в виде неориентированного мультиграфа и гиперграфа не удовлетворяется требование однозначности перехода от модели к схеме. Теория неориентированных и ориентированных графов развита достаточно хорошо. Разработано большое количество алгоритмов решения задач схемно-топологического конструирования методами теории графов. Математический аппарат теории гипер-и ультраграфов в настоящее время только развивается.

Математические модели монтажного пространства

Под монтажным пространством типовой конструкции понимают метрическое пространство, в котором устанавливаются входящие в нее типовые конструкции предыдущих уровней и выполняются электрические соединения выводов. Модель монтажного пространства отображает метрические параметры и топологические свойства конструкции.

Метрические параметры – это габаритные размеры зон монтажа, допустимая ширина проводников и зазора между ними, координаты и размеры внешних монтажных площадок, шаг установки и размеры модулей, координаты и размеры полей их контактов.

Топологические свойства – это число слоев монтажной платы и переходов со слоя на слой, наличие замкнутых областей, запрещенных для проведения соединений, ограничения на взаимное расположение соединений в монтажной области и на количество монтажных проводов, подводимых к одному выводу.

В качестве математической модели монтажного пространства используют неориентированный топологический граф (граф решетки). Плоскость монтажного пространства разбивают на элементарные площадки, стороны которых равны шагу проложения проводника по соответствующему направлению (для печатного монтажа элементарная площадка – квадрат). Каждой элементарной площадке ставят в соответствие вершину графа решетки. Две вершины соединены ребром, если между соответствующими элементарными площадками можно провести соединения с учетом метрических и топологических параметров типовых конструкций, устанавливаемых в данном монтажном пространстве.

Модель монтажного пространства может быть представлена фрагментом верхнего слоя печатной платы (рис. 3.25) и фрагментом с ортогональным монтажом при запрещении проведения проводников под микросхемами (рис. 3.26, *а*), а если проводники разрешается проводить под углом 45°, то каждой вершине может быть инцидентно восемь ребер (рис. 3.26, *б*).

Фрагмент математической модели монтажного пространства многослойной печатной платы описывает вертикальные ребра, интерпретирующие межслойные переходы (рис. 3.27).

Из множества вершин этого графа можно выделить следующие подмножества: вершины, сопоставленные с контактными площадками выводов модулей (вершины – внешние выводы типовой конструк-

ции), и вершины, интерпретирующие контактные площадки межслойных переходов (вершины обозначены кружками).

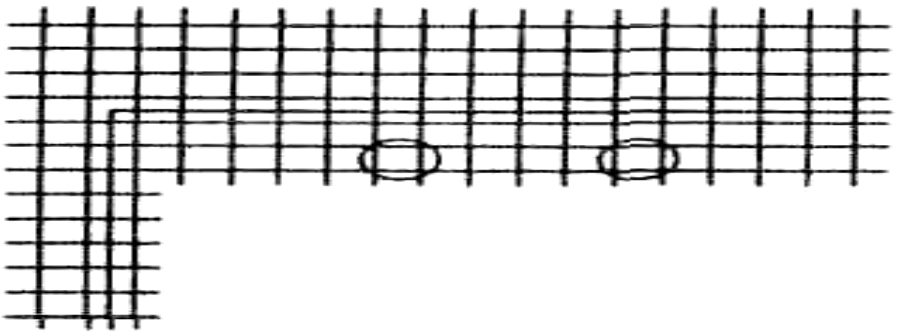
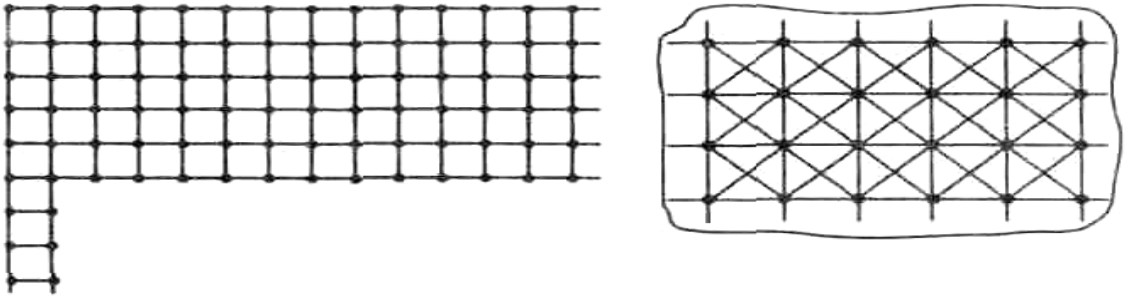


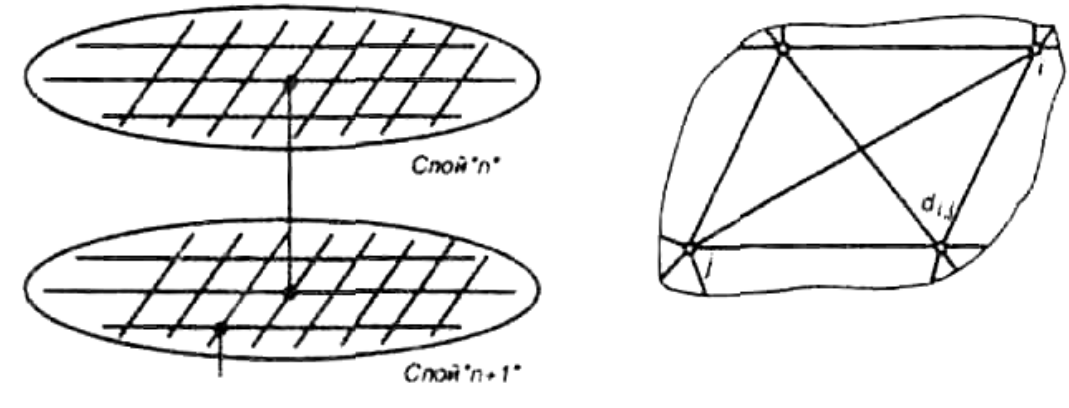
Рис. 3.25. Верхний слой печатной платы



а *б*

Рис. 3.26. Модели монтажной плоскости (*а*)
и многосвязной печатной платы (*б*)

Фрагмент математической модели монтажного пространства многослойной печатной платы описывает вертикальные ребра, интерпретирующие межслойные переходы.



а *б*

Рис. 3.27. Модель многослойной печатной платы (*а*)
и ее представление полным подграфом (*б*)

Из множества вершин этого графа можно выделить следующие подмножества: вершины, сопоставленные с контактными площадками выводов модулей (вершины обозначены кружками); вершины, представляющие внешние выводы типовой конструкции, и вершины, интерпретирующие контактные площадки межслойных переходов (вершины обозначены кружками). В случае выполнения соединений монтажными проводами в любом направлении вершины графа решетки сопоставляют с выводами конструктивного элемента (микросхемы, разъемы соединительной платы и т. п.). Варианты различных соединений представляются полным графом, построенным на этих вершинах (см. рис. 3.27, *б*).

В конкретной реализации соединений необходимо учитывать ограничения на число проводников, подводимых к одному контакту. Расстояние между *i*-м и *j*-м узлами графа решетки в общем случае определяется по формуле

$$d_{i,j} = (|s_i - s_j|^k + |t_i - t_j|^k)^h; \quad i = 1,m; \quad j = 1,m,$$

где m — число узлов графа решетки.

При ортогональной трассировке $k = h = 1$, и получим $d_{i,j} = |s_i - s_j| + |t_i - t_j|$.

Для регулярного монтажного пространства в качестве модели поля размещения используют граф решетки для платы (рис. 3.28).

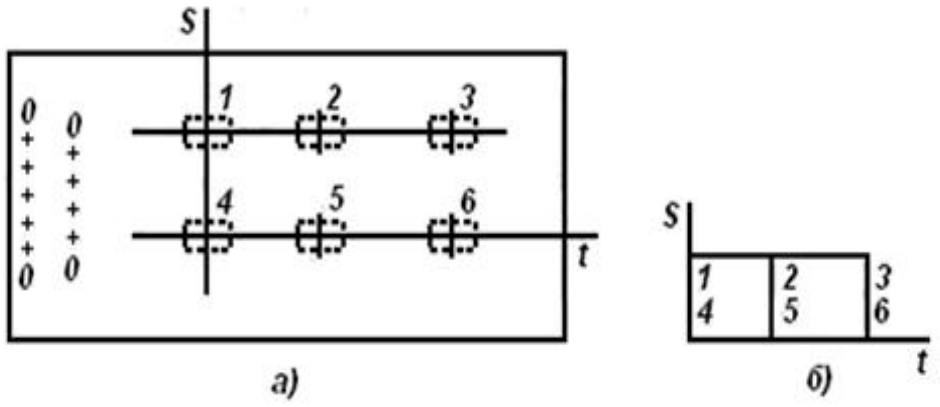


Рис. 3.28. Граф решетки регулярного (*а)* и нерегулярного (*б*) пространства

Приближенный подсчет суммарной длины соединений между модулями можно выполнить следующим образом. Пусть моделью схемы соединения является неориентированный граф G (рис. 3.29), а моделью платы – граф решетки G_λ.

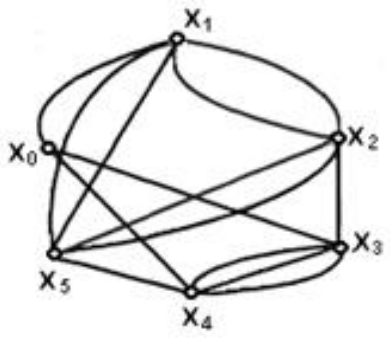


Рис. 3.29. Неориентированный граф *G*

Для графа *G*, преобразованного в решетку G_λ (вершины графа располагаются в узлах решетки G_λ), строится матрица расстояний D_λ, элементы которой подсчитываются по формуле. Если шаги установки модулей по осям *s* и *t* равны, расстояния между соседними узлами решетки принимаются равными единице. Матрица расстояний графа:

$$D_\lambda = \begin{Vmatrix} 0 & 1 & 2 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 & 1 & 2 \\ 2 & 1 & 0 & 3 & 2 & 1 \\ 1 & 2 & 3 & 0 & 1 & 2 \\ 2 & 1 & 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 2 & 1 & 0 \end{Vmatrix}.$$

Суммарная длина ребер графа *G*, преобразованного в решетку G_λ, определяется как полусумма элементов матрицы геометрии D_Y .

Для получения матрицы геометрии D_Y необходимо выполнить поэлементное умножение матрицы D_Y и матрицы смежности *R* графа *G*. Матрицы смежности и геометрии рассматриваемого графа соответственно будут:

$$R = \begin{Vmatrix} 0 & 2 & 0 & 0 & 2 & 1 \\ 2 & 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 3 & 0 & 1 \\ 0 & 0 & 3 & 0 & 1 & 1 \\ 2 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{Vmatrix}; \qquad D_y = \begin{Vmatrix} 0 & 2 & 0 & 0 & 4 & 3 \\ 2 & 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 9 & 0 & 1 \\ 0 & 0 & 9 & 0 & 1 & 2 \\ 4 & 2 & 0 & 1 & 0 & 0 \\ 3 & 0 & 1 & 2 & 0 & 0 \end{Vmatrix}.$$

Для данного случая суммарная длина ребер $L(G) = 25$.

3.7. Алгоритмы конструкторского проектирования

Алгоритмы и модели компоновки

Последовательные алгоритмы структурного синтеза. Алгоритмы такого вида относятся к классу эвристических. Их достоинством является высокая экономичность по затратам машинного времени и требуемому объему оперативной памяти за счет отсутствия процедуры многоразового анализа вариантов структуры. Однако последовательные алгоритмы дают не оптимальные, а близкие к оптимальным решения.

Рассмотрим задачу *компоновки* (монтажной платы), т.е. определения состава типовых конструкций каждого уровня. Задача компоновки решается «снизу вверх», т. е. известные схемы (i-1)-го уровня необходимо распределить по конструкциям i-го уровня. Например, на самом низшем уровне элементами могут быть корпуса микросхем, а конструкциями (блоками) — типовые элементы замены, связанные друг с другом путем разъемных соединений.

В качестве критериев оптимальности при решении задач компоновки используют критерии: минимума суммарного числа N_1 типов модулей, минимума межблочных соединений.

Первый критерий связан с конструктивными характеристиками аппаратуры и показателем технологической стоимости, второй – ведет к повышению надежности конструктивной реализации схемы за счет сокращения числа разъемных соединений, уменьшению помех и задержек сигналов благодаря снижению числа межблочных соединений [44].

Алгоритм компоновки по критерию минимума межблочной связности. Первоначально выбирают исходный элемент схемы. Выбор начального элемента основывается на схемотехнических соображениях:

1. В первый компонуемый узел включены все элементы, смежные с начальным, и сам начальный элемент.

2. Если полученное число элементов равно максимально допустимому числу элементов в первом узле, то компоновка узла заканчивается.

3. Если это число больше или меньше максимально допустимого, то выполняются операции по устранению лишних или добавлению недостающих элементов, причем из нескомпонованных элементов выбирают такой, который имеет наибольшее число связей с элементами, уже вошедшими в состав компонуемого узла.

4. Далее сформированный узел удаляют из схемы и компонуют новые узлы.

5. Процесс повторяется до тех пор, пока схема не будет разбита на требуемое число частей или не будет выяснена невозможность этого.

Сформулируем описанный алгоритм в терминах теории графов.

Первоначально в графе G определяют вершину x, принадлежащую X, с наибольшей локальной степенью $p(x_i)$ (*локальной степенью* $p(x_i)$ вершины x_i, принадлежащей X, называют число ребер, инцидентных этой вершине графа). Если таких вершин несколько, то предпочтение отдается той, которая имеет большее число кратных ребер. Вершина x_i и все смежные с ней вершины включаются в граф G_1. Обозначим это множество вершин через Γx_i. Если $|\Gamma x_i| = n_1$, то G_i образован, если же $|\Gamma x_i| > n_1$, то из графа G_1 удаляют вершины, связанные с остающимися вершинами графа G меньшим числом ребер. Когда $|\Gamma x_i| < n_1$, выбирают вершину x_j, принадлежащую Γx_i, удовлетворяющую условиям

$$\alpha(x_j) = \max\{\alpha(x_k)\} = \max\{p(x_k) - a_k\},$$

где a_k – число ребер, соединяющих вершину x_k со всеми невыбранными вершинами графа G.

Строят множество вершин Γx_i, смежных x_j, и процесс выбора вершин G_1 повторяют. Образованный подграф G_1 исключают из исходного и получают граф $G^* = (X^*, U^*)$, где $X^* = X \backslash X_1 U^* = U | U_1$.

Далее в графе $G^{*\prime}$ выбирают вершину с наибольшей локальной степенью, включают ее в G_2 и процесс повторяют до тех пор, пока граф G не будет разрезан на l частей.

Первоначальную компоновку можно улучшить с помощью итерационных алгоритмов, основанных на реализации методов парных или групповых перестановок элементов из одной части схемы в другую таким образом, чтобы улучшилось значение целевой функции с учетом заданных ограничений.

Размещение конструктивных модулей

Задача размещения заключается в оптимальном размещении (с точки зрения выбранного критерия оптимальности) элементов и связей между ними в монтажном пространстве типовой конструкции с учетом заданных конструктивно-технологических ограничений [9].

Исходными данными в задаче являются принципиальная электрическая схема узла или устройства, метрические параметры и топологические свойства монтажного пространства. *Главная цель размещения* – создание наилучших условий для трассировки с учетом обеспечения тепловых режимов и электромагнитной совместимости электрорадиоэлементов. Несмотря на обилие существующих критериев размещения (минимум пересечений, минимум суммарной длины соединений и т. д.), истинной целью размещения компонентов является максимальное упрощение процесса трассировки соединений, т.е. *достижение минимального числа непроведенных трасс* [5–7]. При размещении n микроэлементов в регулярном монтажном пространстве с числом позиций m общее число размещений $N(n, m)$ определяется как

$$N(n, m) = n!Cm = m_1/(m–n)!$$

В связи с этим поиск оптимального размещения с помощью перебора нецелесообразен уже при $n > 15$.

Имеется много разновидностей алгоритмов размещения. Основной идеей этих алгоритмов является идея упорядочения микроэлементов по определенным признакам. Сначала устанавливают очередность микроэлементов, а затем для каждого из них определяют наилучшую позицию по выбранному критерию, например по суммарной длине связей с уже размещенными компонентами. Затем процесс повторяют для оставшихся компонентов n свободных позиций. Связность размещаемых элементов задается матрицей смежности R графа $G = (X, V)$. Для выбора размещаемого элемента используют различные оценки связности.

Пусть на k-м шаге алгоритма размещено I_k элементов, тогда множество еще не размещенных элементов:

$$I'_k = I \setminus I_k$$

Основные правила для выбора элементов на (k+1)-м шаге алгоритма:

а) максимум суммарной связности со всеми размещенными элементами;

б) максимум разности связей между размещенными и неразмещенными элементами.

Выбранный для размещения элемент устанавливают в такую позицию среди оставшихся незаполненных, при которой будет иметь наименьшее значение некоторая целевая функция.

Для многих задач размещения в качестве такой функции может быть выбрана суммарная длина связей с уже размещенными элементами [7, 8].

Последовательные алгоритмы размещения требуют небольших затрат машинного времени, относят их к классу полиномиальных алгоритмов со сложностью $Q(n)$, приводящих к неоптимальным решениям. Улучшить решение можно применением итерационных алгоритмов компоновки, основанных на изменении позиций одиночных элементов или групп элементов. Итерационные алгоритмы также относятся к классу полиномиальных со сложностью порядка $Q(n^2) - Q(n^4)$.

Алгоритмы и модели трассировки соединений электронных средств

Задача трассировки заключается в определении конкретной геометрии печатного или проводного монтажа, реализующего соединения между элементами схемы. Исходными данными для трассировки являются список цепей, метрические параметры и топологические свойства типовой конструкции и ее элементов, а также результаты решения задачи размещения, по которым находят координаты выводов элементов.

При решении задачи трассировки строят множество трасс, соединяющих выводы элементов соответствующих цепей схемы. Разработка отдельной трассы представляет собой построение на фиксированных вершинах минимального покрывающего или связывающего дерева, а разработка множества трасс сводится к построению леса непересекающихся минимально покрывающих или связывающих деревьев. Известно, что на n вершинах можно построить n^{n-2} различных деревьев, поэтому точное решение задачи трассировки методом полного перебора практически нереализуемо.

В последовательных алгоритмах трассировки трассы цепей проводятся в определенном порядке одна за другой, при этом каждая проложенная трасса становится препятствием для всех последующих цепей. В последовательных алгоритмах выполняют локальную оптимизацию качества трассировки каждой отдельной трассы без учета влияния размещения данной трассы на возможность проведения последующих. Это приводит к тому, что некоторые участки платы могут оказаться заблокированными.

Разновидности задач трассировки. *Трассировка* монтажных соединений – это задача геометрического построения на КП всех цепей данной конструкции, координаты начала и конца которых определены при размещении элементов. Следовательно, задача трассировки состоит в отыскании геометрически определенного способа соединений эквипотенциальных выводов схемы. При этом необходимо учитывать различные конструктивно-технические ограничения: допускаются пересечения или нет, возможен ли переход с одного слоя на другой, сколько слоев отводится для трассировки, допустимые ширина проводников и расстояния между ними и т. д.

Алгоритмы трассировки существенно зависят от принятой конструкции и технологии изготовления ЭС.

Задачи трассировки можно разделить на две группы: трассировка проводных соединений и трассировка печатных (пленочных) соединений.

Трассировка проводных соединений в целом относительно более проста, поскольку отдельные сигнальные цепи электрически изолированы друг от друга. Поэтому в большинстве случаев она может быть сведена к оптимизации трасс соединений отдельных цепей. Наиболее распространенный подход к оптимизации трасс проводных соединений основан на использовании алгоритмов построения минимальных связывающих деревьев. Но и при проводном монтаже возникают проблемы совместной оптимизации соединений монтажных схем, определяемые такими факторами, как, например, электромагнитная совместимость проводов, наличие жгутов заданной формы и размера и др. В подобных ситуациях задачи трассировки проводных соединений становятся по сложности и постановке близкими к задачам трассировки печатного монтажа.

Трассировка печатных и пленочных соединений непосредственно связана с согласованием метрических и топологических параметров схемы соединений и соответствующих параметров коммутационного поля (КП).

К *метрическим параметрам* схемы можно отнести размеры элементов, ширину проводников и допустимые расстояния между ними, предельно допустимые длины соединений и т. д.

Топологические параметры схемы определяются такими ее структурными свойствами, как планарность, т. е. возможность расположения на плоскости без пересечений, минимальное число пересечений и др. Топологические параметры коммутационного поля опре-

деляются принятыми конструктивными способами устранения пересечений.

Общая характеристика методов трассировки. Проектирование схем соединений, иначе трассировка соединений, является одной из наиболее трудных задач в общей проблеме автоматизации проектирования электронных устройств. Прежде всего, это связано с многообразием способов конструктивно-технологической реализации соединений, каждый из которых обуславливает использование специфических *критериев оптимизации* при алгоритмическом решении задачи трассировки.

Исходной информацией для решения задач *трассировки* соединений являются список цепей, параметры конструкции элементов и коммутационного поля, а также данные по размещению элементов. Перед трассировкой соединений для каждой цепи схемы могут быть рассчитаны координаты расположения выводов на КП.

При алгоритмическом решении задача трассировки состоит в построении для всех цепей схемы оптимальных монтажных соединений.

Задача трассировки имеет метрический и топологический аспекты. Метрический аспект предполагает учет конструктивных размеров элементов, соединений и КП. Топологический аспект связан с выбором допустимого пространственного расположения отдельных монтажных соединений на КП при ограничениях на число пересечений соединений, число слоев коммутационной схемы.

Алгоритмические методы проводных и печатных соединений существенно различаются. Для проводного монтажа трассировка осуществляется с помощью алгоритмов построения минимальных деревьев соединений. Полная монтажная схема (таблица проводов) получается при последовательном применении указанных алгоритмов для отдельных цепей схемы. Далее, на основании анализа паразитных связей, в полученной монтажной схеме трассы отдельных соединений могут быть скорректированы.

Алгоритмические методы трассировки печатных (пленочных) соединений зависят от конструкции коммутационного поля и могут быть разделены на две основные группы. К первой относятся так называемые топографические методы, в которых приоритет отдается метрическому аспекту задачи. Вторая группа основана на графометрическом подходе задачи трассировки.

Для трассировки соединений предложено много алгоритмов, различающихся скоростью и требуемым объемом памяти при реали-

зации его на ЭВМ, а также качеством результата: волновой алгоритм и его модификации, алгоритмы трассировки по магистралям и каналам, ряд комбинированных алгоритмов. Эффективность применения каждого из них определяется рядом факторов: конструкцией коммутационного поля, ресурсами машинного времени и памяти ЭВМ, сложностью схемы соединений.

Для ряда конструкций электронных устройств разделение общей задачи проектирования топологии на два этапа – размещение элементов и трассировку соединений – не оправдано. Характерными особенностями таких конструкций являются нерегулярность расположения элементов и соединений, их разнотипность, наличие одного слоя коммутации.

Примерами могут служить односторонние печатные платы с микросхемами и навесными радиодеталями в устройствах аналогового типа, гибридные микросхемы и биполярные ИС с одним слоем коммутации. Основным критерием при разработке топологии таких схем является минимум числа пересечений соединений, а ограничением – площадь, занимаемая схемой.

В последнее время проводятся интенсивные исследования по применению графотеоретических методов к проектированию топологии схем подобного рода, поскольку последовательные топографические методы трассировки в этом случае малоэффективны.

Графотеоретические методы трассировки предполагают предварительный анализ планарности схемы, представленной в виде графа, и последующую ликвидацию пересечений с помощью технологических приемов. Окончательная фаза состоит в построении эскиза топологии схемы при рациональном распределении функции между конструктором и ЭВМ.

Трассировка проводных соединений. Монтажные соединения для цепей схемы представляют собой деревья.

Виды используемых деревьев определяются технологией выполнения соединений и схемотехническими требованиями. При автоматизированном конструировании схем проводного и печатного монтажа возникает задача построения минимальных деревьев соединений. Как правило, минимизации подлежит суммарная длина рёбер дерева.

Могут быть использованы и другие критерии оптимизации.

Задача построения минимального дерева формулируется следующим образом: пусть $P = \{p_1, p_2, \ldots, p_n\}$ – множество точек плоскости, соответствующих выводам произвольной цепи.

Рассмотрим полный граф $G(X, U)$, вершины которого $x \in X$ соответствуют выводам цепи, а рёбра $u \epsilon U$ с приписанным к ним весом $\mu(u)$ характеризуют соединения между парами выводов. Значение $\mu(u)$ может быть равно расстоянию между соответствующими точками множества P. В общем случае $\mu(u)$ может представлять линейную комбинацию нескольких характеристик соединения:

$$\mu(u) = k_1 d_1(u) + k_2 d_2(u) + \ldots k_s d_s(u),$$

где $k_1, k_2, \ldots, k_s$ – коэффициенты;

$d_s(u)$ – некоторая характеристика соединения U.

Теперь исходная задача сводится к определению в графе G дерева, включающего все вершины X и имеющего минимальный вес рёбер. Такое дерево называется минимальным покрывающим деревом или минимальным связывающим деревом.

Наиболее эффективен с точки зрения реализации на ЭВМ *алгоритм Прима*, предполагающий последовательное выполнение двух принципов:

- всякая изолированная вершина соединяется с ближайшей;
- всякий изолированный фрагмент (связанная группа вершин) соединяется с ближайшей вершиной кратчайшим ребром.

Здесь под расстоянием между вершинами понимают значение $\mu(u)$, приписанное рёбрам соответствующего графа. Расстоянием вершины от данного изолированного фрагмента является минимум его расстояний до отдельных вершин фрагмента.

На рис. 3.30 расстоянием вершины x_i от фрагмента $1, 2, 3, 4, 5$ является длина ребра $(5, x)$.

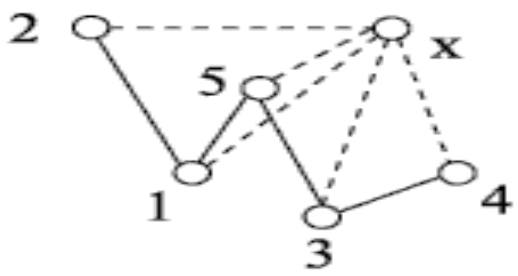


Рис. 3.30. Определение расстояния от вершины до фрагмента (5, *x*)

Алгоритм построения минимального связывающего дерева для цепи с *n* выводами теперь может быть описан следующим образом:

- для произвольного вывода цепи найти ближайший вывод и провести соединение;
- на каждом последующем шаге $i = 2, 3, \ldots, n-1$ из множества неподсоединённых выводов выбрать тот, который находится ближе остальных (в указанном выше смысле) к группе уже связанных выводов, и подсоединить его к этой группе по кратчайшему пути.

Построенное таким образом дерево будет иметь минимальную суммарную длину соединений.

Иногда при построении связывающего дерева в качестве значения $\mu(u)$ принимают суммарную оценку, включающую как длину *ребра* $d(u)$, так и число пересечений $h(u)$ этого *ребра* с рёбрами уже построенных деревьев:

$$\mu(u) = k_1 d(u) + k_2 h(u).$$

В частности, такая оценка используется при построении связывающих деревьев для схем печатного монтажа. В этом случае процедура Прима остаётся без изменений, а расстояние между выводами цепи рассчитывается по формуле $\mu(u) = k_1 d(u) + k_2 h(u)$.

Построение минимального дерева с ограничением на степени вершин может быть осуществлено при использовании процедур, основанных на *методе ветвей и границ*. Однако для практических целей предпочтение следует отдавать эвристическим алгоритмам.

В частности, можно использовать модифицированные принципы Прима:

- всякая изолированная вершина соединяется с ближайшей, не соединенной с λ другими вершинами;
- всякий изолированный фрагмент соединяется кратчайшим ребром с ближайшей вершиной, не соединённой с λ другими вершинами.

Приведённые в научной литературе исследования показывают, что алгоритм, построенный на основании этих принципов, приводит к получению деревьев с длиной, превышающей минимальную не более чем на 5 % при числе выводов $n \leq 15$.

Модифицированные принципы Прима используются иногда при параллельном наращивании нескольких фрагментов дерева. Такой

способ даёт деревья с меньшей длиной соединений последовательного наращивания одного изолированного фрагмента.

В некоторых случаях, помимо ограничения на степени вершин связывающего дерева, задаётся начальная и конечная точка цепи. Например, это имеет место при разработке монтажных схем для высокочастотных цепей, когда необходимо связать в определённой последовательности источник сигнала и несколько нагрузок. Тогда задача сводится к построению кратчайшего пути между двумя заданными выводами, проходящего через все остальные выводы цепи.

Данная задача родственна задаче о маршруте коммивояжера, но отличается от последней тем, что путь обхода должен быть разомкнутым и соединять две заданные точки. Следуя терминологии теории графов, возникает задача построения кратчайшей гамильтоновой цепи между заданными начальной и конечной вершинами.

Рассмотрим алгоритм, дающий приближённое решение этой задачи. Основу алгоритма составляет $(n-1)$-й шаговый процесс:

- выбор кратчайших рёбер в полном графе G;
- проверка каждого ребра на выполнение ограничений задачи;
- составление из выбранных рёбер пути, соединяющего заданные точки.

Пусть задано расположение точек (рис. 3.31).

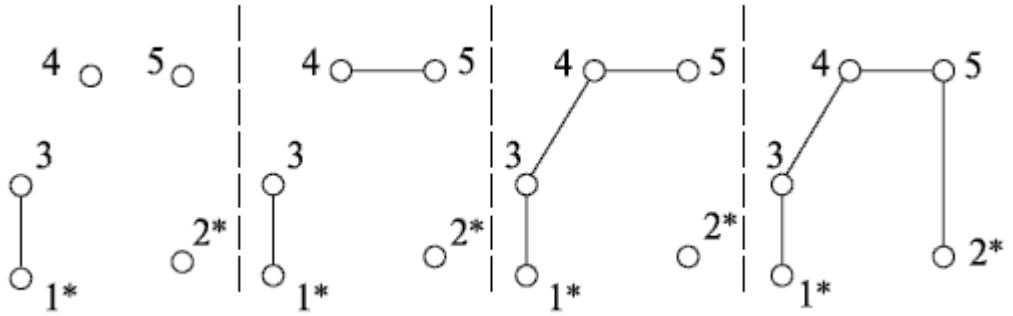


Рис. 3.31. Построение пути

Здесь 1^* и 2^* – соответственно начальная и конечная точка пути.

Составим упорядоченную по возрастанию длин последовательность рёбер полного графа G: (1*–3), (1*–2*), (2*–3), (4–5), (3–4), (3–5), (1–4*), (2*–5), (2*–4), (1*–5).

Очередное ребро $i = 1, 2, \ldots, n-1$ выбирается по порядку из этой последовательности при выполнении условий:

1) ребро не соединяет заданные конечную и начальную точки (1^* и 2^*);

2) при включении ребра в путь степень вершин, соединяемых этим ребром, не превышает допустимой ($\lambda = 1$ – для начальной и конечной точек и $\lambda = 2$ – для остальных точек);

3) ребро не образует цикла с рёбрами, уже включенными в путь;

4) при включении в путь любого ребра, кроме $(n-1)$-го, начальная и конечная точки остаются несвязанными.

Условия 1–3 непосредственно вытекают из ограничений задачи. Условие 4 препятствует образованию тупиковых ситуаций, т.е. такого положения, при котором дальнейшее формирование пути становится невозможным – все подсоединенные точки, кроме начальной и конечной, имеют степень $\lambda = 2$. Пошаговый процесс формирования пути изображен на рис. 3.31.

Шаг 1. Выбираем ребро $1^* - 3$, так как оно удовлетворяет всем условиям (рис. 3.31, *а*).

Шаг 2. Ребро $1^* - 2^*$ отбрасывается, так как не удовлетворяется условие 1, а ребро $2^* - 3$ – так как не удовлетворяется условие 4. Выбирается ребро 4–5 (рис. 3.31, *б*).

Шаг 3. Выбирается ребро 3–4 (рис. 3.31, *в*).

Шаг 4. Ребра 3–5 и 1^*–4 отбрасываются из-за невыполнения условия 3. Выбирается ребро $2^* - 5$. Результирующий путь $1^* - 3 - 4 - 5 - 2^*$ показан на рис. 3.31, *г*.

Если снять ограничение о крайних точках пути, то данный алгоритм приводит к более короткому пути: 2 – 1 – 3 – 4 – 5. В этом случае алгоритм становится частным случаем модифицированного алгоритма Прима.

Использование описанных выше процедур для построения связывающих деревьев с ограниченной степенью вершин обеспечивает вполне приемлемые результаты.

Трассировка печатного (пленочного) монтажа. Волновой алгоритм трассировки и его модификации

Главные принципы волнового алгоритма Ли заключаются в следующем. Плоскость трассировки разбивают на прямоугольные площадки — дискреты заданного размера. Размер дискретной площадки определяется допустимыми размерами проводников и расстояниями между ними. Задача проведения трасс сводится к получению после-

довательности дискретов, соединяющих элементы *a* и *b,* соответствующие началу и концу проводимой трассы.

Введем целевую функцию $F = F(f_1,...,f_l)$ как критерий качества пути. Начиная с элемента *a* дискретам, соседним с ранее просмотренными, присваивают определенное значение целевой функции $Fij = m(i, j)$. Этот этап проводится итерационно до элемента *b,* которому присваивают некоторое значение веса $m(i_b, j_b)$. Затем, начиная от элемента *b,* значения перемещаются к элементу *a* по пройденным дискретам таким образом, чтобы значения целевых функций дискретов монотонно убывали. В результате получается трасса, соединяющая элементы *a* и *b.*

Работа алгоритма Ли реализуется следующим образом. На трассируемой плоскости из источника *a* моделируется распространение волны до тех пор, пока не будет достигнута точка *b* или пока на некотором шаге фронт волны не сможет включить ни один незанятый дискрет. Эту часть алгоритма называют *распространением волны.*

После этого проводят трассу, начиная от конечной точки, по дискретам с последовательно уменьшающимися весами (рис. 3.32).

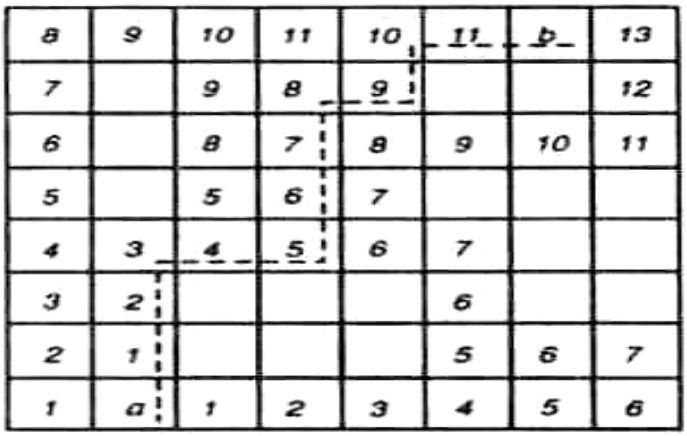


Рис. 3.32. Пример реализации волнового алгоритма

Цифры в квадратах соответствуют весам дискретов, занятые дискреты заштрихованы, а построенная трасса показана штриховой линией. Существует несколько вариантов проведения пути, из которых конструктор (или ЭВМ) выбирает один, наиболее удовлетворяющий заданным требованиям.

Имеется многообразие волновых алгоритмов, направленных на повышение быстродействия трассировки, уменьшение объема тре-

буемой оперативной памяти ЭВМ и т. д. Волновые алгоритмы применяются при разработке сетей ЭВМ [15, 16].

Многие методы трассировки печатных соединений основаны на идеях волнового алгоритма, предложенного Ли. Последний представляет собой развитие алгоритмов построения кратчайших путей в сети и позволяет находить маршруты соединений, оптимальные по ряду параметров.

КП разбивается на элементарные ячейки. Размеры ячеек и их количество определяются:

- площадью поля;
- допустимой плотностью расположения выводов элементов и проводников.

Выбранная система ячеек устанавливает среду, в которой осуществляется построение соединений. В простейшем случае ячейка представляет собой квадрат со стороной *h*, равной расстоянию между средними линиями двух соседних печатных проводников (рис. 3.33).

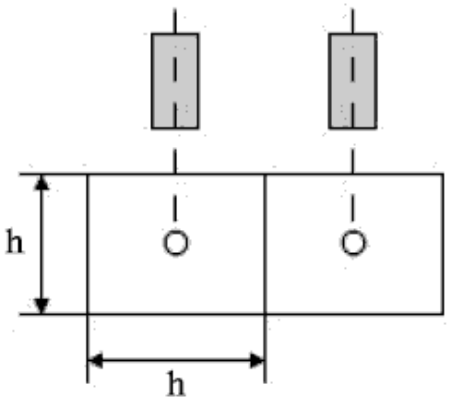


Рис. 3.33. Ячейки коммутационного поля

Если размеры поля по горизонтали и вертикали соответственно A_x и B_y, то получим дискретное рабочее поле (ДРП) с $N_x \times N_y$ ячейками:

$$N_x = \{A_x/h\}; N_y = \{A_y/h\},$$

где $\{A_x/h\}$ и $\{A_y/h\}$ – символы ближайшего большего целого.

Так формируется дискретное рабочее поле. В данном ДРП определяется множество занятых ячеек, соответствующее зонам, запрещённым для проведения соединений: выводы элементов, технологические области, ранее проведённые соединения и пр. По мере прове-

дения соединений множества занятых и свободных ячеек изменяются.

Основу всех модификаций алгоритма Ли составляет процедура построения оптимального в заданном смысле пути между двумя известными ячейками ДРП. Процедура состоит из двух этапов: поиска и проведения пути.

На первом этапе из одной из заданных ячеек ДРП – источника моделируется распространение числовой волны до тех пор, пока её фронт не достигнет второй отмеченной ячейки ДРП. В первом случае искомый путь существует, во втором – нет.

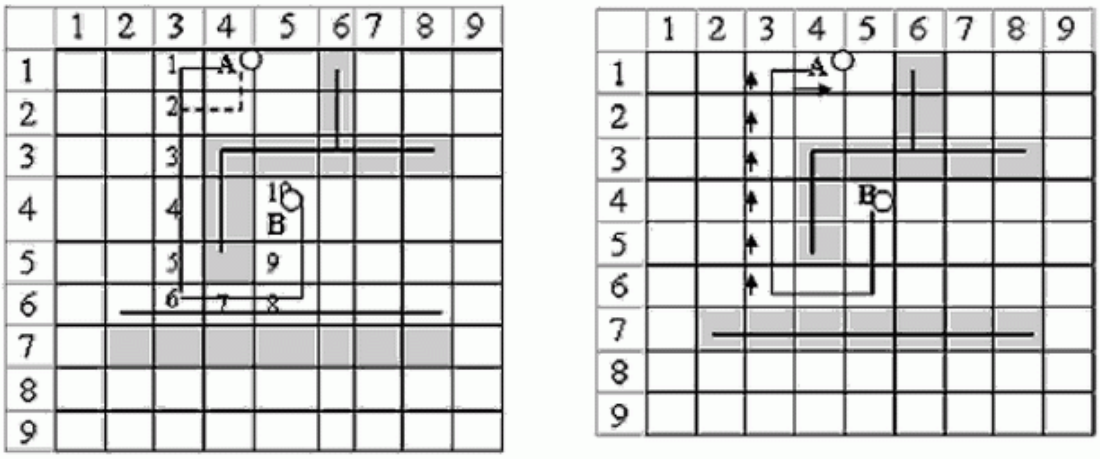
Все условия, которые необходимо выполнить при проведении соединения, в том числе и условия оптимальности, должны быть заложены в правила движения волны, т.е. в правила построения её очередного фронта. В процессе распространения волны ячейкам ДРП присваиваются весовые оценки, связанные с принятым критерием оптимальности.

На втором этапе алгоритма осуществляется проведение пути. Для этого следует, начиная от ячейки-цели, двигаться в направлении, противоположном направлению распространения волны, переходя последовательно от ячейки с большим весом к соседней ячейке с меньшим весом, до тех пор, пока не будет достигнута ячейка-источник. Ячейки ДРП, выделенные в ходе указанного процесса, и определяют искомую оптимальную трассу.

Использование путевых координат при распространении волны позволяет исключить вычисления и хранение весов ячеек ДРП (рис. 3.34, *а*). Назначение путевой координаты ячейке C_i, в случае если имеется несколько соседних ячеек фронта Φ_{k-1}, производится согласно выбранному правилу приоритетов. Например: $\uparrow, \rightarrow, \downarrow, \leftarrow$. Этап проведения пути состоит в отслеживании путевых координат в размеченном ДРП начиная от ячейки-цели (рис. 3.34, *б*).

В методе путевых координат ячейка ДРП может быть в одном из следующих состояний: пустая, занятая или содержать координату: $\uparrow, \rightarrow, \downarrow, \leftarrow$. На основании этой последовательности выделяется искомый путь в ДРП. В неопределённых ситуациях, так же как и в основной схеме алгоритма, должно быть использовано некоторое правило приоритетов.

Рассмотренный выше вариант волнового алгоритма может быть применим для трассировки однослойных соединений, когда пересечения между отдельными проводниками запрещены.



а *б*

Рис. 3.34. Использование путевых координат:
а – распространение волны; *б* – проведение пути

Трассировка по магистралям

Волновой алгоритм в той или иной модификации является основой большинства развитых программ машинной трассировки ввиду его универсальности. Однако в определённых ситуациях эффективнее применять другие алгоритмы, которые дают более быстрое и качественное решение.

Алгоритм построения соединений с малым числом поворотов (лучевой алгоритм). Это такой алгоритм трассировки, в котором основные процедуры поиска и проведения пути осуществляются путём исследования пространства магистралей (линий), а не ячеек ДРП, как в классическом алгоритме Ли.

Для построения процесса рассмотрим ДРП (рис. 3.35).

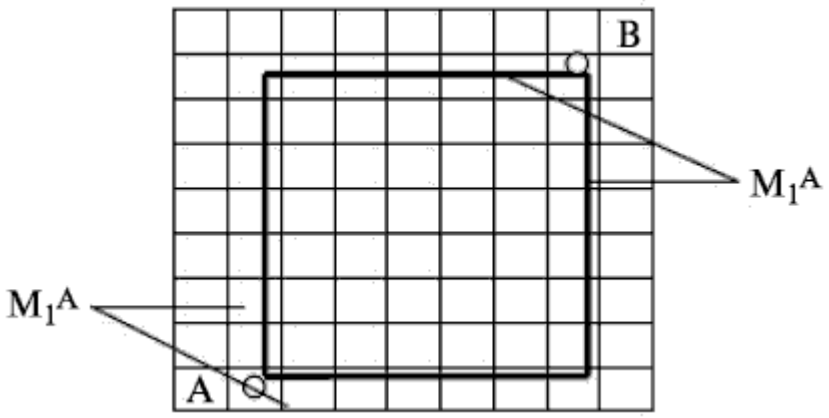


Рис. 3.35. Магистрали первого уровня

Пусть требуется найти соединение между точками A и B. Построим из точек A и B лучи в горизонтальном и вертикальном направлениях, используя лишь свободные ячейки поля. Будем считать их магистралями первого уровня (фронта) и обозначим соответственно M_1^A и M_1^B.

В простейшем случае, когда эти магистрали пересекутся, сразу получаем искомое соединение.

При использовании алгоритма Ли для построения этого же пути пришлось бы последовательно рассмотреть состояние всех ячеек, по крайней мере, входящих в прямоугольник, содержащий ячейки A и B.

Если магистрали M_1^A и M_1^B не пересекаются, строим магистрали второго уровня (фронта). Соответственно, для точек A и B – M_2^A и M_2^B (рис. 3.36).

Эти магистрали перпендикулярны к магистралям первого уровня и проводятся через точки, расположенные в узлах основной сетки. Назовём эти точки базовыми.

При переходе с магистралей первого уровня на магистрали второго уровня в возможную конфигурацию соединения добавляется один поворот.

Использование алгоритма трассировки по магистралям приводит к минимизации числа переходов при трассировке двухслойных схем с ортогональными соединениями. Такие пути называются простейшими или малоповоротными.

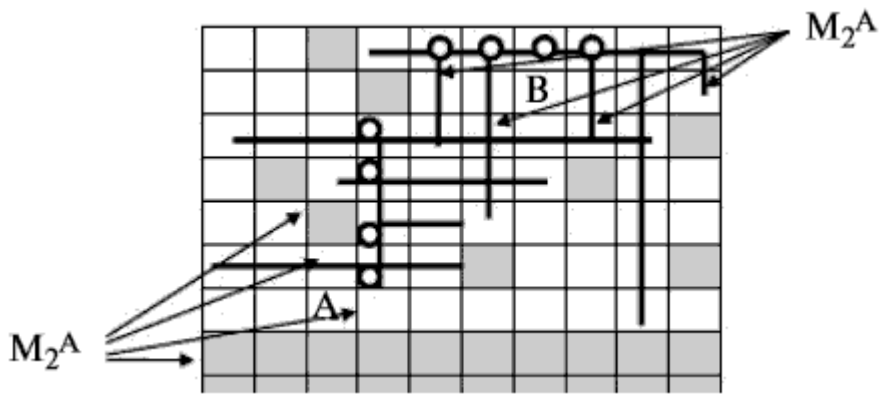


Рис. 3.36. Магистрали второго уровня

В общем случае процесс построения магистралей $(i+1)$-го уровня состоит в выборе базовых точек на магистралях *i*-го уровня и проведении через них отрезков нормалей, не пересекающих области, запрещённые для проведения соединений.

Построение магистралей каждого следующего уровня выполняется попеременно, например, для точек A и B до тех пор, пока либо очередной уровень не может быть образован (проведение пути невозможно), либо на очередном шаге некоторая магистраль из M_i^A пересечётся с магистралью из M_j^B. В последнем случае может быть построен путь, содержащий $i + j - 1$ поворот (переход).

Для минимизации длины пути применяется следующая процедура.

Пусть магистраль уровня i точки A пересекла несколько магистралей уровня j точки B. Тогда выбирается сначала наименьший отрезок магистрали M_i^A, обеспечивающий пересечения с одной из магистралей M_j^B, далее – наименьший отрезок из M_{i-1}^A, обеспечивающий соединение с выбранным отрезком i-го уровня, и т.д. до достижения точки A. Аналогичная процедура применяется для магистралей всех уровней точки B. Результирующий путь $l(A, B)$ состоит из выбранных отрезков и содержит минимальное число поворотов.

Рассмотренный принцип трассировки по магистралям с незначительной модификацией может быть использован для построения многоконцевых соединений.

Следует отметить, что при программной реализации алгоритма на ЭВМ можно отказаться от матричного способа кодирования КП. Поэтому время работы алгоритма и требуемый объём памяти не зависят напрямую от размеров КП, а определяются, главным образом, сложностью конфигураций уже приложенных соединений. В связи с этим рассировка по магистралям с точки зрения затрат времени и памяти ЭВМ более эффективна, чем волновой алгоритм, для полей большой площади, причём требуемые затраты времени существенно меньше для слабозаполненных полей. По мере заполнения поля трассами соединений эффективность процесса поиска снижается, и временные затраты ЭВМ приближаются к затратам времени волнового алгоритма.

Осуществим трассировку схемы с использованием волнового и лучевого алгоритмов (рис. 3.37).

Итак, в ходе трассировки монтажных соединений заданной схемы была решена задача геометрического построения на коммутационном поле всех цепей конструкции. Однако исключить все пересечения не удалось. Использовать двухслойный монтаж для этой схемы

не имеет смысла, так как он будет очень объёмным и дорогостоящим при больших объёмах производства.

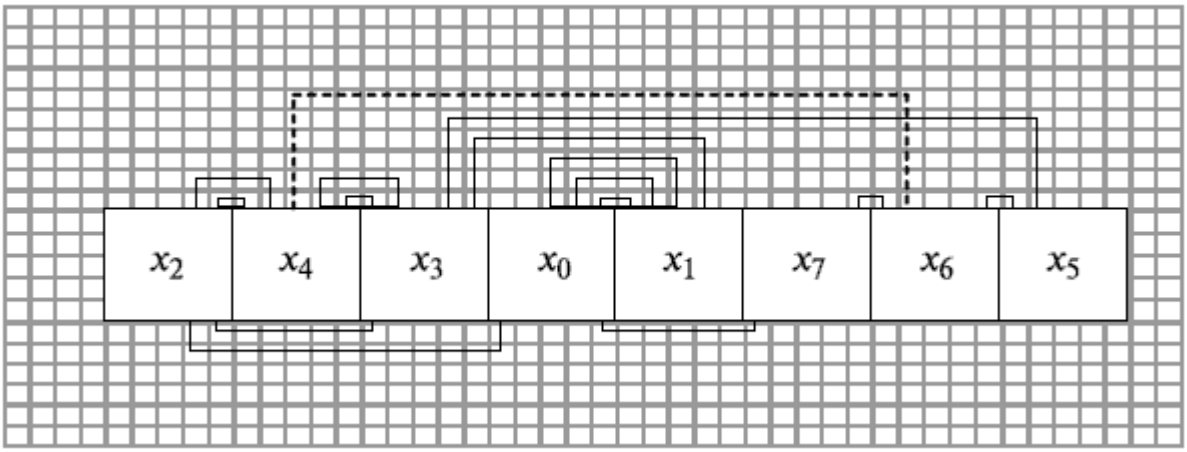


Рис. 3.37. Трассировка схемы по магистралям

Целесообразнее показанное пунктиром соединение сделать проводным, впаяв перемычку в отверстия с контактными площадками, или, применяя более современную базу, над соединяющей элементы x_3 и x_5 дорожкой припаять безвыводную перемычку.

Рассмотренные алгоритмы показывают эффективность их комплексного использования в зависимости от конкретных условий.

Сквозное автоматизированное конструкторско-технологическое проектирование модулей РЭС в САПР

Основная тенденция развития современных систем автоматизированного проектирования – создание комплексов программных средств автоматизации проектирования и производства изделия. Современная система проектирования печатных плат представляет собой сложный комплекс программ, обеспечивающий сквозной цикл проектирования и технологической подготовки производства, начиная с прорисовки принципиальной схемы и заканчивая генерацией управляющих файлов для оборудования по изготовлению фотошаблонов, сверлению отверстий, сборке и электроконтролю.

При использовании систем автоматизированного проектирования рекомендуется применять методику, укрупненный алгоритм которой приведен на рис. 3.38 (в скобках указаны программные средства САПР P-CAD) и состоит из следующих этапов:

1. Создать перечень элементов схемы электрической принципиальной.

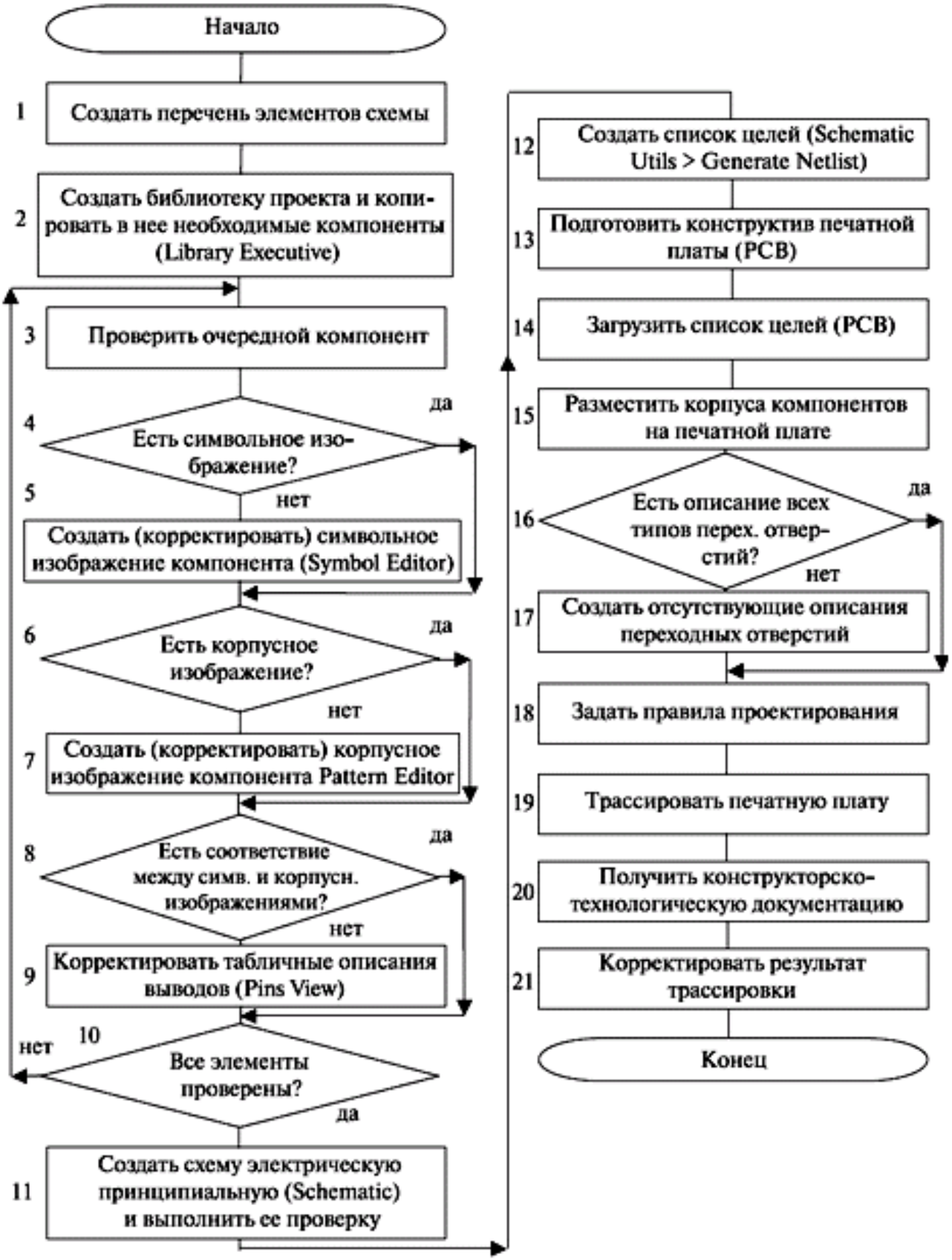


Рис. 3.38. Методика сквозного автоматизированного проектирования печатных узлов

2. Создать библиотеку проекта и копировать в нее необходимые компоненты в соответствии с перечнем элементов (менеджер библиотек Library Executive).

3. Выбрать из библиотеки проекта очередной компонент.

4. Проверить наличие символьного изображения компонента. Если изображение имеется, то следует перейти к этапу 6, иначе – к этапу 5.

5. Создать или корректировать символьное изображение компонента (редактор *Symbol Editor*).

6. Проверить наличие конструкторско-технологического (корпусного) изображения компонента. Если изображение имеется, то следует перейти к этапу 8, иначе – к этапу 7.

7. Создать или корректировать корпусное изображение компонента (редактор *Pattern Editor*).

8. Проверить, установлено ли соответствие между символьным и корпусным изображением компонента. Если соответствие установлено, то следует перейти к этапу 10, иначе – к этапу 9.

9. Корректировать табличные описания выводов компонента, устанавливая связь между символьным и корпусным изображением компонента (менеджер библиотек *Library Executive,* окно *Pins* View).

10. Если проверены все компоненты, то следует перейти к этапу 11, иначе – к этапу 3.

11. Создать схему электрическую принципиальную (редактор *Schematic*) и выполнить ее проверку для выявления как синтаксических ошибок встроенными средствами P-*CAD* (*ERC*), так и системных ошибок, используя внешние программы моделирования аналого-цифровых схем (например, *Protel*). Если обнаружены ошибки, следует исправить их и повторить проверку схемы. Если ошибок не обнаружено, то следует перейти к этапу 12.

12. Создать список цепей схемы (редактор *Schematic,* команда *Utils > Generate Netlist*) для последующего переноса в редактор печатных плат.

13. Подготовить конструктив печатной платы (редактор печатных плат PCB).

14. Загрузить список цепей схемы (PCB).

15. Разместить корпуса компонентов на печатной плате. Эта операция может выполняться вручную или автоматизированными методами.

16. Проверить, имеются ли описания всех типов переходных отверстий. Если описания есть, то следует перейти к этапу 18, иначе – к этапу 17.

17. Создать отсутствующие описания переходных отверстий.

18. Задать правила проектирования. Необходимо задать число слоев печатной платы, указать ширину проводников и величину зазоров между проводниками, указать ограничения, которые должны учитываться при трассировке печатной платы.

19. Трассировать печатную плату. Трассировка может выполняться вручную, автоматизированным или автоматическим способом. Как правило, используется комбинация этих способов.

3.8. Примеры систем автоматизированного проектирования

Информационные технологии проектирования печатных плат

Любая система проектирования печатных плат представляет собой сложный комплекс программ, обеспечивающий сквозной цикл, начиная с прорисовки принципиальной схемы и заканчивая генерацией управляющих файлов для оборудования изготовления фотошаблонов, сверления отверстий, сборки и электроконтроля.

В настоящее время существует достаточно большой спектр систем автоматизированного проектирования печатных узлов РЭС. Выбор конкретной САПР определяется назначением устройства, функциональной ориентированностью САПР (цифровые, аналоговые, аналого-цифровые, СВЧ-устройства), стоимостными и сервисными характеристиками (приобретение, сопровождение, обучение).

Наилучших результатов добилась компания Mentor*Graphics* (www.mentor.com/pcb). Имея собственную систему проектирования печатных плат *Mentor BoardStation*, компания поглотила двух своих конкурентов – компании Verybest и Innoveda и сейчас продолжает развивать линии продуктов Expedition PCB и PADS PowerPCB. Ключом к успеху компании явилась ориентация на современные интегрированные среды проектирования для *Windows*.

Пакет ***Expedition PCB*** представляет сейчас наиболее мощное решение в области проектирования плат. Основу системы составляет среда AutoActive, позволяющая реализовать такие функции, как предтопологический анализ целостности сигналов, интерактивная и автоматическая трассировка с учётом требований высокочастотных

плат и специальных технологических ограничений, накладываемых использованием современной элементной базы.

Единая среда позволяет моделировать наводки в проводниках непосредственно при прокладке трассы или шины и контролировать превышение ими заданного уровня. У данного продукта можно отметить только один недостаток – его высокую стоимость.

Другой продукт компании Mentor – система***PADS PowerPCB*** (www.pads.com) – предлагает более дешёвое решение. Эта система может похвастаться лучшим автотрассировщиком BlaseRouter, поддерживающим все необходимые при трассировке высокочастотных плат функции. Пакет имеет модули предтопологического и посттопологического анализа, тесно взаимодействующие с системой контроля ограничений.

Далее по мощности предлагаемых решений идёт компания Cadence. Для верхнего уровня проектирования предлагается пакет ***PCB Design Studio*** (www.pcb.cadence.com). В качестве редактора печатных плат здесь используется программа ***Allegro***, позволяющая разрабатывать многослойные и высокоскоростные платы с высокой плотностью размещения компонентов. В качестве штатного модуля авторазмещения и автотрассировки здесь используется программа ***SPECCTRA*** (www.specctra.com), управляемая обширным набором правил проектирования и технологических ограничений. Выполняется анализ электромагнитной совместимости.

Другой продукт компании Cadence – пакет ***OrCAD*** (www.orcad.com) – рекомендуется как более легкое и дешевое решение для проектирования печатных плат. Данный пакет рассматривается фирмой Cadence как приоритетная система ввода проектов и моделирования: модули *Capture CIS* и *PSpice* сейчас поставляются в составе пакета *PCB Design* Studio. Редактор печатных плат OrCAD *Layout* имеет три различные конфигурации с разными функциональными возможностями. В проекте платы здесь может присутствовать до 30 слоев, 16 из которых могут быть сигнальными. Имеются встроенные средства авторазмещения и автотрассировки, а также интерфейс с программой SPECCTRA. Однако главным модулем здесь является не редактор печатных плат, а редактор принципиальных схем OrCAD *Capture CIS*, оснащенный единственной в своем роде системой управления базами данных компонентов.

Система *CIS* (*Component Information System*) была разработана для обеспечения всем пользователям OrCAD доступа через Интернет

к централизованным базам данных компонентов на сайте www.spincircuit.com. Гибкость системы *CIS* позволяет организовать корпоративные базы разрешенных к применению компонентов и работать в локальных сетях, а также использовать процедуры автоматизированного нормоконтроля.

Третий производитель САПР печатных плат – австралийская компания Altium *Technologies* (www.altium.com). Благодаря умелой инвестиционной политике эта фирма смогла свести до минимума потери, связанные со спадом рынка высоких технологий в 2002 году. В августе 2002 года компания выпустила в свет пакет ***Protel DXP*** (*www.protel.com*), представляющий собой продолжение собственной оригинальной линии продуктов Protel. Этот пакет обеспечивает сквозной цикл проектирования смешанных аналого-цифровых печатных плат с использованием программируемой логики фирм Xilinx и Altera. Весь инструментарий реализован на базе интегрированной среды проектирования, работающей под управлением *Windows* XP. К имевшимся ранее средствам посттопологического анализа целостности сигналов добавилась возможность выполнять предтопологический анализ.

Компания Altium продолжает развивать свой второй пакет проектирования печатных плат – ***P-CAD*** (*www.pcad.com*). Пакет предназначен для проектирования многослойных печатных плат электронных устройств. Внедрен бессеточный автотрассировщик *Shape*-Based *Router*, в котором применен алгоритм оптимизации нейронных сетей.

Эта система остается достаточно популярной в России, что обусловлено, с одной стороны, хорошей функциональностью программы, а с другой – по ассоциации с распространенными на радиоэлектронных предприятиях ранними версиями PCAD. В 1996 г. фирма ACCEL *Technologies* впервые представила версию широко известной системы разработки печатных плат P-*CAD* на платформе *Windows*. Обновленный продукт получил новое название –ACCEL *EDA*. С этого момента продукт ACCEL *EDA* приобрел широкую популярность среди разработчиков электронных устройств. В сентябре 1999 г. вышла последняя, 15-я версия продукта. 17 января 2000 г. произошло слияние двух ведущих разработчиков систем САПР печатных плат – фирм Protel *International* и ACCEL *Technologies*, которые объединили свои совместные усилия под торговой маркой Protel (ныне Altium). С марта 2000 г. продукт ACCEL *EDA* сменил свое название на P-*CAD*.

Система P-*CAD* выполняет полный цикл проектирования печатных плат, а именно:

- графический ввод электрических схем;
- смешанное аналого-цифровое моделирование на основе ядра SPICE3;
- упаковку схемы на печатную плату;
- интерактивное размещение компонентов;
- интерактивную и автоматическую трассировку проводников;
- контроль ошибок в схеме и печатной плате;
- выпуск документации;
- анализ целостности сигналов и перекрестных искажений;
- подготовку файлов Gerber и NC *Drill* для производства печатных плат;
- подготовку библиотек символов, топологических посадочных мест и моделей компонентов.

Основные возможности P-CAD :

- Удобный пользовательский интерфейс для Windows.
- Хранение проектной информации в бинарных и текстовых файлах.
- Удобная справочная система.
- Проект схемы может содержать 999 листов, проект платы – до 999 слоев (11 из них – стандартные).
- Число цепей в проекте – до 64 000.
- Число вентилей в компоненте – до 5000.
- Максимальное число выводов у компонента – 10 000.
- Максимальные размеры листа схемы или чертежа печатной платы – 60×60 дюймов.
- Поддержка дюймовой и метрической систем мер.
- Предельное разрешение – 0,0001 дюйма (0,1 мила), или 0,01 мм (10 микрон).
- Минимальный угол поворота компонентов на плате – 0,1 град.
- Длина имен компонентов – до 30 символов, максимальный объем текстовых надписей и атрибутов – до 20 000 символов.
- Механизм переноса изменений печатной платы на схему и наоборот.
- Библиотеки компонентов, содержащие более 27000 элементов и сертифицированные по стандарту *ISO 9001*.

Система обладает программой посттопологического анализа электрических характеристик печатных плат с учетом паразитных параметров реальных конструкций, позволяет поддерживать CAM-технологии благодаря встроенным функциям генерации управляющих программ для технологического оборудования, поддерживает форматы файлов для обмена информацией с программными средствами OrCAD, Protel, системами автоматизированного конструкторского проектирования *AutoCAD*, SolidWorks, Компас.

Перечисленные достоинства, наличие обученного персонала, внедренные и апробированные посттрансляторы управляющих программ для имеющегося на предприятиях оборудования определяют широкое использование САПР P-*CAD* при разработке печатных узлов РЭС.

Достаточно мощный и популярный в мире продукт – ***Visula*** компании ZUKEN (www.zuken.com). Продукты этой компании обеспечивают сквозной цикл проектирования и предлагают мощные средства моделирования и синтеза программируемой логики с последующей разработкой печатной платы. Здесь имеется стандартный набор инструментария, а также собственные средства авторазмещения и автотрассировки. Следует отметить, что компания ZUKEN также предлагает пользователям интегрированные средства трёхмерного твердотельного моделирования разрабатываемых устройств.

CircuitMaker, разработанный фирмой *MicroCode Engineering*, после слияния ее с компанией Protel стал предлагаться как самое дешевое решение для проектирования несложных печатных плат. Стандартная версия позволяет разрабатывать платы, содержащие до шести сигнальных слоев и до двух слоев металлизации. Этот продукт имеет удобный и гибкий редактор схем, а также программу моделирования.

Автоматическое размещение и трассировка реализуются и в ряде других систем проектирования печатных плат, в частности в отечественной САПР ***RELIEF*** с оригинальным алгоритмом быстрой плотной упаковки разногабаритных элементов. Алгоритм основан на многократном дихотомическом делении множества размещаемых элементов.

Применение САПР значительно сокращает сроки разработки изделий. Моделирование (элементы CAE-систем) позволяет еще до этапа макетирования выявить большое число ошибок схемотехнической реализации радиоэлектронных устройств. Автоматизация конструирования (CAD-системы) повышает качество конструкторской доку-

ментации (КД) при разработке изделия и значительно сокращает время внесения изменений в КД при доводке РЭС. Использование элементов CAM-систем для получения технологической документации сокращает сроки технологической подготовки производства.

Современной тенденцией развития САПР является интеграция различных видов проектирования (схемотехническое, конструкторское, технологическое). Сквозные системы проектирования сочетают в себе элементы *CAE*/*CAD*/*CAM* -систем. Это позволяет уменьшить число ошибок при передаче информации от одного этапа проектирования к другому, по результатам моделирования оценить качество принятых решений на каждом этапе разработки и в случае необходимости оперативно внести изменения в КД как текущего, так и предыдущих этапов. Все это повышает качество изделия и сокращает время полного цикла разработки, доводки и внедрения изделия в производство.

Программы конструкторского проектирования РЭС

Существуют чисто конструкторские пакеты, обеспечивающие более полное решение различных задач конструкторского проектирования РЭС.

Пакет программ P-CAD фирмы *Personal CAD* Systems Inc – это полное комплексное программное решение для проектирования электронных устройств, в частности ввода схемы и проектирования схемной печатной платы. Комплексное решение предполагает, что логика, описанная в схеме, воплощается в топологию печатной платы. Программы осуществляют функции логического моделирования, проверяют соблюдение правил проектирования, создают список соединений для моделирования, автоматически размещают компоненты, трассируют печатную плату и создают документы для автоматизированных производственных систем. Пакет содержит взаимодействующие средства проектирования, удобную для пользователя оболочку и интеллектуальную базу данных, обширную библиотеку, диалоговые редакторы, средства сопряжения с популярными средствами анализа. Пакет имеет открытую архитектуру, обеспечивает выдачу готовых документов для технологии монтажа и другую проектную документацию.

Вывод документации после контроля на дисплее может осуществляться на принтер, плоттер или фотоплоттер. Оболочка системы помогает пользователю двигаться сквозь процесс проектирования

с помощью меню, подсказок и правок. Система проектирования печатной платы обеспечивает средства для полной разработки топологии: от диалогового редактора до автоматического размещения компонентов, автотрассировки, проверки соблюдения правил проектирования и сопряжения с производством.

Библиотека пакета содержит обширную информацию о компонентах электронных схем – от дискретных и электромеханических деталей до существующих и заказных микросборок интегральных схем. Программные средства сопряжения превращают данные из списка соединения компонентов схемы в формат, необходимый для конкретной программы моделирования цифровой и аналоговой схемы (типа PSPICE). Пакет позволяет проектировать печатные платы, имеющие до 500 элементов и 2000 связей.

Пакет программ OrCAD фирмы Or *CAD System* Corp. является законченным и гибким программным блоком схемотехнического и конструкторского проектирования. Он обеспечивает ввод и вывод на печать принципиальных схем, трассировку печатной платы и другие операции. Пакет управляется с помощью иерархической разветвленной системы меню, легок в обучении, обладает многими дополнительными возможностями ввода и вывода схем.

Библиотека пакета содержит более 2700 изображений компонентов РЭС; можно легко создавать собственные начертания элементов. Простым нажатием клавиши выполняются многие графические операции при вводе и выводе схем: увеличение и уменьшение масштаба, преобразование (вращение, перенос, отображение) элементов и любых заданных фрагментов схемы. В системе предусмотрены создание перечня элементов (спецификаций), возможность разведения проводников, шин, входов модулей.

Пакет OrCAD в настоящее время является самым удобным и богатым по своим возможностям для ввода и вывода графических изображений принципиальных схем РЭС.

Пакет имеет удобный выход на подсистемы моделирования и анализа РЭС, а также другие графические пакеты (PSPICE, P-CAD).

Пакет универсального назначения AutoCAD фирмы AutoDesk разработан на самом современном уровне машинной графики и предоставляет разработчику исключительно широкие возможности проектирования разнообразных объектов, технических систем и устройств: домов, печатных плат, станков, деталей и одежды. Пакет представляет собой систему автоматизированной разработки черте-

жей, причем чертежи, рисунки и схемы создаются в интерактивном режиме, управляемом системой иерархических меню. В любой чертеж может быть вставлен поясняющий текст. В набор функций входят панорамирование, увеличение, масштабирование, поворот, секционирование, штриховка и другие операции преобразования изображений. В системе предусмотрены подсказки в любом состоянии и для любой команды.

В пакете разработан богатый выбор драйверов графических устройств – графических дисплеев, матричных принтеров, графических планшетов и плоттеров. Одним из важнейших достоинств пакета является возможность работы с трехмерной графикой, позволяющей строить реальные объекты, которые можно наблюдать в различных ракурсах (при желании невидимые линии на изображении стираются). Применен специальный метод полилиний для вывода сложных кривых контуров деталей.

Система *AutoCAD* непрерывно совершенствуется. Так, в последние версии системы включен интерпретатор языка Auto *Lisp* – одной из версий языка *LISP*, широко применяемого в символьной обработке и в системах искусственного интеллекта. Этот язык позволяет пользователю, с одной стороны, определять собственные функции и команды в среде *AutoCAD*, а с другой – обеспечивать связь *AutoCAD* с другими приложениями.

Продукт *AutoCAD* позволяет разрабатывать сложные качественные проекты и выпускать сопроводительную документацию к ним. С помощью улучшенных средств моделирования проектируются любые объекты и поверхности. *AutoCAD* обеспечивает быстрое оформление документации и анализ проектов в виртуальной трехмерной среде. Проектные идеи можно визуализировать в формате PDF, реализовать в макетах, получаемых посредством 3D-печати.

Сейчас появляются еще более сложные системы, включающие не только язык программирования, но и экспертные системы (экспертные настройки) для принятия решений и подсказок конструктору в процессе разработки. В эти настройки включен набор правил и математических моделей; конструктор в процессе работы может получить советы по оптимальному выбору тех или иных параметров разрабатываемой системы.

Известным производителем программного обеспечения для проектирования интегрированных электронных систем, печатной платы

и формирования технической документации является комания **Cadence Design Systems** Inc.

Программно-технический центр *ПТЦ САПР* обеспечивает сквозной маршрут автоматизированного проектирования микроэлектронной аппаратуры и элементной базы. Средства *САПР* используются при создании перспективных образцов аппаратуры, а также являются инструментом разработки сверхбольших интегральных схем (СБИС), в частности, «систем на кристалле» (SoC – *System* on *Chip*), сложность которых достигает нескольких миллионов вентилей. *ПТЦ САПР* позволяет проектировать интегральные схемы с технологическими нормами до 0,13 мкм («глубокий субмикрон»).

Программную основу комплекса составляет лицензионное прикладное программное обеспечение (ППО) фирмы Cadence Design Systems (www.cadence.*com*), которое позволяет выполнять сквозное проектирование цифровых, аналоговых и аналогово-цифровых СБИС: от системного уровня до разработки топологии. ППО функционирует на сервере и рабочих станциях фирмы *Sun* Microsystems (www.*sun*.*com*) в операционной среде *UNIX* (Solaris 8.x) (рис. 3.39).

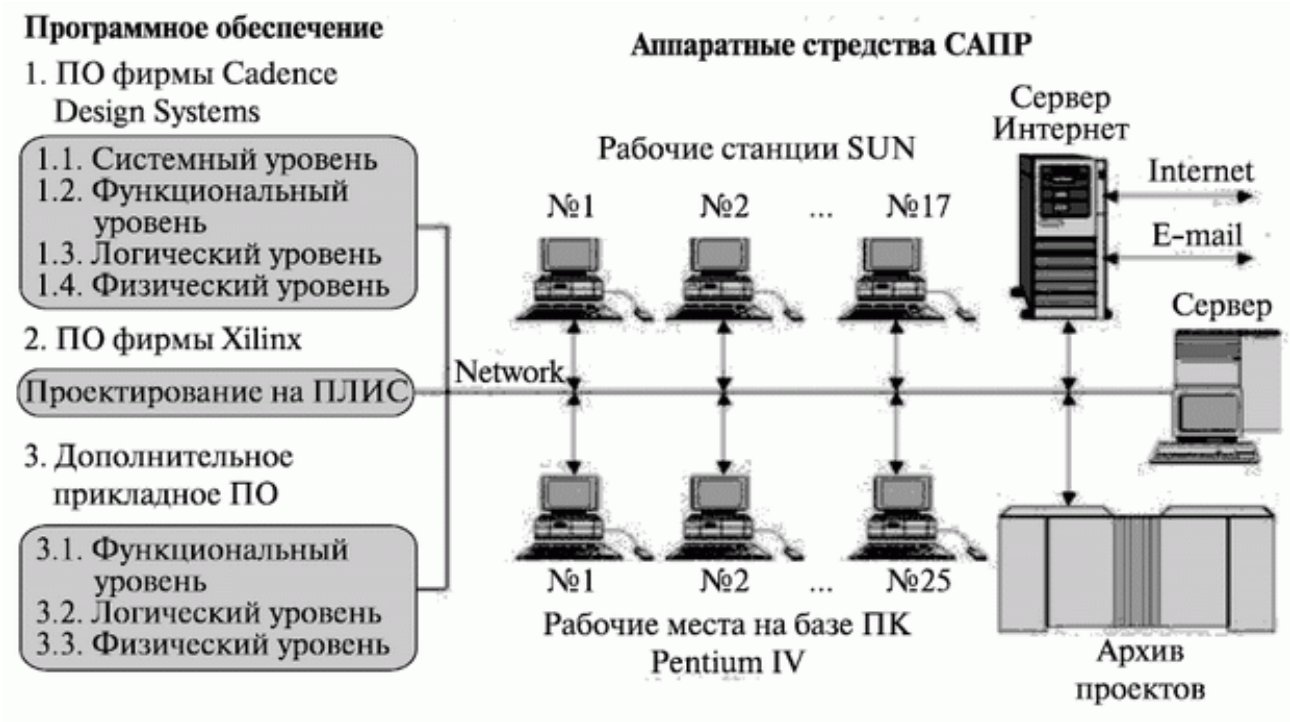


Рис. 3.39. Структура ПТЦ САПР

Наиболее часто используемые пакеты ПО – Verilog и *VHDL*, моделирование, синтез логических схем, моделирование аналоговых схем – перенесены на платформу *IBM PC* под управлением операци-

онной системы (ОС) Linux *Red Hat* (версии 8.0 и выше). На этой же платформе функционирует лицензионное ПО фирмы Xilinx, которое используется для оперативного проектирования и программирования схем на основе ПЛИС (*FPGA*). Все технические средства САПР объединены в единую сеть, которая обеспечивает безбумажную технологию проектирования и изготовления образцов микроэлектронной аппаратуры и субмикронных СБИС.

Rodnik Altium Designer – сквозная система проектирования печатных плат. В Rodnik Altium Designer работа над проектами печатных плат ведется в тесной интеграции с программированием цифровых устройств на уровне ПЛИС в единой управляющей оболочке Design Explorer.

ProgeCAD – универсальная 2D/3D САПР система с базовыми форматами DWG и DXF, обладающая пользовательским интерфейсом, сходным с *AutoCAD*. Кроме общих технических характеристик, профессиональная версия программы включает в себя и улучшенные функциональные возможности, обладающие высокими эксплуатационными
качествами.

3.9. Выбор критериев оптимальности

На разных этапах проектирования встает задача выбора наилучшего варианта из множества допустимых проектных решений, удовлетворяющих предъявленным требованиям.

Процесс принятия решения при оптимальном проектировании характеризуют следующие основные черты: наличие цели (критериев оптимальности) и альтернативных вариантов проектируемого объекта, учет существенных факторов при проектировании.

Понятие «оптимальное решение» при проектировании имеет вполне определенное толкование — лучшее в том или ином смысле проектное решение, допускаемое обстоятельствами. В подавляющем большинстве случаев одна и та же техническая задача может быть решена несколькими способами, приводящими не только к различным выходным характеристикам, схемам и конструкциям, но и к физическим принципам, положенным в основу построения объекта. При этом одно из решений может превосходить другое по одним свойствам и уступать ему по другим. В этих условиях часто чрезвычайно

трудно сказать не только, какая из систем оптимальна, но даже какая из них предпочтительнее.

Если выделяют один параметр, который характеризует свойства, он принимается за целевую функцию. При этом другие параметры подпадают под категорию ограничений. При решении однокритериальных задач применяется математический аппарат исследования операций. При создании вычислительной сети (ВС) в большинстве случаев однокритериальные задачи не удовлетворяют полученному решению. Сложные ВС характеризуются многими параметрами (емкость памяти, время счета, пропускная способность каналов и т. п.), определяющими ее качество. Значения одних параметров желательно всемерно увеличивать, а других – минимизировать.

Таким образом, ограничения и связи между отдельными параметрами ВС приводят к необходимости идти на компромисс – выбирать для каждой характеристики не максимально возможное в принципе значение, а меньшее, но такое, при котором и другие важные характеристики будут иметь приемлемые значения. Поэтому следует принимать во внимание всю совокупность характеристик ВС. Задачи проектирования, проводимые по нескольким критериям оптимизации, носят название *многокритериальных* или *задач векторной оптимизации.*

Известные методы векторной оптимизации прямо или косвенно сводят решаемые задачи к задачам скалярной оптимизации, т. е. частные критерии $F_i(X); i=\overline{1,n}$ тем или иным способом объединяются в составной критерий $F(X)=\Phi(F_1(X),\ldots,F_n(X))$, который затем максимизируется (или минимизируется). Если составной критерий отражает физическую суть ВС и вскрывает объективную связь между частными критериями и составным, то оптимальное решение считается объективным.

На практике из-за сложности составной критерий объединяет частные, что ведет к субъективности решения; такой критерий является *обобщенным,* или *интегральным.* В зависимости от того, каким образом частные критерии объединяются в обобщенный критерий, различают критерии аддитивные, мультипликативные и минимаксные (максиминные).

Если оптимизация ведется без учета статистического разброса характеристик, то соответствующий критерий оптимальности называют *детерминированным критерием;* если разброс параметров учитывается, то имеем *критерий статистический.* Последний наиболее

полно отражает качество ВС, но его использование требует больших затрат машинного времени.

Частные критерии. При проектировании по частным критериям в качестве целевой функции *F(X)* принимается наиболее важный выходной параметр проектируемой ВС, все остальные параметры в виде соответствующих условий работоспособности относятся к ограничениям. В этом случае задача оптимального проектирования является однокритериальной задачей математического программирования: максимизировать (или минимизировать) значение целевой функции *P(X)* max(min) при наличии ограничений на параметры ВС.

Из постановки задачи вытекает, что параметры, для которых выполняются ограничения в виде строгих неравенств, имеют определенный запас по сравнению с заданными техническими требованиями. Ряд параметров, для которых условия работоспособности имеют вид неравенств, запасов вообще не имеет, и любые изменения технических требований для этих параметров приводят к изменению как характеристик и структуры проектируемого объекта, так и значения целевой функции.

Частные критерии используются при проектировании ВС различного назначения.

Аддитивные критерии. В этих критериях целевая функция образуется путем сложения нормированных значений частных критериев. Частные критерии имеют различную физическую природу и в соответствии с этим — различную размерность. Поэтому при образовании обобщенного критерия следует оперировать не с «натуральными» критериями, а с их нормированными значениями. Нормированные критерии представляют собой отношение «натурального» частного критерия к некоторой нормирующей величине, измеренной в тех же единицах, что и сам критерий. При этом выбор нормирующего делителя должен быть логически обоснован. Возможны несколько подходов к выбору нормирующего делителя.

Первый подход предлагает принимать в качестве нормирующего делителя директивные значения параметров, заданные заказчиком. Логически слабым моментом такого подхода является негласное предположение того, что в ТЗ на проектируемую ВС заданы оптимальные значения параметров объекта и что совокупность заданных значений критериев рассматривается как образцовая.

Второй подход предполагает выбор в качестве нормирующих делителей максимальные значения критериев, достигаемые в области

существования проектных решений (в области компромисса). Возможен подход, при котором в качестве нормирующих делителей выбирают разность между максимальным и минимальным значениями критерия в области компромисса.

Выбор подхода к формированию безразмерной формы частных критериев носит иногда субъективный характер и должен обосновываться в каждом конкретном случае. Пусть при проектировании ВС существует n частных критериев. Тогда целевая функция задачи оптимизации в случае применения аддитивного критерия определяется как

$$F(X) = \sum_{i=1}^{n} c_i \frac{F_i(X)}{F_i^{(0)}(X)} = \sum_{i=1}^{n} c_i f_i(X),$$

где c_i – весовой коэффициент i-го частного критерия;

$F(X)$ – i-й нормирующий делитель;

$f_i(X)$ – нормированное значение i-го частного критерия.

Такая целевая функция позволяет осуществить компромисс, при котором улучшение значения одного нормированного частного критерия компенсирует ухудшение значений других.

Введение весовых коэффициентов должно учитывать различную значимость частных критериев при формировании аддитивного критерия. Определение весовых коэффициентов сталкивается с серьезными трудностями и, как правило, сводится либо к использованию формальных процедур, либо к применению экспертных оценок. С появлением обобщенного критерия исчезают логические проблемы, связанные с установлением взаимосвязей между частными критериями различной размерности и выбором наилучшего варианта ВС, и остаются лишь вычислительные трудности. Но аддитивный критерий имеет ряд недостатков, главный из которых состоит в том, что он не вытекает из объективной роли частных критериев в функционировании ВС и поэтому выступает как формальный математический прием, придающий задаче удобный для решения вид.

Другой недостаток заключается в том, что в аддитивном критерии может происходить взаимная компенсация частных критериев. Это значит, что значительное уменьшение одного из критериев вплоть до нулевого значения может быть покрыто возрастанием другого критерия. Для ослабления этого недостатка следует вводить ограничения на минимальные значения частных критериев и их весовых коэффициентов.

Несмотря на слабые стороны, обобщенный аддитивный критерий позволяет в ряде случаев успешно решать многокритериальные задачи и получать полезные результаты.

Мультипликативные критерии. Аддитивные критерии основаны на использовании принципа справедливой компенсации абсолютных значений нормированных частных критериев. Но иногда целесообразным является оперирование не с абсолютными, а с относительными изменениями значений частных критериев.

Принцип справедливой относительной компенсации формулируется следующим образом: справедливым следует считать такой компромисс, когда суммарный уровень относительного снижения значений одного или нескольких критериев не превышает суммарного уровня относительного увеличения значений других критериев. Условия оптимальности на основе принципа справедливой относительной компенсации имеют вид

$$\sum_{i=1}^{n} \frac{\Delta F_i(X)}{F_i(X)} = 0,$$

где $\Delta F_i(X)$ – приращение величины i-го критерия;

$F_i(X)$ – первоначальная величина i-го критерия.

Из этого выражения следует, что принцип справедливой относительной компенсации приводит к мультипликативному обобщенному критерию оптимальности:

$$F(X) = \prod_{i=1}^{n} F_i(X).$$

Мультипликативный критерий образуется путем простого перемножения частных критериев в том случае, если все они имеют одинаковую важность.

Достоинством мультипликативного критерия является то, что при его использовании не требуется нормировка частных критериев. Недостатки состоят в том, что критерий компенсирует недостаточную величину одного частного критерия избыточной величиной другого и имеет тенденцию сглаживать уровни частных критериев за счет их неравнозначных первоначальных значений.

Минимаксные критерии. В теории векторной оптимизации особое место занимает принцип компромисса, основанный на идее равномерности. На основе этого принципа составлены минимаксные (максиминные) критерии.

Сущность принципа максимума заключается в следующем. При создании ВС и наличии большого числа частных критериев довольно трудно, а порой и невозможно установить аналитическую зависимость между критериями. Поэтому, основываясь на идее равномерного компромисса, стараются найти такие значения переменных проектирования $X = (x_1 \ldots x_m)$, при которых нормированные значения всех частных критериев становятся равными между собой, т. е. $f_i(X) = K$, $i = 1, n$. С учетом весовых коэффициентов важности частных критериев выражения трансформируются в соотношения вида $C_i f_i(X) = K$, $i = 1,n$.

При большом числе частных критериев из-за сложных взаимосвязей иногда трудно добиться соотношений, указанных выше. Тогда применяют принцип максимина, заключающийся в такой вариации значений переменных проектирования X, при которой последовательно повышаются те нормированные критерии, численные значения которых в исходном решении оказались наименьшими. Завышение одного критерия неизбежно приводит к снижению значений части остальных критериев. Но при проведении ряда операций можно добиться определенной степени уравнивания противоречивых (конфликтных) частных критериев, что и является целью принципа максимина.

Принцип максимина формулируется следующим образом: необходимо выбрать такое X_0, на котором реализуется максимум из минимальных значений частных критериев:

$$F(X^{(0)}) = \max_X \min_i \{f_i(X)\}, \quad i = \overline{1,n}, \quad X = (x_1, \ldots, x_m).$$

Такой принцип выбора X_0 иногда носит название принципа «гарантированного результата». Он заимствован из теории игр, где, по существу, является основным принципом.

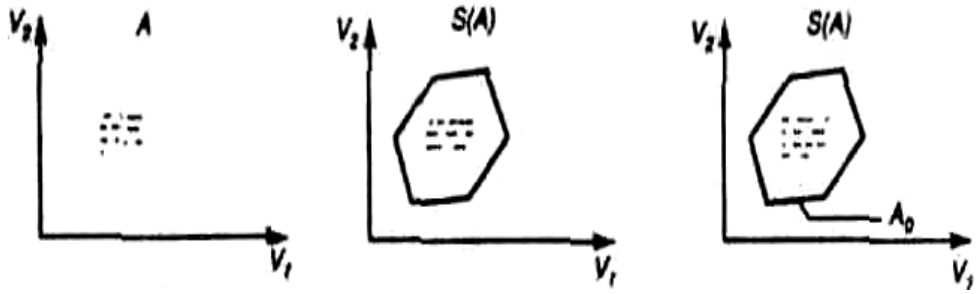
Если частные критерии $f_i(X)$ следует минимизировать, то самым «отстающим» критерием является тот, который принимает максимальное значение. В этом случае принцип равномерной компенсации формулируется в виде минимаксной задачи:

$$F(X^{(0)}) = \min_i \max_X \{f_i(X)\}, \quad i = \overline{1,n}, \quad X = (x_1, \ldots, x_m).$$

Геометрическая интерпретация принципа минимакса заключается в следующем. Пусть проектируется некоторый объект по n частным критериям $y_i = f_i(X), \quad i = \overline{1,n}$. Каждый вариант объекта пред-

ставлен в пространстве E_n в виде точки $A^{(i)}$ с координатами $A^{(i)} = (v_1^i, \ldots, v_n^{(i)})$, а множество вариантов может быть отражено в конечное множество точек $A = \{A^{(i)}, \ldots, A^{(k)}\}$, заключенное в выпуклую оболочку $S(A)$. Иными словами, область принятия решений при проектировании ограничена выпуклой оболочкой $S(A)$ в пространстве E_n (рис. 3.40).

$$c_i v_i = K, \quad c_i \geqslant 0, \quad \sum_{i=1}^{n} c_i = 1, \quad i = \overline{1, n}.$$



а *б* *в*

Рис. 3.40. Графическая интерпретация принципа минимакса:
а – множество вершин объекта; *б* – ограничение вариантов;
в – область принятия решения

Пусть все частные критерии минимизируются. Тогда областью компромисса является левая нижняя граница выпуклой оболочки $S(A)$, а решение должно находиться в области компромисса. В общем случае при неравнозначных критериях $v_i = f_i(X)$ решение на основе принципа равномерной компенсации будет соответствовать точке A_0, лежащей в области компромисса, для которой будет удовлетворяться минимаксное соотношение.

Направление, определяемое вектором $C = (c_1 \ldots c_n)$, задается в первом октанте в пространстве E_n. Произвольный вектор весовых коэффициентов C позволяет отдавать предпочтение друг перед другом частным критериям $v_i = f_i(x)$, выраженным в количественной шкале.

Выводы по выбору критериев оптимальности:

1. Выбор критерия может производиться неоднозначно. Источником сложности служит противоречивость целей (стоимость и надежность функционирования, энергоемкость и производительность,

объем ЗУ и скорость считывания всегда будут находиться в противоречии друг с другом).

2. Если требуется оптимизировать один из параметров при соблюдении ограничительных требований на остальные параметры, то формируется частный критерий $F(X)$.

3. При наличии нескольких критериев оптимальности аддитивный критерий выбирают тогда, когда существенное значение имеют абсолютные величины критериев при выбранном векторе параметров X.

4. Если существенную роль играют изменения абсолютных величин – частных критериев при вариации вектора переменных X, то применяют мультипликативный критерий оптимальности.

5. Если стоит задача достичь равенства нормированных значений конфликтных частных критериев, то оптимальное проектирование выполняют по минимаксному критерию.

Оценка значений весовых коэффициентов. В многокритериальных задачах оптимального проектирования возникает необходимость объективно оценить важность частных критериев, включаемых в аддитивный, мультипликативный или минимаксный критерий оптимальности. Важность критериев $F_i(X)$, $i=1, n$ оценивают с помощью весовых коэффициентов c_i, которые должны количественно отражать значимость соответствующих частных критериев. Значение c_i выбирается из анализа современного мирового уровня развития вычислительной техники, из требований к проектируемой ВС и из существующих возможностей реализации этих требований. Открытие новых физических принципов и разработка новых методов проектирования могут существенно повлиять на значение c_i.

Экспертные оценки. В теории экспертных оценок разработан ряд методов проведения экспертиз. Наиболее эффективными оказались методы ранжирования и приписывания баллов.

Метод ранжирования заключается в следующем. Пусть экспертиза проводится группой из l экспертов, которые являются квалифицированными специалистами в той области, где принимается решение. Метод ранжирования основан на том, что каждого эксперта просят расставить частные критерии $F_i(X)$, $i = 1, n$ проектируемой ВС в порядке их важности. При этом цифрой 1 обозначают наиболее важный частный критерий (параметр), цифрой 2 — следующий по

степени важности частный критерий и т. д. Эти ранги преобразовывают таким образом, что ранг 1 получает оценку *n,* ранг 2 – оценку (*n* – 1) и т. д. до ранга *n,* которому присваивается оценка 1, где *n* — число частных критериев.

Зная преобразованный ранг r^k *i*-го критерия у *k*-го эксперта, весовые коэффициенты c_i , $i = 1, n$ определяют из соотношения

$$c_i = \frac{\sum_{k=1}^{l} r_i^{(k)}}{\sum_{i=1}^{n}\sum_{k=1}^{l} r_i^{(k)}}, \qquad i = \overline{1, n}.$$

Метод приписывания баллов основан на том, что эксперты оценивают важность частного критерия по шкале 0—10, при этом разрешается оценивать дробными величинами или приписывать одну и ту же величину из выбранной шкалы нескольким критериям. Зная балл h_i *i*-го критерия у *k*-го эксперта, весовые коэффициенты c_i определяют из предыдущего выражения, заменив в нем r_i на

$$H_i^{(k)} = \frac{h_i^{(k)}}{\sum_{i=1}^{n} h_i^{(k)}}.$$

Последний называют весом, подсчитанным для *i*-го частного критерия $F_i(X)$ на основе оценок *k*-го эксперта. Особое место занимает обработка результатов экспертных оценок. Если рассматривать результаты оценок каждого из экспертов как реализацию некоторой случайной величины, то к ним можно применить методы математической статистики.

В общем случае при определении степени важности частного критерия $F_i(X)$ получают набор оценок r_i, $k = 1, l$, подлежащих статистической обработке. Среднее значение оценки

$$r_i = \frac{\sum_{k=1}^{l} \mu_k r_i^{(k)}}{\sum_{k=1}^{l} \mu_k}, \quad 0 \leqslant m \leqslant 1,$$

где *m* — коэффициент авторитета *k*-го эксперта.

Среднее значение оценки r_i выражает коллективное мнение группы экспертов. Степень согласованности мнений экспертов характеризуется величиной σ^2, называемой *дисперсией экспертных оценок:*

$$\sigma^2 = \frac{\sum_{k=1}^{l} (r_i^{(k)} - r_i)^2}{l},$$

Чем меньше величина дисперсии, тем с большей уверенностью можно опираться на найденное значение r_i оценки степени важности частного критерия $F_i(X)$.

Надежность экспертизы тем выше, чем меньшую долю среднего значения составляет среднеквадратический разброс оценок. По среднему значению оценки r определяются весовые коэффициенты:

$$C_i = \frac{r_i}{\sum_{i=1}^{n} r_i}, \qquad i = \overline{1, n}.$$

Статистическая обработка результатов экспертных оценок подобна статистической обработке результатов измерений. На достоверность экспертизы существенно влияют такие факторы, как численный состав экспертной группы, уровень компетентности экспертов, состав вопросов, предъявляемых экспертам, и т. д. Сейчас получили распространение интерактивные методы решения многокритериальных задач, когда информация о важности и предпочтении приходит как от инженера-разработчика, так и от ЭВМ. Уточнение обобщенных критериев и упорядочивание критериев по важности выполняется на основе диалога конструктора с ЭВМ.

Для нахождения наилучшего решения конструктору приходится решать задачи структурной и параметрической оптимизации. Тогда модель принятия решения описывается как задача многокритериальной оптимизации. В этом случае используют интерактивный режим оптимизации. На любом этапе разработчик может изменить процесс и метод решения задачи, параметры, математическое описание задачи. Проблемами здесь являются разработка эффективных пакетов прикладных программ, сценариев диалога, эвристических и точных алгоритмов с учетом неопределенности интеллектуальной деятельности инженера-разработчика.

3.10. Алгоритмы и модели проектирования технологических процессов производства электронных средств

Понятия и определения технологических процессов. *Производственный процесс* (ПП) — совокупность всех действий людей

и орудий производства, необходимых на данном предприятии для изготовления или ремонта изделий — это получение, транспортирование, контроль и хранение; подготовка производства, процесс изготовления технологической оснастки.

Производственный процесс делится на вспомогательный и основной. Основной — изготовление продукции, вспомогательный — изготовление оснастки, ремонт оборудования, энергоснабжение.

Технологический процесс (ТП) — часть производственного процесса, содержащая целенаправленные действия по изменению и последующему определению состояния предмета труда (заготовки и изделий).

Основной частью ТП является *технологическая операция* (ТО). Это законченная часть ТП, выполняемая непрерывно на одном рабочем месте (сверление и т. д.). ТО состоит из установа, технологического перехода, вспомогательного перехода и позиции.

Установ — часть ТО, выполняемая при неизменном закреплении обрабатываемых заготовок или собираемой сборочной единицы.

Переход — законченная часть ТО, выполняемая одними и теми же средствами технологического оснащения при постоянных технологических режимах и установке.

Вспомогательный переход — законченная часть ТО, состоящая из действий человека и оборудования, которые не сопровождаются изменением свойств предметов труда, но необходимы для выполнения технологического перехода (закрепление заготовки, смена инструмента и т. д.).

Рабочий ход — законченная часть технологического перехода, состоящая из однократного перемещения инструмента относительно заготовки, сопровождаемого изменением формы, размеров и качества поверхности или свойств заготовки.

Вспомогательный ход — законченная часть технологического перехода, состоящего из однократного перемещения инструмента относительно заготовки и необходимого для подготовки рабочего хода.

Позиция — фиксированное положение, занимаемое неизменно закрепленной обрабатываемой заготовкой или собираемой сборочной единицей совместно с приспособлением относительно инструмента или неподвижной части оборудования для выполнения определенной части операции. Например, обработка в поворотном приспособлении.

Выделяют три типа производства: единичное, серийное, массовое.

Единичное производство — малый объем выпуска одинаковых изделий, повторное изготовление и ремонт которых, как правило, не предусматривается.

Серийное производство характеризуется изготовлением или ремонтом изделий периодически повторяющимися партиями. В зависимости от количества изделий в партии различают мелко-, средне- и крупносерийное производство.

Массовое производство характеризуется большим объемом выпуска изделий, непрерывно изготавливаемых или ремонтируемых продолжительное время, в течение которого на большинстве рабочих мест выполняется одна рабочая операция.

Программа выпуска в массовом производстве обеспечивает возможность узкой специализации рабочих мест, за которыми закрепляется выполнение только одной операции. В этом случае оборудование располагается в полном соответствии с технологическим процессом. Если производительность и количество рабочих мест рассчитаны так, что переход с одной операции на другую осуществляется без задержек, такая организация производства называется *поточной.*

Выполнение каждой технологической операции на потоке должно осуществляться с установленным тактом и ритмом выпуска.

Такт выпуска — интервал времени, через который периодически производится выпуск изделий или заготовок определенных наименований, размеров и исполнений.

Ритм выпуска — количество изделий или заготовок определенных наименований, размеров и исполнений, выпускаемое в единицу времени.

Для радиоэлектронного производства характерно изменение серийности производства. На этапах изготовления элементов и блоков производство надо рассматривать как массовое или крупносерийное, а на этапах окончательной сборки (сборки всего изделия) — как мелкосерийное.

Порядок проектирования технологического процесса. Правильно разработанный технологический процесс обеспечивает выполнение требований, указанных в чертеже и технологических условиях, высокую производительность труда и высокие экономические показатели.

Исходными данными для проектирования технологических процессов являются чертеж детали и общие виды изделий, спецификация всех деталей, монтажные и полумонтажные схемы (для сборки), технические условия на наиболее ответственные детали, сборочные еди-

ницы и изделия, размер производственного задания, руководящие технические материалы (данные об оборудовании, нормали на инструмент, типовые ТП и др.).

Виды технологических процессов. Различают: единичный, типовой и групповой ТП.

Единичный ТП разрабатывается для изготовления или ремонта изделия одного наименования, типоразмера и исполнения независимо от типа производства.

Этапы создания ТП:

1) анализ данных и выбор аналога ТП;

2) выбор заготовки и метод ее получения;

3) определение ТО, составление технологического маршрута;

4) выбор технологического оборудования;

5) назначение и расчет режимов выполнения ТО, определение профессии и квалификации исполнителей;

6) расчет точности, производительности, эффективности ТП;

7) оформление технологической документации.

Типовой ТП характеризуется единством содержания и последовательности большинства ТО и переходов для групп изделий с общими конструктивными признаками. Типизацию начинают с классификации изделий. *Классом* называют совокупность деталей, характеризуемых общностью технологических задач. В пределах класса детали разбивают на группы, подгруппы и так далее до типа. Практически к одному типу относят детали, для которых можно составить один технологический процесс. Типовой ТП составляют на основе последних достижений науки и техники, опыта производства. При изготовлении ЭВМ применяют типовые ТП изготовления гибридных и полупроводниковых микросхем, ПП, ТЭЗ.

Групповой ТП — для совместного изготовления или ремонта групп изделий с разными конструктивными, но общими технологическими признаками. Групповые ТП используют для сборочных ТО и комплексной детали.

Виды технологических баз. Подразделяются на конструкторские, технологические и измерительные. *Конструкторская база* служит для определения положения детали или сборочной единицы в изделии; *технологическая база* — для определения положения заготовки или изделия при изготовлении; *измерительная база* — для определения относительного положения изделия или заготовки и средств измерения.

Виды контроля. Технический контроль охватывает весь технический процесс и предотвращает попадание дефектных материалов и изделий на последующие этапы изготовления или ремонта. На этапе производства устанавливают три вида контроля: входной, операционный, приемочный.

Входной контроль — это проверка соответствия материалов, заготовок, изделий, поступающих на предприятие, нормативным требованиям.

Операционный контроль — проверка деталей в процессе изготовления или ремонта, а также количественных и качественных характеристик ТП.

Приемочный контроль — проверка соответствия качества готовых изделий требованиям, установленным в нормативно-технической документации.

Сплошной контроль организуется при требовании высокого качества изделий, у которых абсолютно недопустим пропуск дефектов.

Выборочный контроль осуществляется для изделий при большой трудоемкости контроля или при контроле, связанном с разрушением изделий или с операциями, выполняемыми на автоматизированных комплексах.

Непрерывный контроль — проверка ТП при их нестабильности и необходимости постоянно обеспечивать количественные и качественные характеристики (автоматическими или полуавтоматическими методами).

Периодический контроль — проверка изделий при установившемся производстве и стабильных ТП.

Летучий контроль применяют в специальных случаях, установленных стандартами предприятия. Показателями контроля являются точность измерений, достоверность, трудоемкость и стоимость контроля.

Технологическая документация. Состав и правила выполнения технологической документации определяются Единой системой технологической документации (ЕСТД). Она представляет собой комплекс государственных стандартов и руководящих нормативных документов, устанавливающих взаимосвязанные правила и положения по порядку разработки, комплектации, оформления и обращения технологической документации, применяемой при изготовлении и ремонте изделий (контроль, испытания и перемещения).

Основное назначение ЕСТД — установление во всех организациях и на всех предприятиях единых правил выполнения, оформления,

комплектации и обращения технологической документации в зависимости от типа и характера производства.

Состав документов зависит от стадии разработки ТП, типа и характера производства. В условиях серийного и массового производства используются следующие документы: карта эскизов; технологическая инструкция; карты: маршрутная, технологического процесса, операционная, типового (группового) ТП, типовой (групповой) операции, комплектовочная, технико-нормировочная, наладки; ведомость технологических маршрутов; ведомость деталей (сборочных единиц) к типовому (групповому) ТП (операции).

Маршрутная карта (МК) является обязательным документом, предназначена для маршрутного описания технологического процесса или полного указания состава технологической операции, включая контроль и перемещения по всем операциям в технологической последовательности с указанием данных об оборудовании, технологической оснастке, материальных нормативах и трудовых затратах. Допускается взамен МК использовать соответствующую карту ТП.

Карта ТП — операционное описание ТП изготовления или ремонта изделия (составных частей) в технологической последовательности по всем операциям одного вида формообразования, обработки, сборки или ремонта с указанием переходов, технологических режимов и данных о средствах технологического оснащения, материальных и трудовых затратах.

Операционная карта имеет описание ТО с указанием переходов, режимов обработки и данных о средствах технологического оснащения, используется на рабочем месте.

Карта типового ТП — описание типового ТП изготовления или ремонта деталей и сборочных единиц, а карта типовой ТО – описание типовой ТО.

Правила оформления технологических документов приведены в ГОСТ 3.1130–93. В соответствии с этими правилами операции следует нумеровать числами ряда арифметической прогрессии (5, 10, 15 и т. д.), переходы – числами натурального ряда (1, 2, 3 и т. д.). Для обозначения позиций допускается применять римские цифры. Формы МК и правила ее оформления установлены ГОСТ 3.1118–82, а общие требования к оформлению комплектов документов на единичные процессы изложены в ГОСТ 3.1119–83. Требования к безопасности труда излагают перед описанием ТО.

Технологическая подготовка производства (ТПП). Это – совокупность мероприятий, обеспечивающих готовность предприятия к выпуску изделия (наличие всей конструкторской, технологической документации и средств технологического оснащения, необходимых для осуществления заданного объема выпуска продукции с установленными технико-экономическими показателями).

Организация и управление технологической подготовкой производства ведется в соответствии с Единой системой технологической подготовки производства (ЕСТПП), предусматривающей широкое применение прогрессивных типовых ТП, стандартной технологической оснастки и оборудования средств механизации и автоматизации производственных процессов, инженерно-технических и управленческих работ.

Основное назначение ЕСТПП заключается в решении задач обеспечения технологичности конструкции изделия, разработки ТП, проектирования и изготовления средств технологического оснащения, организации и управления технологической подготовки производства (ТПП).

Важнейшей составляющей ТПП является проектирование ТП, которое требует больших затрат времени и труда технолога (решение комплексных сложных задач, а именно: вычисления, составление текста технологической документации, вычерчивание эскизов).

Проектирование ТП традиционным методом зависит от опыта и навыка технолога, наличия справочного материала и т. д. Создание автоматизированных систем технологической подготовки производства позволяет повысить производительность и качество проектных работ, сократить сроки ТПП.

Технологичность элементов и деталей ЭВМ. Технологичность является одной из важнейших характеристик изделия. Под *технологичностью изделия* понимают совокупность свойств конструкции изделия, определяющих ее приспособленность к достижению оптимальных затрат при производстве, эксплуатации и ремонте для заданных показателей качества, объема выпуска и условий выполнения работ. Различают производственную и эксплуатационную технологичность.

Производственная технологичность конструкции изделия проявляется в сокращении затрат средств и времени на конструкторско-технологическую подготовку производства и процессы изготовления, включая контроль и испытания.

Эксплуатационная технологичность заключается в сокращении затрат времени и средств на техническое обслуживание и ремонт изделия. Технологичность конструкции оценивается качественно и количественно.

Качественная оценка характеризует технологичность конструкции обобщенно на основании опыта исполнителя. Такая оценка выполняется на стадии проектирования, когда выбирается лучшее конструктивное решение и не требуется определения степени технологичности сравниваемых вариантов. Качественная оценка предшествует количественной.

Количественная оценка осуществляется с помощью системы базовых показателей (трудоемкость изготовления изделия, технологическая себестоимость). Базовыми показателями технологичности производства электронных устройств служат коэффициенты: использования микросхем и микросборок в блоке, автоматизации и механизации монтажа, автоматизации и механизации подготовки электрорадиоэлементов, автоматизации и механизации операций контроля и настройки, повторяемости и применяемости электрорадиоэлементов, прогрессивности формообразования деталей.

При анализе результатов следует учитывать сложность изделия и уровень основного производства. Конструкция детали должна отвечать следующим требованиям: состоять из стандартных и унифицированных деталей; изготовляться из стандартных заготовок; иметь оптимальные точность и шероховатость поверхностей; обеспечивать применение стандартных и типовых процессов ее изготовления, одновременного изготовления нескольких деталей и прогрессивных процессов формообразования: литья под давлением, литья по выплавляемым моделям, прессования пластмасс, металлокерамики, холодной штамповки.

Контрольные вопросы

1. Сформулировать задачу компоновки.
2. Основные критерии задачи размещения.
3. Основные критерии задачи трассировки.

4. В чем разница волнового и лучевого алгоритмов трассировки?

5. Что означает трассировка?

6. Что является критерием трассировки?

7. На какие группы делится трассировка?

8. Дайте характеристику трассировки проводных соединений.

9. На чем основаны алгоритмы трассировки проводных соединений?

10. Какие ограничения необходимо учитывать при трассировке проводных соединений?

11. Как работает алгоритм трассировки Прима?

12. Дайте характеристику печатного и пленочного монтажа.

13. Что относится к метрическим параметрам схемы?

14. Что относят к топологическим параметрам схемы?

15. Что называют минимальным связывающим деревом?

16. Как строится дискретное рабочее поле (ДРП)?

17. Как работает волновый алгортм Ли?

18. Как осуществляется трассировка по магистралям?

19. Что называют магистралями 1-го и 2-го уровней?

20. Дать характеристику минимаксного и максиминного критериев.

Задачи для самостоятельного решения

Задача 1. Дан граф G = (X, U) с X = {a,b,c,d,e,f} и U = {ad,ae,af,bd,be,cd,ce,cf}. Начертить этот граф на плоскости так, чтобы его ребра изображались прямолинейными отрезками и не пересекались друг с другом.

Задача 2. Доказать, что изоморфизм графов представляет собой отношение эквивалентности, т.е. удовлетворяет трем условиям: $L \cong L'$ (рефлексивность), если $L' \cong L$ (симметрия); если $L \cong L'$ и $L' \cong L''$, то $L = L''$ (транзитивность). Отсюда следует, что всякое множество графов однозначно разбивается на попарно непересекающиеся классы изоморфных.

Задача 3. Изобразить графы, представленные следующими матрицами смежности:

01100	01110
10000	10001

а) 10011 б) 10010
00100 10101
00100 01010

Задача 4. Пусть Г! Г = (X,E) – граф и IXI = n. Какое может быть максимально возможное значение IEI?

Задача 5. Сколько существует различных графов, имеющих *n* вершин?

Задача 6. Пусть G = (V,E), где V = $\{v_1,v_2,v_3,v_4\}$ и E = $\{|v_1,v_2|, |v_1,v_3|, |v_2,v_4|, |v_3, v_4|\}$.

Используя матрицу смежности, определить:

а) число маршрутов длины 2 из v_3 в v_2.

б) число маршрутов длины 3 из v_1 в v_2.

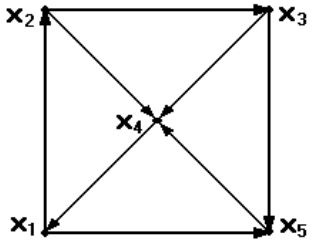
в) является ли граф связным.

Задача 7. Дать матричную характеристику ацикличности в графе.

Задача 8. Дан граф G = C$\{v_1,v_2,v_3,v_4\}$, $\{[v_1,v_2], [v_1,v_3], [v_1, v_4], [v_2,v_3], [v_2,v_4], [v_3,v_4]\}$. Представить изображение графа и планарный ему граф (карту).

Задача 9. Дать определение орграфа.

Задача 10. Представить орграф системой линейных алгебраических уравнений:

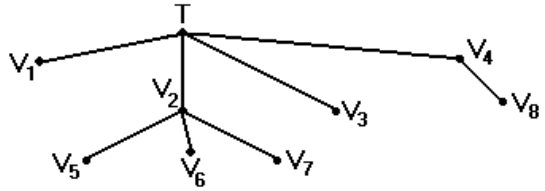


Задача 11. Пусть *T* – ориентированное дерево. Разрезом *C* дерева *T* называется подмножество вершин *T*, таких, что:

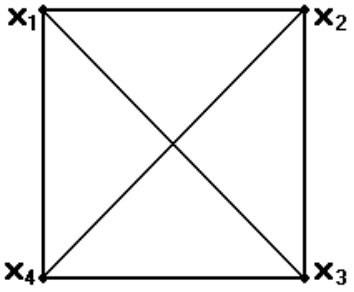
а) не существует двух вершин *C* в дереве *T* на маршруте;

б) ни одна вершина не может быть добавлена к *C* без нарушения *а*.

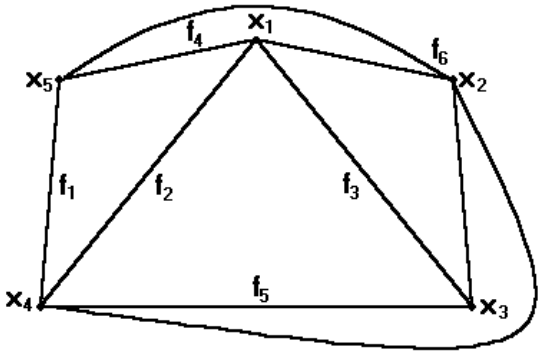
Определить все разрезы ориентированного дерева, изображенного на рисунке:



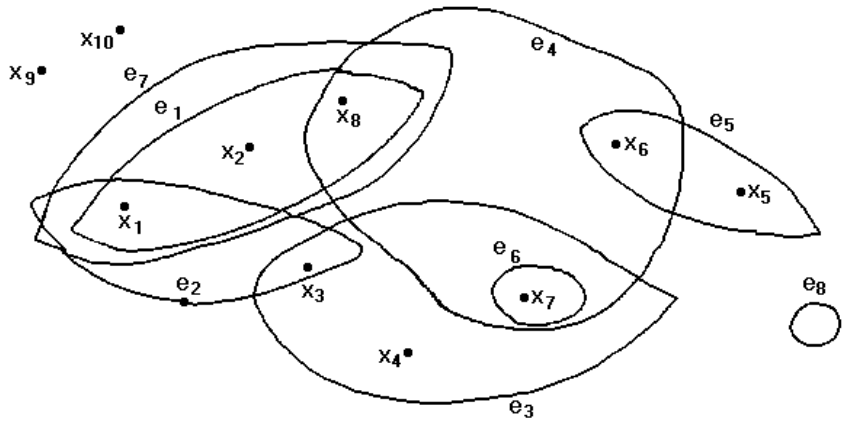
Задача 12. Определить все покрывающие деревья для графа:



Задача 13. Для графа представить планарный изоморфный граф:



Задача 14. Представить матрицу инциденций для гиперграфа:



Задания для самостоятельной работы

Алгоритмы размещения

I. *Последовательно-итерационный алгоритм размещения вершин графа на плоскости с минимизацией СДС (суммарная длина соединений).*

Задан граф $G(X, U)$, отображающий схему и коммутационное монтажное пространство, на котором размещен G_r (дискретная сетка посадочных мест элементов на типовой плате). Необходимо размес-

тить все вершины $x_i \in X$ в узлы сетки с минимизацией суммарной длины ребер $u_i \in U$. Часть элементов размещена вручную (если нет, то первый элемент закрепляется произвольно). Разместим вершину x_i в (j+1)-ю позицию, где j – число занятых позиций. Введем понятие коэффициента связности $\Delta(x_i) = a_{i,h} - a_{i,z}$, где $a_{i,h}$ – число ребер, соединяющих вершину x_i с подмножеством ранее размещенных вершин графа, $a_{i,z} = \rho(x_i) - a_{i,h}$ – число ребер, соединяющих вершину x_i с неразмещенными вершинами графа. Тогда $\Delta(x_i) = 2a_{i,h} - \rho(x_i)$, где $\rho(x_i)$ – степень вершины x_i.

Идея алгоритма заключается в определении коэффициента связности для всех неразмещенных вершин и в помещении в первую свободную позицию рядом с размещенной вершиной вершины с максимальным значением $\Delta(x_i)$. Последовательно рассматривая все вершины графа, выполняют их размещение.

Алгоритм можно использовать при размещении разногабаритных элементов, но с кратными размерами. В этом случае КП покрывается масштабной сеткой, линейные размеры дискретов которой равны минимальным размерам элементов. При размещении элемента, кратного нескольким дискретам, в список занятых позиций заносят все те элементарные позиции, которые он занимает.

Итерационная часть алгоритма заключается в парной перестановке некоторых вершин графа $G(X, U)$ и в оценке СДС при каждой замене. Замена целесообразна, если СДС при этом уменьшается. Среди всех вершин графа выбираются претенденты для замены. Для этого просматривается матрица расстояний D, отображающая расстояния между элементами как результат размещения вершин графа схемы $G(X,U)$ на дискретном поле последовательным алгоритмом. Множество инцидентных ребер U_i для каждой вершины $x_i \in X$ разбивается на два непересекающихся подмножества U_i' и U_i''. К подмножеству U_i' (α-ребра) относятся «короткие» ребра, вес которых не превышает некоторого значения α, равного 2-3, вес ребра отражает расстояние между связываемыми им вершинами. К подмножеству U_i'' (β-ребра) относятся остальные ребра («длинные» ребра). Для каждой вершины подсчитывается разность $\Delta_i = |U_i''| - |U_i'|$.

Поиск вершин, которые выступают претендентами на замену, осуществляется среди тех вершин, для которых $\Delta_i > 0$. Первым претендентом (вершина x_i) является вершина, имеющая наибольшее положительное значение Δ_i.

Для поиска второй вершины (вершина x_j) – претендента на замену с вершиной x_i – множество всех вершин X разбивается на два подмножества X_1 (включаются, кроме вершины x_i, все отстоящие от нее на расстояние, не большее α) и X_2 (все остальные вершины). Вторая вершина выбирается среди тех вершин подмножества X_2, которые инцидентны β-ребрам вершины x_i. Для каждой такой вершины подсчитывается число ребер U_J'', инцидентных вершинам подмножества X_1. Для каждой вершины подсчитывается разность: $\sigma_J = |U_J''| - |U_J'|$. Претендентом на замену является вершина x_j, имеющая наибольшее значение σ_j.

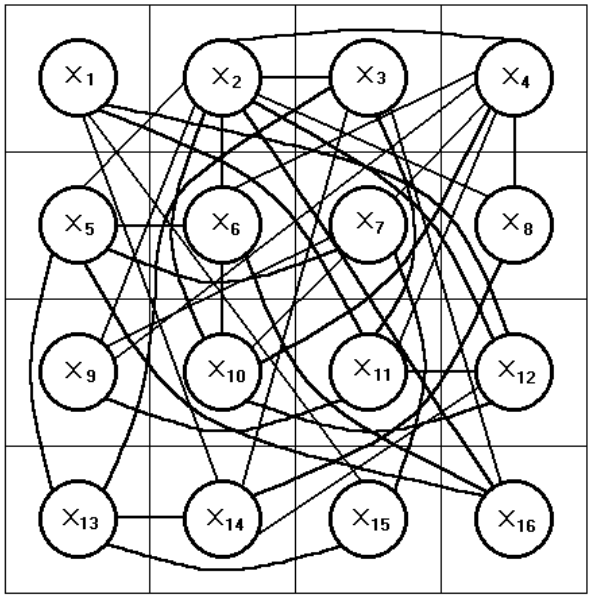
Перестановка вершин x_i и x_j приводит к уменьшению числа β-ребер и увеличению числа α-ребер, т.е. к уменьшению суммарной длины ребер графа.

Если невозможно выполнить перестановку вершин, уменьшающую СДС, то производят перестановку групп вершин. Для этого производится факторизация графа G: разбивая все подмножество вершин на классы $x_1, x_2, \ldots, x_n$, строим новый граф (мультиграф), в котором вершинами являются классы X_i, а ребрами – связи между классами. Далее производится перестановка групп элементов (классов), каждая из которых принимается за вершину мультиграфа.

Пример, поясняющий итерационную часть алгоритма

Заданы граф схемы $G(X, U)$ с начальным размещением и матрица расстояний D, полученная путем перемножения матриц смежности графа КП G_r и графа G.

Исходный граф выглядит следующим образом:



Результаты подсчета разности длин инцидентных ребер приведены в следующей таблице:

Номер вершины	U_J'	U_J''	σ_J
1	0	4	4
2	7	2	-5
3	3	3	0
4	5	2	-3
5	4	1	-3
6	4	1	-3
7	5	0	-5
8	3	1	-2

Номер вершины	U_J''	U_J'	σ_J
9	3	1	-2
10	4	1	-3
11	5	1	-4
12	3	2	-1
13	3	1	-2
14	2	3	-1
15	3	1	-2
16	0	4	-4

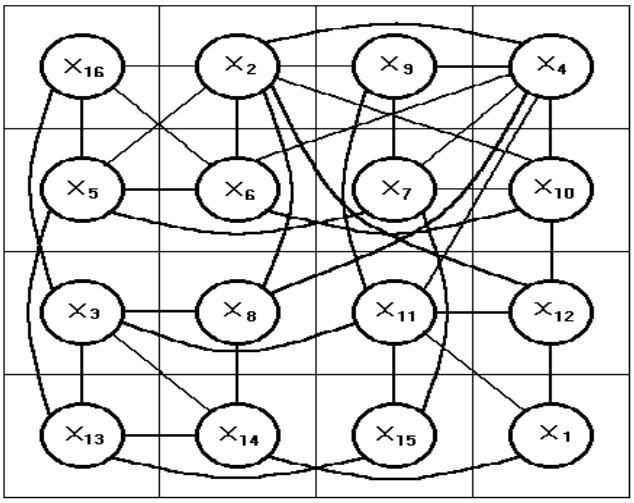
Первой выбираем вершину x_i с максимальным положительным значением $\Delta_i = 4$, для которого подсчитаем разность длин ребер преобразованного графа σ_j, определив область целесообразных ее перестановок.

Выбираем вершину x_{16}, имеющую максимальное значение $\sigma_i = 4$, и производим перестановку вершин x_1 и x_{16}.

Разность длин ребер преобразованного графа:

Номер вершины	U_J'	U_J''	σ_J
11	4	2	2
12	3	2	1
14	3	2	1
15	3	1	2
16	4	0	4

Далее выполняем одиночные перестановки вершин до тех пор, пока выполнится условие $\Delta_i + \sigma_i > 0$. Получаем размещение, в котором суммарная длина ребер сократилась от 114 до 74:



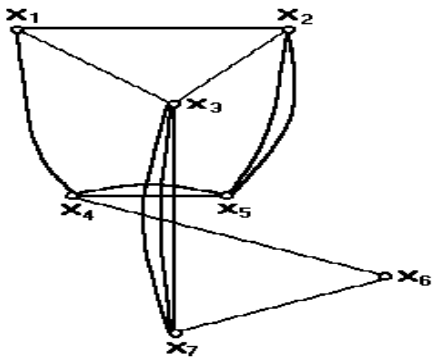
II. *Алгоритм последовательного размещения элементов (модулей) в линейку.*

Заданы граф $G(X, U)$ и матрица расстояний D, элемент которой d_{ij} равен числу связей между вершинами x_i и x_j. Определяется суммарное значение связей каждого элемента с остальными $\Sigma\Sigma$, и на вакантное установочное место p_i назначается элемент, имеющий минимальное значение $\Sigma\Sigma$. На следующее вакантное место из оставшихся претендует элемент, имеющий минимальное значение $\Sigma\Sigma$:

R=

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	∑
1	0	0	0	0	0	0	0	0	0	0	4	5	0	4	5	0	18
2	0	0	1	2	2	1	0	1	3	2	0	4	0	0	0	5	18
3	0	1	0	0	0	0	0	2	0	0	2	0	5	4	0	4	18
4	0	2	0	0	0	3	2	1	5	4	3	0	0	0	0	0	20
5	0	2	0	0	0	1	2	0	0	0	0	0	2	0	0	5	12
6	0	1	0	3	1	0	0	0	0	1	0	0	0	0	0	4	10
7	0	0	0	2	2	0	0	0	3	2	0	0	0	0	2	0	11
8	0	3	2	1	0	0	0	0	0	0	0	0	0	4	0	0	10
9	0	3	0	5	0	0	3	0	0	0	2	0	0	0	0	0	13
10	0	2	0	4	0	1	2	0	0	0	0	2	0	0	0	0	11
11	4	0	2	3	0	0	0	0	2	0	0	1	0	0	1	0	13
12	5	4	0	0	0	0	0	0	0	2	1	0	0	3	0	0	15
13	0	0	5	0	2	0	0	0	0	0	0	0	0	1	2	0	10
14	4	0	4	0	0	0	0	4	0	0	0	3	1	0	0	0	16
15	5	0	0	0	0	0	2	0	0	0	1	0	2	0	0	0	10
16	0	5	4	0	5	4	0	0	0	0	0	0	0	0	0	0	18

Задание. Используя итерационный алгоритм, выполнить компоновку для данного графа:



4. ТЕХНИЧЕСКОЕ, ПРОГРАММНОЕ И ИНТЕЛЛЕКТУАЛЬНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ЭЛЕКТРОННЫХ СРЕДСТВ

В этом разделе рассмотрены технические средства систем автоматизированного проектирования (САПР) электронных средств. Описаны базы данных, этапы их проектирования, модели, системы управления базами данных и средства обработки. Дан анализ современных требований к системам проектирования, предложена архитектура интеллектуальных САПР. Представлены количественные и качественные характеристики интеллектуальных систем проектирования, различные методы структурного и параметрического синтеза.

4.1. Характеристика технического обеспечения САПР. Технические средства машинной графики. Специализированные сопроцессоры. Вычислительные сети САПР

Техническое обеспечение (ТхО) САПР совместно с программным обеспечением (ПО) является инструментальной базой САПР, в среде которой реализуются другие виды обеспечения САПР. Компоненты ТО САПР включают в себя средства вычислительной и организационной техники, средства передачи данных, измерительные и другие устройства или их сочетания, обеспечивающие функционирование САПР. Совокупность компонентов ТО образует комплекс технических средств (КТС) (рис. 4.1). Требования к составу и структуре КТС формируются из: общих требований к структуре САПР; эффективного решения выделенного класса задач проектирования; активного включения пользователя в процесс проектирования; возможности работы с графическим материалом.

КТС САПР объединены в группы взаимодействия оборудования. Группа *базовой конфигурации* — это максимальный состав, позволяющий решать задачи определенного класса. Основным компонентом базовой конфигурации является АРМ.

Режимы работы зависят от задач, решаемых в САПР. *По характеру вычислительного процесса* задачи решаются без участия пользователя и с его участием.

С учетом сложностей вычисления задачи требуется несколько секунд, минут и время, значительно большее, чем время диалога.

По объему информации задачи используют основную память ЭВМ и частично внешнюю память ЭВМ. Исходя из этой классификации, выделяют необходимые режимы работы технических средств:

1) однопрограммный, когда доступны все ресурсы ЭВМ;

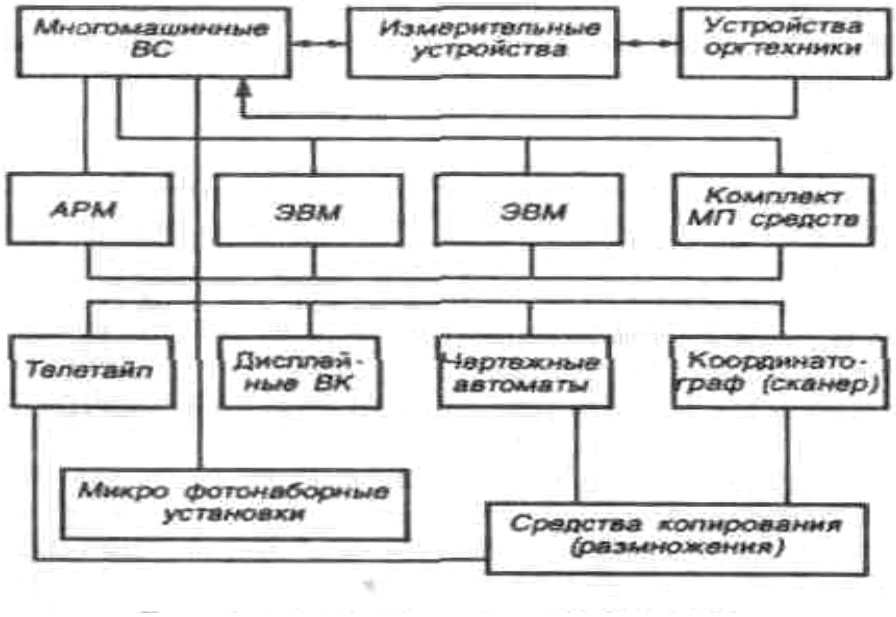


Рис. 4.1. Классификация КТС САПР

2) мультипрограммный с фиксированным количеством задач; при таком режиме оперативная память ЭВМ делится на фиксированное число разделов, которые определены для выполнения одной задачи в каждом;

3) мультипрограммный режим с переменным числом задач, все ресурсы ЭВМ – общие.

Режимы работы КТС классифицируют по удаленности проектировщика от основного компонента технических средств:

– местный режим – пользователь непосредственно у ЭВМ;

– дистанционный режим, при котором часть периферийного оборудования связана с процессором канала связи.

Режимы КТС классифицируются по степени участия пользователей в процессе решения задачи:

• пакетный режим – пользователь готовит задание в составе пакета, обработка задач производится по очереди; после решения поль-

зователю требуется проанализировать результаты и подготовить новый вариант решения, что замедляет отладку и увеличивает время получения окончательного результата;

• режим разделения времени (РРВ) — каждой решаемой задаче выделяется поочередно определенный квант времени работы процессора; пользователь за терминалом составляет программу, транслирует, редактирует и выполняет расчеты на ней.

От выбора режима использования КТС САПР зависит эффективность эксплуатации технических средств. Поэтому при создании САПР определенного уровня ЭВМ, комплекса, системы или сети необходимо провести четкий анализ решаемых задач.

Пакетный режим предназначен для задач с большим временем счета и задач, не требующих вмешательства пользователя в процесс решения.

Режим разделения времени удобен для задач, время счета у которых соизмеримо с временем отклика пользователя на запрос ЭВМ, а также когда необходимо вмешательство пользователя в процесс решения.

Технические средства машинной графики. Технические средства машинной графики классифицируются по следующим признакам: назначению, степени автоматизации, методу обработки информации, способу отсчета текущих координат (рис. 4.2). Устройства ввода графической информации (УВГИ) повышают производительность труда и достоверность кодируемой информации. В полуавтоматических устройствах кодирование информации происходит при участии человека, который, перемещая регистрирующий орган, обходит последовательно все элементы графического документа.

Перемещение регистрирующего органа фиксируется. Применяются дополнительные панели ввода графической информации (клавиатура, сканер, мышь) — устройства различных типов, например: ПАСГИ, ЭМ-709, ГАРНИ, Benson, Ca1comp и др.

Полуавтоматические устройства ввода графической информации сопрягаются с ЭВМ, выполняющими контроль ошибок, допущенных оператором, позволяют выбирать фрагменты из библиотеки элементов, ранее введенных участков изображения и осуществляют оперативный вывод закодированной информации. Такие операции выполняют системы QED и IED фирмы Quest Automation и системы Anistigid фирмы Anisto.

Автоматические устройства ввода графической информации характеризуются высоким быстродействием и отсутствием несистематических ошибок, но для их работы требуются графические документы высокого класса и точности.

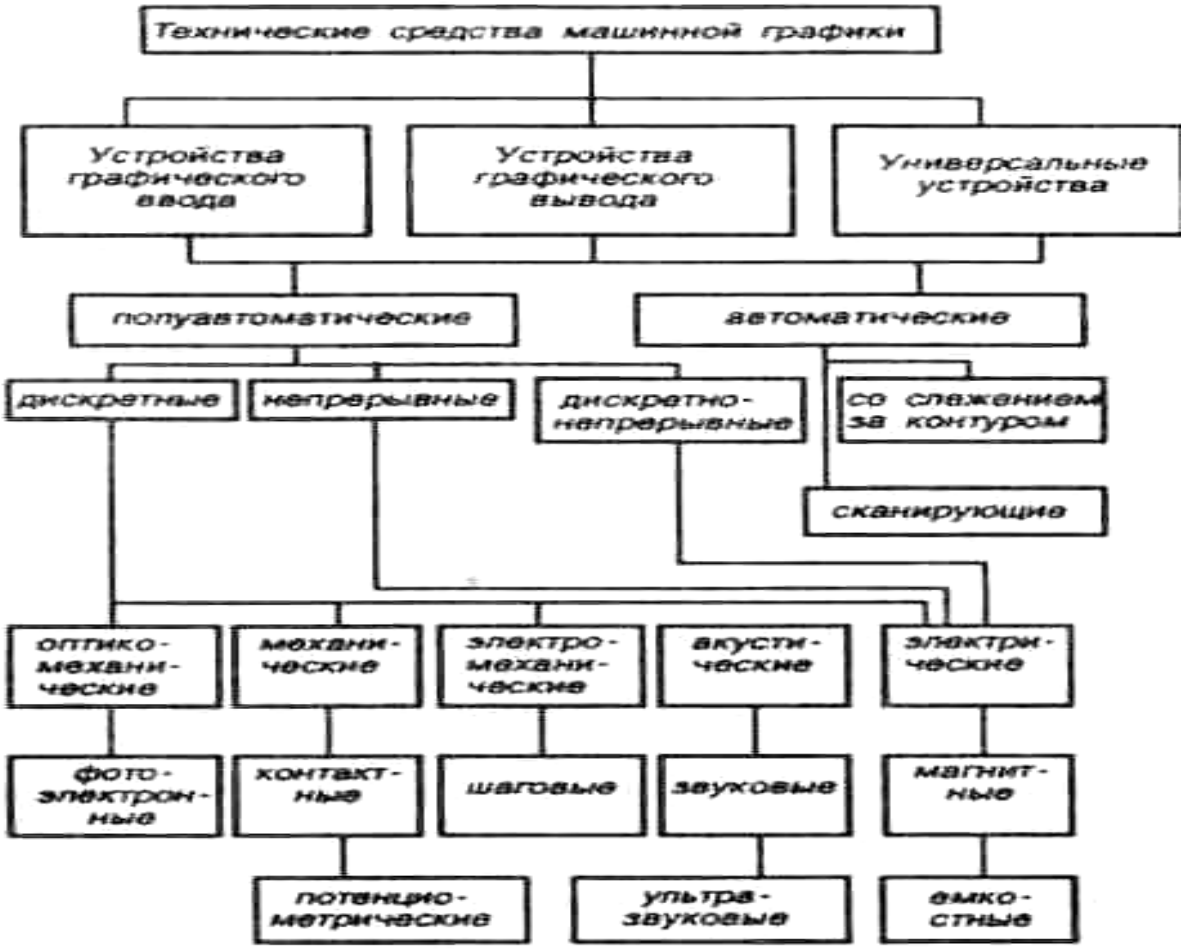


Рис. 4.2. Технические средства машинной графики

По методу считывания различают устройства непрерывного, дискретного и непрерывно-дискретного действия. Для графической информации большого объема желательно использовать дискретные устройства, обладающие более жесткими требованиями к точности. В зависимости от способа отсчета текущих координат УВГИ разделяются на электромеханические, оптико-механические, магнитные, контактные и звуковые.

Сканирующие устройства служат для преобразования глиняной модели в эквивалентную математическую модель, которая формируется в ЭВМ в результате съемки параметров глиняной модели. Щуп сканирующего устройства, перемещаемый по поверхности глиняной

модели, представляет собой светочувствительный элемент, который замыкается при касании поверхности модели. Для каждой точки соприкосновения происходит автоматическая запись ее координат, рассчитанных относительно некоторой базовой точки. Таким образом, из совокупности тысяч дискретных точек глиняной модели сканирующее устройство формирует трехмерный образ, например, корпуса автомобиля. Результаты съемки размеров модели поступают в мини-компьютер, соединенный с устройством сканирования, которое способно записывать трехмерные координаты со скоростью 4 точки в секунду, автоматически увеличивая плотность измерений при съемке критических областей.

После введения данных в компьютер они могут выводиться на экран дисплея и подвергаться дальнейшей обработке. Таким же образом можно получить комбинацию из нескольких наиболее удачных конструктивных решений. Щуп устройства сканирования можно заменить на фрезу, с помощью которой автоматически формируется вторая половина модели по данным, воспроизводимым при сканировании.

Представителями группы устройств вывода графической информации являются чертежные автоматы, позволяющие получить документацию в виде чертежей, графиков, схем, диаграмм. Классификацию чертежных автоматов можно провести по следующим признакам: способу программного управления; методу обработки данных; принципу действия исполнительного механизма.

Чертежные автоматы работают как в автономном режиме от машинных носителей информации, так и непосредственно от ЭВМ. К их недостаткам относятся малое быстродействие и низкая надежность. Чертежные автоматы входят в комплекс периферийных устройств со своей малой ЭВМ, которая сопрягается с вычислительными устройствами.

Чертежные автоматы типа «Итекан» различны по назначению, программному управлению и конструктивному исполнению. Изображение вычерчивается цветными чернилами, тушью или карандашом на обычной бумаге или кальке линиями различной толщины. Размеры рабочего поля – 818×40000 мм, скорость вычерчивания – 40 мм/с.

К устройствам графического вывода информации относятся программно-управляемые координатографы, предназначенные для изготовления прецизионных фотооригиналов печатных плат, полос-

ковых, микрополосковых линий. От чертежных автоматов координатографы отличаются повышенной точностью и способами нанесения изображения (экспонирование, вырезание, гравирование, скрайбирование и др.).

Программно-управляемые координатографы отличаются большими размерами рабочего поля. Управление осуществляется через каналы связи или с помощью промежуточных носителей информации.

Координатограф КПА-1200 предназначен для изготовления фотошаблонов микросхем и печатных плат. В состав координатографа входят: фотосчитыватель; пульт управления; координатный стол с размерами рабочего поля 1200×1200 мм; устройство управления с блоками ввода информации; операционное устройство; интерполятор; блок задания скоростей; блок обработки информации; блок ориентации инструмента; блок технологических операций; блок управления приводом; цифровая индикация; блок центрального управления. Скорость перемещения – от 25 до 90 мм в секунду. В настоящее время для данных целей используется лазерный генератор.

Наиболее перспективными устройствами ввода-вывода графической информации остаются дисплеи. Скорость обмена информацией этих устройств сравнима со скоростью обработки информации в ЭВМ.

Дисплеи классифицируются по таким признакам, как:

– назначение и вид выводимой информации;

– метод формирования изображения;

– физические принципы создания информации.

Дисплеи служат для обработки символьной информации (высокая надежность и экономичность).

Сканеры осуществляют ввод изображения по контуру рисунка. Дисплеи позволяют выводить на экран графические изображения, имеют широкий набор встроенных функций преобразования информации: сдвиг, масштабирование, поворот, редактирование и мультиплексирование.

Специализированные сопроцессоры. Для решения задач САПР применяют специализированные ЭВМ. Их создание оправдано, если достигаемый с их помощью экономический эффект превышает затраты на разработку и эксплуатацию.

Целью создания специализированных ЭВМ является повышение производительности труда проектировщика. Основная идея заключается в аппаратной реализации типовых методов решения задач проектирования (замена программ схемами). Частые обращения к программам приводят к большим затратам машинного времени, а при замене схемами имеют большой выигрыш во времени. Другой путь повышения производительности связан с применением параллельной обработки информации.

Способы реализации специализированных ЭВМ:

1) расширение системы команд универсальных ЭВМ путем включения команд вычисления часто встречающихся функций;

2) разработка периферийных процессоров (приставок), подключаемых к универсальной ЭВМ для реализации специализированных вычислительных процедур (например, трассировки печатной платы) или операций;

3) создание специализированных ЭВМ или сопроцессора, структура которого ориентирована на решение узкого класса задач большой сложности.

По функциональному назначению специализированные ЭВМ и процессоры разделяются на группы:

• для решения частных простых задач проектирования;

• для выполнения отдельных проектных процедур;

• для обеспечения удобного взаимодействия с проектировщиком (например, графический процессор);

• для обеспечения отладки задач проектирования (эмулятор системы команд ЭВМ для отладки программ на уровне АРМ);

• для решения задач структурной организации КТС САПР (ЭВМ-СУБД, ЭВМ-шлюзы для связи локальных и глобальных ВС).

Речевые устройства для оперативной связи проектировщика. Средства речевого ввода недостаточно совершенны, но постоянно модернизируются. *Активный речевой терминал* применяется для оперативного сообщения об аварийных ситуациях, а также для указания пароля при входе в систему. *Пассивный речевой терминал* выделяет из информационного потока адресованные им речевые команды и данные, не мешая обычной работе за дисплеем. Включается синтезатор речи, как правило, между дисплеем и центральным процессором. Для генерации фразы он использует специальный словарь (несколько сот слов или фраз). Чтобы устройство не мешало работе других проектировщиков, предусмотрен головной телефон (наушники).

Вычислительные сети САПР. Локальную вычислительную сеть (ЛВС) объединяют в единую информационную систему АРМ, ЭВМ, графопостроители, терминальные станции и другую специализированную аппаратуру. ЛВС имеют открытую архитектуру, обеспечивающую возможность подключения к сети любых других ЛВС. Основное достоинство — низкая стоимость системы передачи данных.

ЛВС САПР обеспечивают:

1) использование режимов пакетной и диалоговой обработки, разделения времени, виртуальной памяти;

2) экономичную обработку информации по принципу: «наиболее важные процессы САПР выполняются КТС с высокой производительностью, меньше всего – на дешевых ЭВМ»;

3) высокую надежность и достоверность функционирования, высокую производительность;

4) применение разнообразного проблемно-ориентированного ПО, централизованных и локальных БД;

5) работу с АРМ различного назначения;

6) централизованную и децентрализованную обработку информации.

Использование ЛВС позволяет создать САПР нового поколения, объединяющие контрольно-измерительные комплексы и места сбора информации с АРМ конструкторов, механиков и т. д. Основное назначение ЛВС — распределение ресурсов ЭВМ (программ, периферийных устройств, терминалов, памяти) для эффективного решения задач САПР. ЛВС должна иметь надежную, быструю и дешевую систему передачи данных (СПД). Для достижения этого ЛВС выполняется на основе следующих принципов:

1. *Принцип единых протоколов.*

Протоколы межмашинной связи в ЛВС предназначены для организации обмена информацией между компонентами сети. Протоколы сети определяют форму сообщения или пакета сообщений (длина, заголовок, знак окончания, дополнительная информация для повышения достоверности передачи информации и др.). Все процедуры управления и соответствующие им протоколы едины для всей сети и не зависят ни от типа ЭВМ, ни от происходящих в них процессов.

2. *Принцип единой передающей среды.*

При построении СПД для ЛВС используют активную или пассивную структуру передающей среды. Активная структура выполня-

ется на основе распределенных усилителей и преобразователей, обеспечивающих передачу информации в параллельном и последовательном кодах. Пассивная структура выполняется на основе пассивного носителя (коаксиального либо плоского кабеля) и использует преобразователи-усилители одного типа, обеспечивающие возможность работы либо в параллельном, либо в последовательном коде.

Структура передающей среды реализована с применением либо моноканала, либо многопроводной связи. Более дешевой является структура с моноканалами, так как снижаются издержки на эксплуатацию и прокладку соединений. Моноканалами являются физическая среда, аппаратные и программные средства, предназначенные для параллельной передачи одновременно всем абонентским системам. Моноканал предназначен для коллективного использования большим числом абонентских систем (с высокой пропускной способностью передачи информации). Физическая среда моноканала реализуется посредством волоконно-оптических линий связи, коаксиальных или плоских кабелей, скрученных пар проводов и т. д.

3. *Принцип единого метода управления.*

Протоколы ЛВС применяют централизованные и децентрализованные формы управления одноузловой структурой моноканала. Принцип единого метода управления проявляется в выборе одной из этих форм, обеспечивающей достаточную надежность работы СПД и максимальную загрузку каналов связи. При этом для определения метода управления следует учитывать структуру соединений, их длину, число абонентов и сложность обработки информации с помощью ресурсов ЛВС.

Для централизованных форм управления характерны обилие служебной информации и приоритетность подключаемых к моноканалу станций. Защита от конфликтов в моноканале реализуется центральной управляющей вычислительной машиной. В децентрализованных формах управления, допускающих одинаковый приоритет всех станций, подключаемых к моноканалу, применяют многоступенчатые тракты защиты от конфликтов, учитывающие противоречивые требования надежности и максимальной загрузки моноканала. При использовании в ЛВС нескольких методов управления средой передачи данных существенно возрастает сложность схемных решений контроллеров, с помощью которых станции ЛВС подключаются к СПД.

4. *Принцип информационной и программной совместимости.*

Данный принцип предусматривает совместимость операционных систем, программ и систем управления базами данных, рассредоточенных в рамках ЛВС (возможность адаптации процессов к пересылаемой информации и применения единых систем кодирования и контроля информации).

5. *Принцип гибкой модульной организации.*

Предусматривается проектирование СПД ЛВС на основе набора гибких конструктивно законченных модулей.

ЛВС классифицируются (рис. 4.3):

– по *топологическим признакам* — иерархической, кольцевой и звездообразной конфигурации, конфигурации типа «общая шина»;

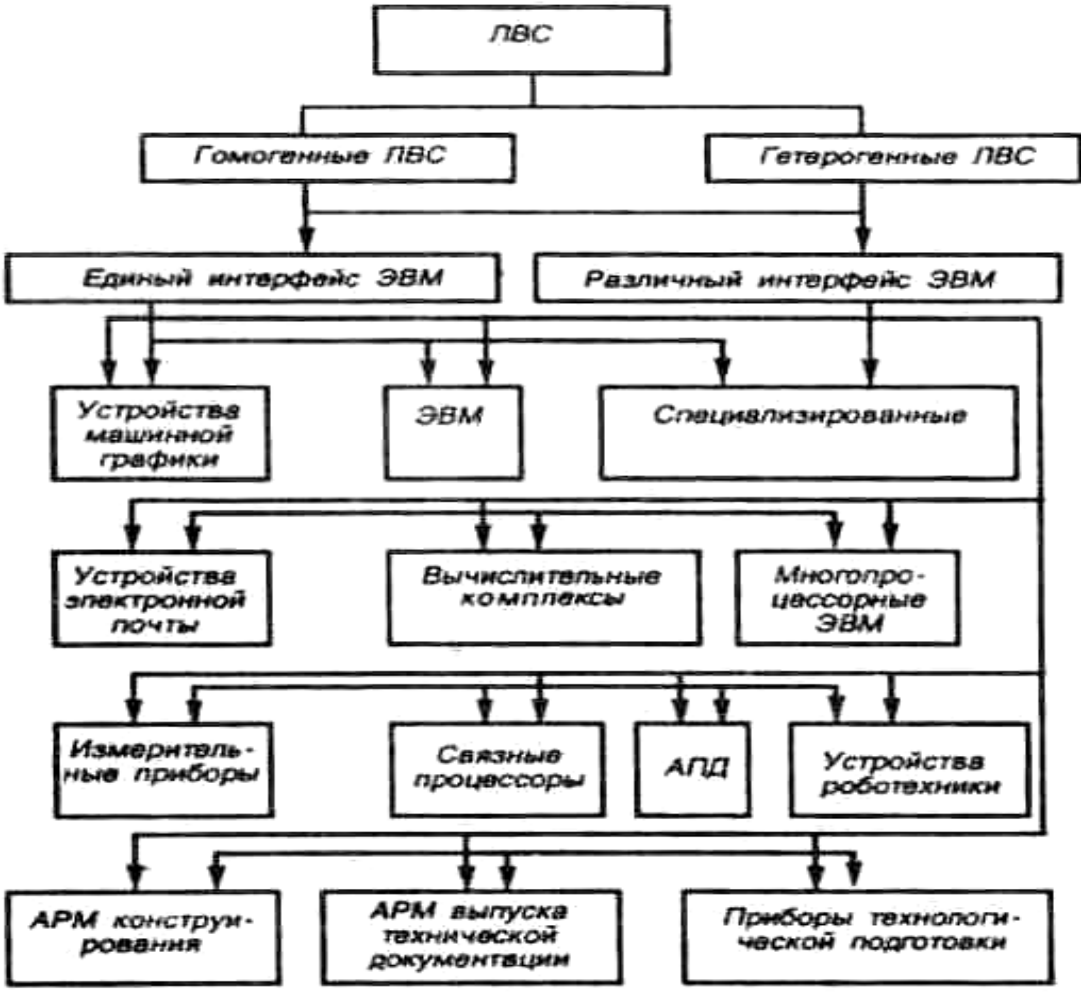


Рис. 4.3. Классификация ЛВС

– по *методам управления ресурсами* СПД – с детерминированным и случайным доступом к моноканалу;

– *программному обеспечению* — с единой операционной поддержкой и едиными методами теледоступа, ориентированными на конкретную ЛВС и ЛВС с наборами компонентов поддержки;

– *методу передачи данных* — сети с коммутацией каналов, сообщений и пакетов, причем в современных ЛВС характерно использование коммутации пакетов;

– *техническому обеспечению* — гомогенные (применение в станциях однотипного оборудования) и гетерогенные (подключение любых абонентских комплексов) ЛВС.

Анализируя способы реализации технического обеспечения САПР на базе стандартных многоуровневых структур вычислительных систем коллективного пользования и на базе ЛВС, можно сделать следующие выводы. Сетевая архитектура по сравнению со стандартной многоуровневой имеет ряд преимуществ:

1) возможность взаимодействия с одного и того же терминала с ресурсами всех рабочих и терминальных машин ЛВС;

2) обеспечение высокой надежности обработки путем замены вышедшей из строя рабочей машины резервной;

3) повышение эффективности функционирования ЭВМ за счет их специализации на выполнении определенных функций хранения и управления данными, геометрического моделирования, подготовки управляющей информации для программного управляемого оборудования.

4.2. Информационное обеспечение САПР

Базы данных в САПР. Основу ИО САПР составляет совокупность данных, которые необходимы для выполнения процесса проектирования. Совокупность данных, используемых всеми элементами САПР, называется *информационным фондом.* В этом фонде выделяют БД и архивы. Архивами пользуются редко и их помещают в долговременные ЗУ. Но в определенные моменты времени содержимое архивов помещают в БД. Информационное обеспечение представляет собой совокупность информационного фонда и средств его ведения. Основное назначение ИО состоит в создании, поддержке и организации доступа к данным. Ядром ИО является БД, которая в САПР играет роль инструмента, объединяющего отдельные элементы.

Базой данных называется структурированная совокупность связанных данных конкретной предметной области разнообразного назначения, в которой отражаются состояние объектов, их свойства и взаимоотношения.

Данные несут информацию об объектах, которые могут быть материальными (схема, плата) и нематериальными (методы оптимизации). Каждый объект характеризуется *атрибутами.* Например, объект ЭВМ можно характеризовать скоростью вычисления, объемом оперативной памяти, числом элементарных операций, числом процессоров, габаритными размерами, количеством мультиплексных каналов. Сведения, содержащиеся в каждом атрибуте, называют *значениями данных.*

Каждый объект характеризуется рядом основных атрибутов *(элементов данных).* Среди элементов данных выделяют *ключевые,* по которым можно определить другие атрибуты объекта. Например, если известен заводской номер ЭВМ, то можно определить, какая это вычислительная машина, ее скорость и т. д. Объединение значений связанных атрибутов называют *записью данных* (ф.и.о., должность). Упорядоченная совокупность записей данных – это *файл данных* или *набор данных.*

База данных — совокупность файлов, отображающая состояние объектов и их отношений в условиях САПР, совокупность файлов, специально организованная и обрабатываемая с целью создания массивов данных, их обновления и получения справок. Основное различие между БД и файлом данных состоит в том, что последний имеет несколько назначений, но соответствует одному представлению хранимых данных, а база данных, имея также несколько назначений, соответствует различным представлениям о хранимых данных.

Требования к БД:

1) целость данных — их непротиворечивость и достоверность;

2) организация БД должна обеспечивать согласование времени выборки данных прикладными программами с частотами их использования прикладными программами САПР;

3) универсальность, т. е. наличие в БД всех необходимых данных и возможность доступа к ним в процессе решения проектной задачи;

4) открытость БД для внесения в нее новой информации;

5) наличие языков высокого уровня взаимодействия пользователей с БД;

6) секретность, т. е. невозможность несанкционированного доступа к информации и ее изменений;

7) оптимизация БД — минимизация избыточности данных.

Одним из принципов построения САПР является информационная согласованность частей ее программного обеспечения, т. е. пригодность результатов выполнения одной проектной процедуры для использования другой проектной процедуры без их трудоемкого ручного преобразования пользователем. Отсюда вытекают условия информационной согласованности:

– использование программами одной и той же подсистемы САПР единой БД;

– применение единого внутреннего языка для представления данных.

Единство информационного обеспечения достигается либо созданием единой БД, либо сопряжением нескольких БД с помощью специальных программ, которые перекодируют информацию, приводя ее к требуемому виду. Части ПО и методы, осуществляющие управление базой данных, составляют *систему управления базами данных* (СУБД), которая позволяет получить доступ к интегрированным данным и допускает множество различных представлений о хранимых данных. Программное обеспечение, которое позволяет прикладным программам работать с БД без знания конкретного способа размещения данных в памяти ЭВМ, называют СУБД (рис. 4.4).

Рис. 4.4. Схема СУБД

СУБД выступает как совокупность программных средств, предназначенных для создания, ведения и совместного использования БД

многими пользователями. СУБД должна обеспечивать: простоту физической реализации; возможность централизованного и децентрализованного управления БД; минимизацию избыточности хранимых данных; предоставление пользователю по запросам непротиворечивой информации; простоту разработки, ведение и совершенствование прикладных программ; выполнение различных функций.

СУБД реализует два интерфейса:

• между логическими структурами данных в программах и в БД;

• между логической и физической структурами БД.

Порядок работы СУБД в одном из режимов – следующий:

1) программа запрашивает возможность чтения данных у СУБД, она передает необходимую информацию о программисте, типе записи;

2) осуществляет поиск описания данных, на которые выдан запрос;

3) определяет, какого типа логические и физические записи необходимы;

4) выдает ОС запрос на чтение требуемой записи;

5) ОС взаимодействует с физической памятью;

6) записывает запрошенные данные в системные буферы;

7) выделяет требуемую логическую запись, выполняя необходимые преобразования;

8) передает данные из системных буферов в программу пользователя, а затем программе пользователя – информацию о результатах выполнения запроса;

9) прикладная программа обрабатывает полученные данные.

Проектирование баз данных. Процесс разработки структуры БД на основании требований пользователя называют проектированием БД. Результатом проектирования БД является структура БД, состоящая из логических и физических компонент, и руководство для прикладных программистов.

Жизненный цикл БД делится на стадии анализа, проектирования и эксплуатации. *Анализ* – формулирование требований концептуального проектирования. Основная цель – обеспечить согласованность целей пользователей и представлений об информационных потоках.

Проектирование — реализация БД, анализ функционирования и поддержки, модификация и адаптация.

Логическое проектирование состоит в проектировании БД и программ. Его результатом является логическая структура БД, функциональное описание программных модулей и наборы запросов БД.

Физическое проектирование – это выбор физической структуры БД и отладка программных модулей, полученных при проектировании программ. Результатом является подготовка к эксплуатации БД.

Реализация БД — задача разработки программ доступа к БД.

В БД используются языки описания данных (ЯОД) и языки манипулирования данными (ЯМД).

ЯОД определяет различные типы записей, их имена, форматы, служит для определения типов элементов данных, которые нужны в качестве ключей; отношений между записями или их частями; типа данных, которые используются в записях; диапазона их значений; числа элементов, их порядка; режима доступа.

Различают три уровня абстракции для описания данных:

– концептуальный (с позиции администратора);

– реализации (с позиции программиста и пользователя);

– физический (с позиции системного программиста).

На концептуальном уровне описывают объекты, атрибуты и значения данных. На уровне реализации имеют дело с записями, элементами данных и связями между записями. На физическом уровне оперируют блоками, указателями, данными переполнения, группировкой данных.

ЯМД дает возможность манипулировать данными без знания несущественных для программиста подробностей. Они могут реализовываться как расширение языков программирования общего назначения путем введения в них специальных операторов или путем реализации специального языка.

Процесс проектирования БД начинают с построения *концептуальной модели* (КМ), которая состоит из описания объектов и их взаимосвязей без указания способов физического хранения. Построение КМ ведется с анализа данных об объектах и связей между ними, сбора информации о данных в существующих и возможных прикладных программах. КМ является моделью предметной области.

Версия КМ, обеспечиваемая СУБД, называется *логической моделью* (ЛМ). Подмножества ЛМ, которые выделяются для пользова-

телей, – это *внешние модели* (подсхемы). ЛМ отображается в физическую, которая описывает размещение данных и методы доступа. Физическую модель называют *внутренней.*

Внешние модели не связаны с используемыми КТС и методами доступа к БД (рис. 4.5). Они определяют первый уровень независимости данных. Второй уровень независимости данных связан с отсутствием изменений внешних моделей при изменении КТС (КМ).

При разработке и проектировании БД важным является *словарь данных* (СД), который предназначен для хранения сведений об объектах, атрибутах, значениях данных, взаимосвязях между ними, их источниках, форматах представления. СД позволяет получить информацию обо всех ресурсах данных. Назначение СД – документирование данных, централизованное ведение и управление данными, взаимодействие между разработками САПР.

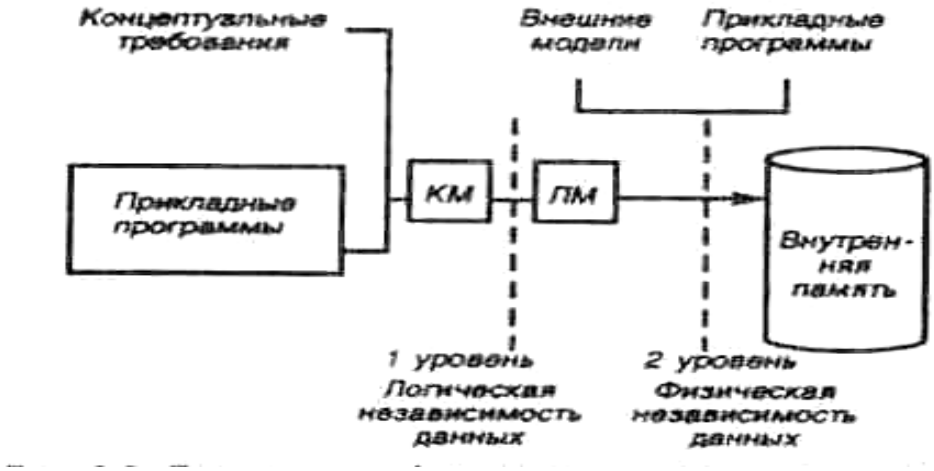


Рис. 4.5. Логическая и физическая независимость данных

Словарь данных может быть в виде части пакета программ СУБД или отдельного пакета программ в виде дополнения к СУБД (рис. 4.6).

СД — связующее звено в ПО обработки данных, которое включает в себя процессор, СУБД, языки запросов, монитор телеобработки. СД обязаны поддерживать КМ, ЛМ, внешнюю и внутреннюю модели; обеспечивать обмен информацией с СУБД и процесс изменений БД.

Словарь данных имеет свою БД, которая включает в себя: атрибут; объект; групповой элемент данных; выводимый объект данных; *синонимы,* т. е. атрибуты, имеющие одинаковое назначение, но раз-

личные идентификаторы; *омонимы,* т.е. атрибуты с различным назначением, но с одинаковыми идентификаторами; описание КМ, ЛМ, внешних и внутренних моделей; описание, позволяющее пользователям формально и однозначно выбирать атрибуты для решения задач.

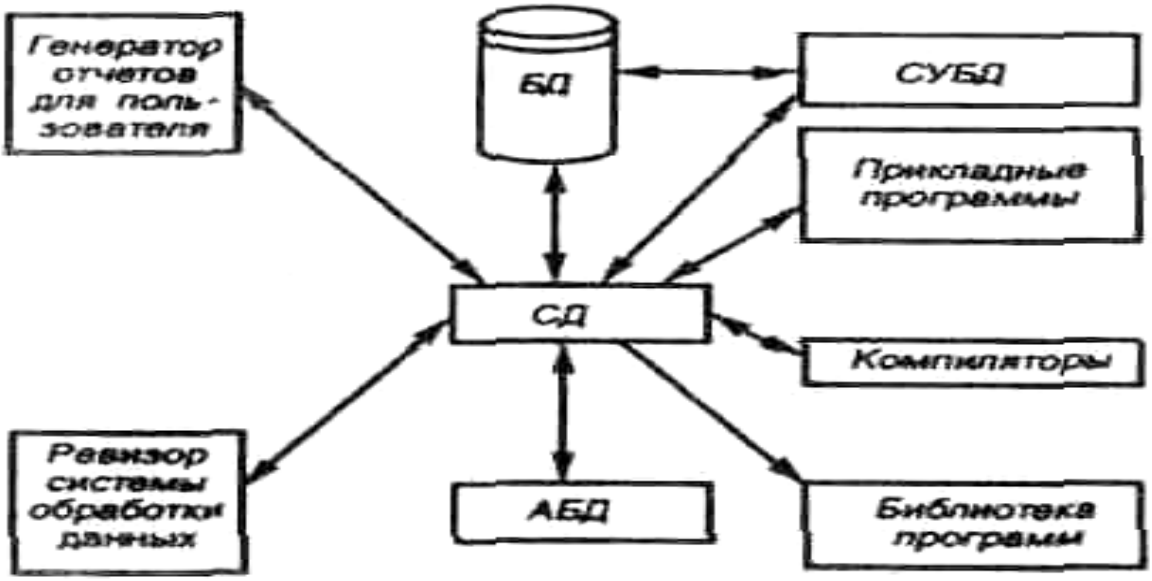


Рис. 4.6. Интерфейсные СД в системе с БД

При проектировании БД (рис. 4.7) выполняется идентификация основных объектов предметной области и прикладных программ, подлежащих использованию; определяются объекты и их взаимосвязи; разрабатываются СД, КМ, ЛМ, ФМ с проведением анализа и оценок.

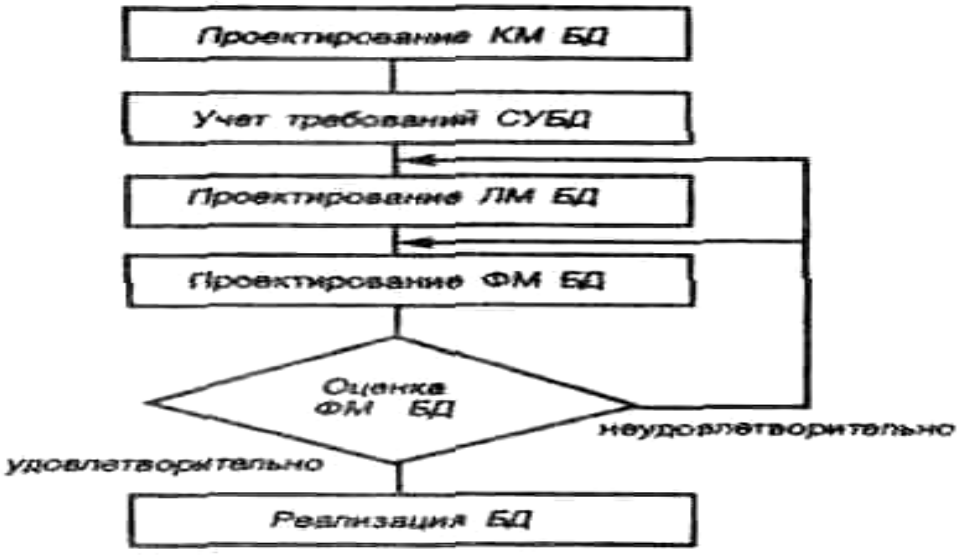


Рис. 4.7. Структурная схема проектирования БД

Например, концептуальное структурное представление информации (ISP-информация) не связано с конкретными способами обработки и приложениями, а описывает концептуальные связи в БД:

Описание сущности
наименование ..... ЭВМ
количество 10
Описание атрибута
наименование ..... номер
скорость вычислений ..... 5 000 000 оп/с
вероятность существования 1,0
Описание связи
наименование ..... –
сущность ..... работает на ВЦ
отображение 1:1
вероятность существования 1,0

Информация, описывающая концептуальное представление (UP-информация), определяет требование организации к обработке данных и описывает данные и связи:

Описание процесса
наименование ..... компоновка
частота применения ..... еженедельно
вероятность применения 1,0
приоритет ..... высший
требуемые данные ..... схема
правила проектирования..... ГОСТ
объем данных ..... 100 схем
Оператор
оператор ..... разбивка
критерий поиска..... минимум внешних связей
вероятность события 1,0

Концептуальное представление информации обеспечивает эффективность проектирования, структурное представление информации создает гибкость и адаптивность.

4.3. Модели данных. Реляционная, иерархическая и сетевая модели данных

Современные СУБД основываются на использовании моделей данных (МД), позволяющих описывать объекты предметных облас-

тей и взаимосвязи между ними. Существуют три основные МД и их комбинации, на которых основываются СУБД: реляционная модель данных (РМД), сетевая модель данных (СМД), иерархическая модель данных (ИМД).

Основное различие между МД состоит в описании взаимодействий между объектами и атрибутами. Взаимосвязь выражает отношение между множествами данных, используя взаимосвязи «один к одному», «один ко многим», «многие ко многим».

«Один к одному» – это взаимно однозначное соответствие, которое устанавливается между одним объектом и одним атрибутом. Например, в определенный момент времени в одной ЭВМ используется один определенный процессор. Номеру выбранной ЭВМ соответствует номер выбранного процессора.

«Один ко многим» – соответствие между одним объектом и многими атрибутами.

«Многие ко многим» — соответствие между многими объектами и многими атрибутами. Например, на множестве ЭВМ может одновременно работать множество пользователей. Взаимосвязи между объектами и атрибутами удобно представлять в виде графов и гиперграфов.

Реляционная модель данных. В этой модели объекты и взаимосвязи между ними представляют в виде таблиц, и операции взаимодействия между данными осуществляются по правилам реляционной алгебры (объединение, пересечение, вычитание, декартово произведение, проекция, ограничение, соединение, деление).

Основу РМД составляют данные, сформированные в виде таблиц. Формальным аналогом таблицы является отношение. Напомним, что для совокупности множеств D_1 .... D_n отношением R является некоторое подмножество декартова произведения этих множеств:

$$R = D_1 * D_2 * \ldots * D_n,$$

где множество D_i — сомножители;

n — степень отношения R;

декартово произведение $D_1 * D_2 * \ldots * D_n$ — *домен,* из которого извлекают фактические значения.

Набор конкретных значений R называется *кортежем.* Если кортежи записать друг под другом и обозначить столбцы, то получается *таблица представления множества:*

A_1	A_2
a_2	b_2
a_4	b_1
a_4	b_2

В реляционной модели оперируют только с *нормализованными отношениями,* когда каждый столбец сам отношением не является. Ненормализованные отношения приводятся к нормализованной форме путем корректировки обозначений столбцов.

Таблица, состоящая из строк и столбцов, называется *отношением.* Каждый столбец в таблице является атрибутом, а строки таблицы – кортежами, т. е. упорядоченными множествами. Значения в столбце определяются из множества значений, которые принимает атрибут. Столбцы таблицы — элементы данных, а строки — записи. Первичным ключом в таблице является номер этапа проектирования. Таблица имеет два атрибута и шесть кортежей (табл. 4.1).

Таблица 4.1

Представление данных с помощью РМД

Номер	Этапы проектирования	Атрибуты	
		Сложность	Время реализации
1	Покрытие	$0(n^3)$	минуты
2	Типизация	$0(n^2)$	секунды
3	Компоновка	$0(n^3)$	минуты
4	Размещение	$0(n^3)$	минуты
5	Трассировка	$0(n^4)$	часы
6	Контроль	$0(n^3)$	минуты

Достоинства РМД — простота и доступность. Пользователи абстрагированы от физической структуры памяти. Это позволяет эксплуатировать БД без знания методов и способов ее построения. Также достоинствами РМД являются независимость данных, гибкость, непроцедурные запросы.

К недостаткам РМД относят низкую производительность по сравнению с ИМД и СМД, сложность ПО, избыточность.

Иерархическая модель данных. Модель основана на понятии дерева, состоящего из вершин и ребер. Вершина дерева ставится в соответствие совокупности атрибутов данных, характеризующих

некоторый объект. Вершины и ребра дерева образуют иерархическую древовидную структуру, состоящую из *n* уровней (рис. 4.8).

Первую вершину в дереве называют *корневой вершиной.* ИМД удовлетворяет следующим условиям:

1. Иерархия начинается с корневой вершины.

2. Каждая вершина соответствует одному или нескольким атрибутам.

3. На уровнях с большим номером находятся зависимые вершины. Вершина предшествующего уровня является начальной для новых зависимых вершин.

4. Каждая вершина, находящаяся на уровне *I*, соединена с одной и только одной вершиной уровня (*I* – 1), за исключением корневой вершины.

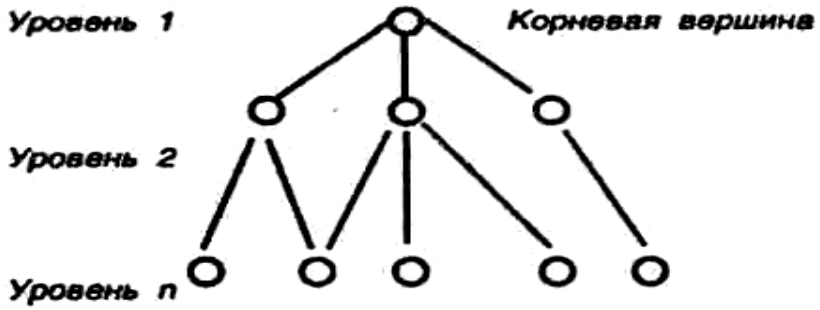


Рис. 4.8. Пример иерархической древовидной структуры

5. Корневая вершина может быть связана с одной или несколькими зависимыми вершинами.

6. Доступ к каждой вершине происходит через корневую по единственному пути.

7. Существует произвольное количество вершин каждого уровня. ИМД, состоящая из нескольких деревьев, является лесом. Каждая корневая вершина образует начало записи логической БД. В ИМД вершины, находящиеся на уровне *I*, называют порожденными вершинами на уровне (*I* – 1).

Рассмотрим пример представления информации в ИМД, реализующей отношение «один ко многим» (рис. 4.9). Для каждого пользователя может иметься экземпляр корневой вершины. ИМД позволяет для каждого пользователя получать представление о нескольких операциях и нескольких ЭВМ.

ПОЛЬЗОВАТЕЛЬ соответствует корневой вершине и находится на более высоком уровне иерархии, чем ЭВМ, ОПЕРАЦИЯ

и РЕЗУЛЬТАТ. Выбор ИМД осуществляет администратор БД на основе операционных характеристик.

Рис. 4.9. Представление информации в РМД, реализующей отношение «один ко многим»

Введение двух ИМД, связанных между собой, позволяет решать вопросы включения и удаления данных. Достоинствами ИМД считаются: простота построения и использования, обеспечение определенного уровня независимости данных, наличие существующих СУБД, простота оценки операционных характеристик.

Недостатки состоят в следующем: отношение «многих ко многим» реализуется очень сложно, дает громоздкую структуру и требует хранения избыточных данных, что нежелательно на физическом уровне; иерархическая упорядоченность усложняет операции удаления и включения; доступ к любой вершине возможен через корневую, что увеличивает время доступа (рис. 4.10).

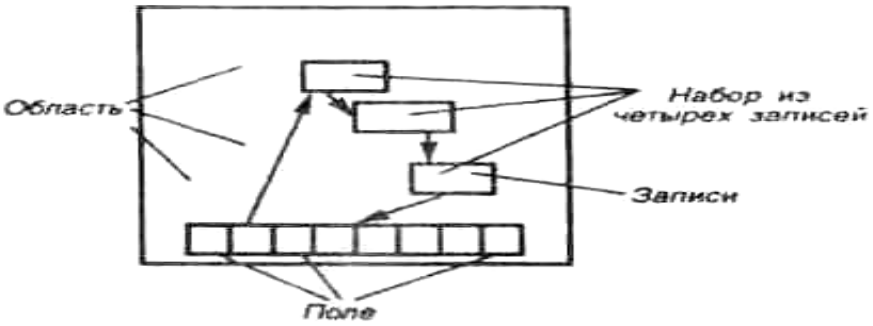


Рис. 4.10. Пример размещения записей в областях

Сетевая модель данных. В СМД элементарные данные и отношения между ними представляются в виде ориентированной сети (вершины – данные, дуги – отношения). БД, описываемая сетевой

моделью, состоит из нескольких областей (рис. 4.11). Область содержит записи. Одна запись состоит из нескольких полей. Набор, состоящий из записей, может размещаться в одной или нескольких областях.

В СМД объекты предметной области объединяются в сеть. Графически сетевая модель описывается прямоугольниками и стрелками. Каждый тип записи может содержать несколько атрибутов.

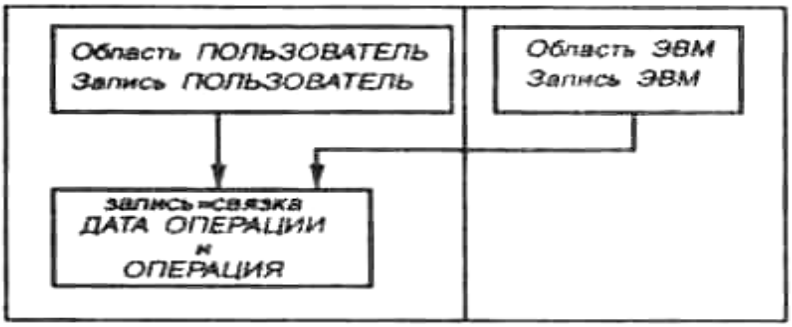


Рис. 4.11. Сетевая модель данных

На рис. 4.12 показан пример представления области в СМД. Здесь область — это часть БД, в которой располагаются записи, наборы и части наборов. Стрелками соединены несколько типов записей, изображающих типы набора. Тип набора предоставляет логическую взаимосвязь «один ко многим». Каждый экземпляр набора ПОЛЬЗОВАТЕЛЬ — ВЫПОЛНИЛ — ПРОЦЕДУРУ реализует иерархическую связь между пользователем и операцией.

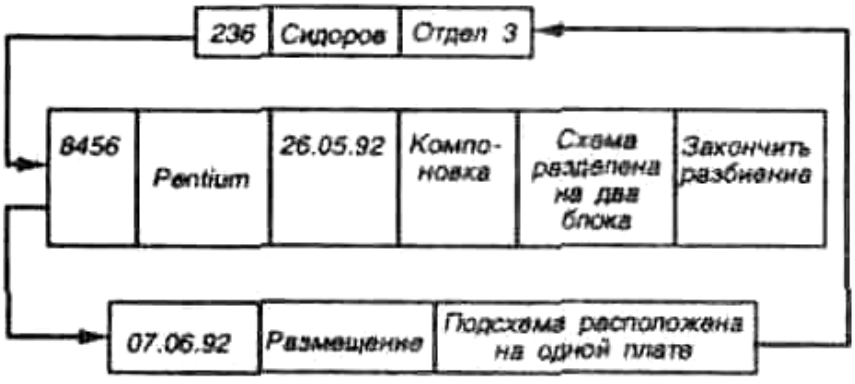


Рис. 4.12. Экземпляр набора в СМД

Важное отличие СМД от ИМД состоит в том, что в СМД каждая запись может быть в любом числе наборов и находиться как на верхнем, так и на нижнем иерархическом уровне. Следовательно, любая запись может быть задана как точка входа.

Существует три типа наборов:

а) первый тип образуется типами записей: ПОЛЬЗОВАТЕЛЬ и ОПЕРАЦИЯ (рис. 4.13, *а*);

б) второй тип образуется из трех и более записей и называется многочисленным (рис. 4.13, *б*);

в) третий тип представляется сингулярными наборами и объединяет записи, у которых нет корневой вершины, но которые могут иметь ее впоследствии (рис. 4.13, *в*).

Достоинства СМД — наличие реализованных СУБД; простота реализации отношений «многие ко многим».

Недостаток СМД — сложность. При реорганизации БД возможна потеря независимости данных.

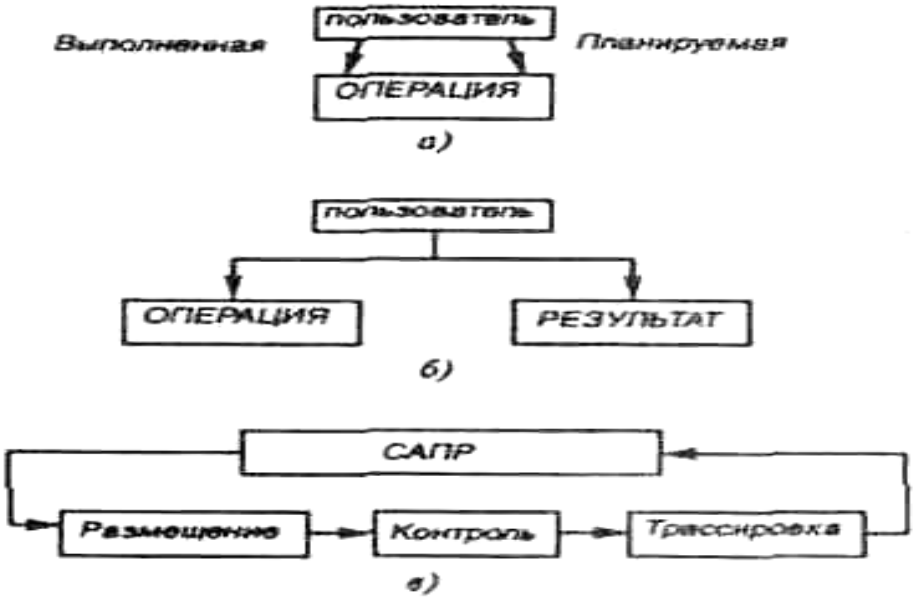


Рис. 4.13. Виды записей в СМД

Система управления базами данных. Задачи автоматизации проектирования и производства связаны с хранением значительного объема информации в виде единообразных записей и со сравнительно простой их обработкой для извлечения новой информации в удобной для пользователя форме. Предназначенные для этого программные средства обычно объединяются под общим названием системы управления базами данных, в которой под базой данных понимается совокупность взаимосвязанных и хранящихся в удобном для использования виде данных.

Наилучшая форма организации данных зависит как от их содержания, так и от того, как эти данные должны использоваться. Например, существует немало СУБД, предназначенных для обработки спи-

сков, содержащих разные сведения о перечисленных в этих списках людях. Эти СУБД существенно различаются в зависимости от того, какие сведения о каждом лице имеются в списке (в зависимости от логической структуры данных), и от того, как осуществляется поиск того или иного лица.

Вообще, *логической структурой данных* принято называть конкретные логические взаимосвязи между данными, обусловленные содержанием этих данных. Например, логической структурой стипендиальной ведомости студенческой группы является взаимосвязь каждой фамилии студента с величиной его стипендии и взаимосвязь всех перечисленных в ведомости фамилий с номером группы.

Совокупность количества взаимосвязанных данных (*полей)* принято называть *логической записью,* а совокупность логических записей данного вида — *файлом.* Таким образом, каждая точка стипендиальной ведомости образует отдельную логическую запись, состоящую из двух полей — фамилии и величины стипендии (например, ИВАНОВ 9000), а вся ведомость образует файл.

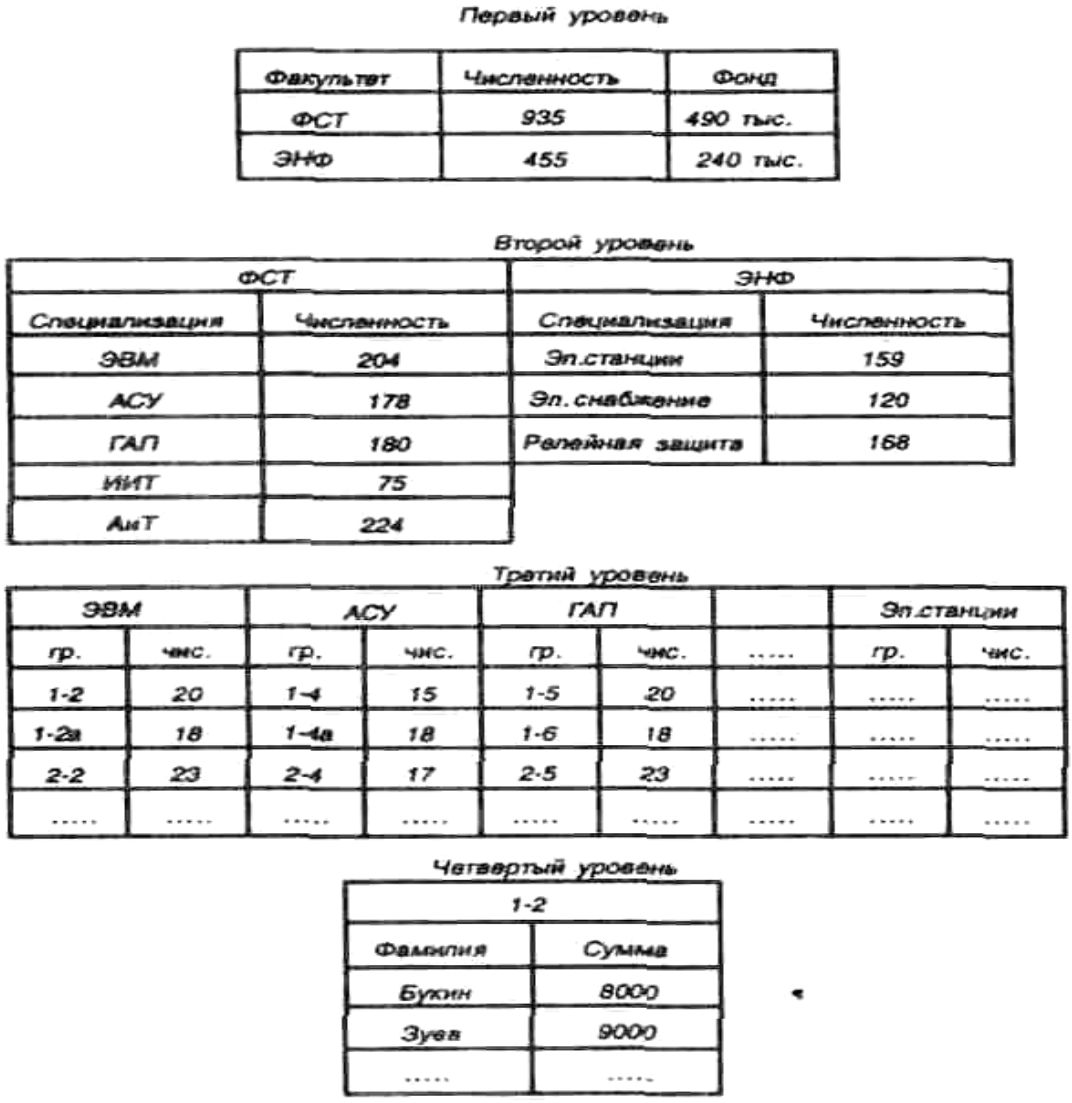
В базе данных может одновременно храниться много различных файлов, поэтому пользователь присваивает каждому файлу свое имя. Всякая БД должна храниться в ЗУ ЭВМ. Форма организации данных зависит не только от логической структуры, но и от ее физического представления, т. е. от способа хранения данных в памяти ЭВМ. Порцию данных, которую СУБД считывает из ЗУ или записывает в ЗУ как единое целое, принято называть *физической записью.*

Одной из самых важных функций СУБД является поиск логической записи в файле. Поиск производится по ключу, который может состоять из поля, части поля, из комбинации полей. Например, при поиске величины стипендии студента по фамилии ИВАНОВ ключом является фамилия ИВАНОВ.

Категории баз данных. По характеру связи между логическими записями, полями и ключами все БД делятся на три категории: иерархические, сетевые и реляционные.

Иерархическая БД. В ней существует упорядоченность полей внутри каждой логической записи: одно поле считается главным, а другие поля ему подчинены (рис. 4.14). Группы логических записей упорядочиваются в определенную последовательность, как ступеньки лестницы, и поиск данных ключей осуществляется прохождением по этой лестнице в соответствии с порядком, установленным последова-

тельностью главных полей. В качестве примера рассмотрим данные о размерах стипендии всех студентов института.



Первый уровень

Факультет	Численность	Фонд
ФСТ	935	490 тыс.
ЭНФ	455	240 тыс.

Второй уровень

ФСТ		ЭНФ	
Специализация	Численность	Специализация	Численность
ЭВМ	204	Эл.станции	159
АСУ	178	Эл.снабжение	120
ГАП	180	Релейная защита	168
ИИТ	75		
АиТ	224		

Третий уровень

ЭВМ		АСУ		ГАП			Эл.станции	
гр.	чис.	гр.	чис.	гр.	чис.	.....	гр.	чис.
1-2	20	1-4	15	1-5	20	.....	.....	.....
1-2а	18	1-4а	18	1-6	18	.....	.....	.....
2-2	23	2-4	17	2-5	23	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....	.....	.....

Четвертый уровень

1-2	
Фамилия	Сумма
Букин	8000
Зуев	9000
.....	.....

Рис. 4.14. Структурная схема иерархической БД

Последовательно пройдя через несколько уровней иерархии, можно отыскать нужную информацию, такую, например, как размер стипендии студента по фамилии Иванов. В результате для многоуровневой иерархической БД время ответа на запрос может быть весьма продолжительным.

Первый уровень в организации логической структуры такой базы может включать таблицу, содержащую название факультетов, численность студентов этих факультетов, получающих стипендию, и стипендиальный фонд факультетов. Название факультета выступает в качестве главного поля, и, выбрав нужный факультет, можно получить доступ к таблице второго уровня.

Второй уровень иерархии может включать таблицы, содержащие перечни специализаций каждого факультета. При этом в качестве главного поля выступает название специализации.

В таблицах следующих уровней может содержаться информация о номере студенческих групп (главное поле) по каждой специализации, их численности и о размерах стипендии, получаемых студентами каждой группы.

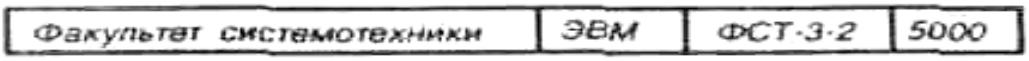
Сетевая база данных. Обладает несколько большей гибкостью по сравнению с иерархической, поскольку в сетевой БД можно установить связи не только между файлами соседних уровней, но и перескакивать через один или несколько уровней.

Так, в БД связь может быть установлена между списком факультетов и таблицей номеров студенческих групп, и тогда станет возможным отыскать номер студенческой группы без предварительного поиска промежуточной таблицы, содержащей перечень специализации данного факультета.

Реляционная база данных. Эта БД обладает наиболее гибкой логической структурой. При иерархической или сетевой организации БД ответы на одни виды запросов занимают значительно меньше времени, чем на другие. Более того, сказать, на какие запросы идет много времени, а на какие мало, можно только после того, как база данных уже создана.

Например, в иерархической БД самое продолжительное время займет ответ на запрос о размере стипендии конкретного студента, поскольку поле «сумма», содержащее размер стипендии, находится в таблице самого последнего уровня.

К тому же лишь только после затрат большого труда на создание такой БД может выясниться, что «неудобные» для базы запросы, требующие продолжительного времени на ответ, поступают чаще всего. От подобной неприятности избавлена реляционная БД, потому что все поля в логических записях такой базы равноправны: нет главного поля и подчиненных ему полей. Например, если преобразовать иерархическую БД в реляционную, то получится файл, содержащий 935 + 455 = 1390 логических записей (по одной записи на каждого студента). Каждая из этих записей может содержать, например, следующие поля: факультет, специализация, группа, фамилия, сумма. Так, студенту ИВАНОВУ из группы ФСТ-3-2 соответствует следующая логическая запись:



Не следует думать, что не возникает проблем при поиске информации в реляционной БД. Если 1390 логических записей, подобных этой, не подчиняются никакому заранее установленному в файле порядку, то ответ на запрос о размере стипендии студента ИВАНОВА связан со сплошным просматриванием всех записей файла. С увеличением общего количества записей в файле такая процедура поиска потребует весьма значительного времени. Чтобы ускорить процесс поиска, записи в файле можно упорядочить, например, по алфавитному порядку фамилий студентов. Упорядочение записей позволяет ускорить их поиск с помощью так называемой *двоичной схемы,* или *двоичного поиска.*

Согласно двоичной схеме, выбирается запись, находящаяся посредине файла. По фамилии студента в этой записи легко судить, в какой из двух половин файла находится искомая запись. Затем процесс повторяется до тех пор, пока не будет найдена нужная запись. Пусть, например, в самой середине файла оказалась запись с фамилией ОСОКИН. Сравнивая первые буквы этой фамилии с искомой фамилией ИВАНОВ, легко понять, что искомая запись находится в первой половине файла. Пусть далее записью, которая делит первую половину файла пополам, оказалась запись, представленная на рис. 4.15.

Тогда искомая запись находится во второй четверти файла. Продолжая далее точно так же деление второй половины файла пополам, получим 1/8 и 1/16 части файла и т. д. В конце концов будет найдена нужная запись, и дан ответ на запрос о размере стипендии студента ИВАНОВА.

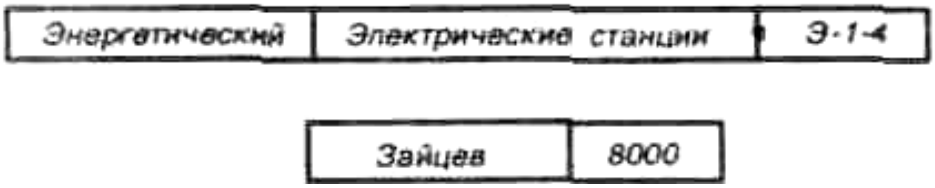


Рис. 4.15. Запись, делящая файл БД пополам

Двоичный поиск ведется быстрее, чем поиск просмотром. Пусть файл, в котором ведется поиск, включает *N* логических записей. Если эти записи не упорядочены, то окажется, что среднее арифметическое

количество записей, которое нужно просмотреть, чтобы ответить на большое количество запросов, приближается к $N11$ с ростом числа запросов. Если записи упорядочены, то при двоичном поиске количество просмотренных при ответе на любой запрос записей не будет превышать число $1\log_2 N$, округленное до ближайшего целого.

Поскольку $\log_2 N$ во всяком случае намного меньше $N/2$ при большом N, то максимальное время, необходимое для ответа на запрос при двоичном поиске, значительно меньше среднего времени ответа при поиске в неупорядоченном файле.

Однако упорядоченный файл имеет и недостатки: во-первых, при большом количестве логических записей пополнение такого файла новыми записями связано с перемещением старых записей, чтобы не нарушать упорядоченность файла в целом, во-вторых, всякая запись упорядоченного файла может быть найдена только по значению того поля, по которому упорядочены все записи файла. Например, если бы все 1390 записей в примере были упорядочены в файле не по фамилиям, а по номерам студенческих групп, то найти размер стипендии студента ИВАНОВА (с помощью двоичного поиска, не зная номера группы этого студента) было бы невозможно. Конечно, можно создать несколько дубликатов одного и того же файла, упорядоченных по значению разных полей, но для этого потребуется много дополнительного места в памяти ЭВМ. Еще более гибкая (двоичная схема) процедура связана с *хешированием* (перемешиванием).

Согласно этой процедуре, каждой логической записи в файле по некоторому правилу ставится в соответствие число, которое аналогично адресу этой записи в памяти компьютера. Например, если пронумеровать все буквы русского алфавита, то отдельные буквы фамилии студента ИВАНОВ будут иметь соответственно двухрядные номера: 10, 03, 01, 15, 16, 03. Составленное из этих номеров число 100301151603 порождает адрес логической записи в памяти ЭВМ.

Сложность такого подхода заключается в том, что адреса логических записей будут получаться очень большими (например, адрес записи с фамилией ИВАНОВ получится больше 100 млрд), так что для размещения этих записей потребуется невероятно большой объем памяти компьютера, а при этом память может оставаться очень слабозаполненной, поскольку большинство комбинаций букв не образуют фамилий.

Выход из этого затруднения можно найти, если предложить другое правило вычисления адреса логической записи, например: если сложить номера (10+3+1+15+16+3 = 48) букв фамилии ИВАНОВ, то полученную сумму 48 тоже можно рассматривать как адрес записи и разместить эту запись в 48-й ячейке памяти.

Здесь адрес не является очень большим числом, а для размещения записи уже не требуется, чтобы память компьютера имела большой объем. Однако такой метод приводит иногда к ситуациям, когда в двух записях получится один и тот же адрес. Например, для записи, соответствующей студенту по фамилии ИВАНОВ, тоже получится адрес 48.

Всякий метод вычисления адреса логической записи по ее содержимому называется *хеш-функцией.* Практически никакая хеш-функция не гарантирует от конфликтов, поэтому на случай конфликтов договариваются использовать некоторую другую (дополнительную) хеш-функцию. Этого бывает достаточно для устранения всех конфликтов. Если хеш-функция выбрана удачно, то логические записи распределяются более или менее равномерно по ячейкам памяти компьютера (точнее, по той области памяти, которая отведена на данный файл), и конфликты будут встречаться редко. Если бы, например, заданию была известна относительная доля появления каждой буквы алфавита в большом количестве фамилий студентов, то можно было бы раз и навсегда сконструировать самую удачную из всех возможных хеш-функций. Однако такой благоприятной ситуации не бывает.

При хешировании значение хеш-функции, т. е. адрес логической записи, вычисляется всякий раз при поиске этой записи в файле. Вычислить хеш-функцию несложно, и когда ее значение, т. е. адрес искомой ячейки, известно, выбор нужной логической записи осуществляется прямо по адресу хеширования — чрезвычайно быстрый метод поиска. При использовании хеширования дополнение файла новыми логическими записями тоже эффективно, поскольку при этом не сохраняют какую-либо упорядоченность файла в целом.

Все рассмотренные методы поиска информации в настоящее время получили широкое распространение в системах управления небольшими БД для ПЭВМ. Для СУБД, создаваемых на мощных компьютерах, разработаны другие, более сложные методы поиска, требующие более сложных, но зато и более эффективных программ.

4.4. Принципы организации САПР с элементами искусственного интеллекта. Анализ современных требований к САПР

Основные поколения существующих САПР:

• моделирующие (системы автоматизированного моделирования);

• синтезирующие (синтез выполняется не через многократное моделирование или оптимизацию, а путем создания сразу работоспособного варианта — генерационный синтез);

• САПР с формальными языками, где средства общения с пользователем ограничены определенными конструкциями, не подлежащими изменениям;

• САПР с неформальными языками, где имеется переменная (перестраиваемая) лексика со свободной грамматикой и синтаксисом. Существующие САПР представлены поколениями, где описание объекта выполнено формальными языками. Создание синтезирующих САПР с неформальными языками зависит от использования методов ИИ. Поэтому все САПР можно разделить на интеллектуальные и неинтеллектуальные.

Обычно САПР работала в жестком режиме, по строго заданным алгоритмам, не используя опыт проектировщика, не обеспечивая взаимодействия пользователя с ЭВМ на языке, близком к естественному. Существует три метода интеллектуализации САПР:

1. Внешняя универсальная интеллектуализация с помощью инструментальных систем искусственного интеллекта (ИСИИ), где используются различные оболочки экспертных, диалоговых, обучающих систем. Главное достоинство — высокая скорость разработки и малые финансовые затраты. Недостаток — низкое качество проектирования (скорость работы, требуемая память, надежность, мощность, учет всех особенностей предметной области). ИСИИ удобно пользоваться на начальных этапах разработки ЭВМ.

2. Внешняя специализированная интеллектуализация с помощью специализированных программных приставок, работающих на принципах ИИ. Это метод как развитие предыдущего, но с более высоким качеством работы. Здесь все сводится к улучшению сервисных характеристик САПР, что обеспечивает возможность формулировки типовой задачи проектирования на предметно-ориентированном язы-

ке, организации обучения пользователя и т.д. Совокупность средств общения пользователя с САПР представляет собой интеллектуальный интерфейс.

3. Внутренняя интеллектуализация со встроенными в САПР алгоритмами и методами ИИ (расширены возможности синтеза, адаптации, самоорганизации, работы с нечеткой формулировкой задачи).

Как известно, в ИИ можно выделить бессловесный ИИ (низкий уровень, где нет интеллектуального интерфейса, пользователь работает в терминах предметной области, но присутствуют процедуры, имитирующие целесообразное поведение, например, адаптацию, самоорганизацию), словесный ИИ (высокий уровень, где есть интеллектуальный интерфейс с языком предметной области, используются понятия для представления знаний на любых символах и знаках с явно заданной семантикой) и искусственный разум (технически не реализован) [1, 42].

Системы, в которых существенную роль играют понятия, миры слов, т. е. семиотические (знаковые) с их явно заданной семантикой, называют *понятийными.*

Традиционные САПР, дополненные адаптационными процедурами приспособления к оперативной проектной обстановке, самонастройкой на особенности постановки задачи, поиска и обоснованного выбора цели среди множества вариантов, т. е. процедурами, имитирующими целенаправленное поведение, следует считать интеллектуальными САПР (ИСАПР) нижнего уровня. У понятийных САПР, способных работать с различными понятиями, семантика, смысл которых определяется либо пользователем на входном языке, либо автоматически внутри программы, уровень интеллекта выше.

Традиционные САПР работают по строгому алгоритму, понятийные САПР порождают иной алгоритм решения. Это позволяет пользователю применять эвристические, словесные, нематематические правила. Понятийные САПР (развитие традиционных САПР, алгоритмических) используют информацию не в количественном, а в качественном виде, например: «если X_1 имеет свойство a, то X_2 имеет свойство b». Эти высказывания (предикаты) можно рассматривать как качественные аналоги количественных формул.

Наглядным примером понятийной формулы является *фрейм.* Фрейм-прототип можно рассматривать как понятийный аналог функции $Y = F(X_1, X_2, ..., X_n)$, где F — имя фрейма-прототипа (сложного понятия), а $X_1, X_2, ..., X_n$ – имена его незаполненных слотов (простых

понятий). Заполняя слоты разными значениями имен (понятийных аргументов), получаем различные значения имени фрейма, т. е. понятийной функции.

Архитектура интеллектуальных САПР

Традиционная САПР характеризуется:

• описанием объекта и задач проектирования с помощью формализованных, фиксированных, символьно-цифровых языковых конструкций;

• ориентацией работы САПР на жесткую, формализованную постановку задачи проектирования;

• использованием процедуры моделирования как основы процесса проектирования;

• использованием алгоритмов неадаптируемых и не учитывающих опыта пользователя;

• фиксированием функциональной структуры САПР;

• применением в качестве основы математического аппарата теории численных методов, теории множеств, имитационного моделирования;

• представлением информации в численном виде;

• использованием традиционных языков (Паскаль, Си и др.);

• хранением больших объемов информации в виде БД;

• представлением выходной информации в виде таблиц, графиков, чертежей.

Архитектура традиционных САПР отражает черты административно-командной системы, перенесенные в технику: чрезмерная централизация управления, штампы в языковых средствах, негибкость алгоритмов и принципов работы, представление информации в неосмысленном виде, сложность модификации, невозможность учета человеческого фактора, т. е. опыта и его психологии.

Основные концепции ИСАПР состоят в следующем:

1. Входная информация представляется в виде фраз на ограниченном естественном языке или на предметно-ориентированном языке, допускающем описание пользователем не только объекта, но самого алгоритма проектирования. Тем самым достигается высокая функциональная гибкость САПР. Постановка задачи может быть нечеткой или некорректной, тогда ИСАПР путем диалога с пользователем устраняет нечеткость и некорректность. Это заключается в распознавании задачи и отнесении ее к типовой при определении

необходимых уточнений. В этом случае ИСАПР имеет архив типовых постановок задач [45] с соответствующими наборами корректной входной информации.

2. Информация представляется в виде не данных (чисел), а знаний, т.е. характеризуется независимостью от пользователя, активностью и ситуативными связями. Это влечет за собой:

• представление и обработку информации не только в числовом, но и в символьном виде;

• переход к языкам программирования, удобным для работы с символьной информацией (Си, Лисп, Пролог);

• разработку и применение в ИСАПР способов представления информации в виде знаний (правил, фреймов, семантических сетей);

• хранение информации в виде баз знаний, частью которых могут быть БД.

3. Активность знаний выражается в том, что ИСАПР функционируют под управлением алгоритмов (процедур) и данных, т. е. диспетчирование работ подсистем в ИСАПР выполняется не с помощью внешней управляющей программы и автоматически по факту наличия или отсутствия необходимых для работы этих подсистем данных или знаний, а с помощью процедуры контроля необходимой информации. Как только эта информация появляется, автоматически срабатывают процедуры ее обработки. Таким образом, информация побуждает к действию.

4. Представление информации в виде знаний, а не чисел позволяет вести обработку данных не численными, а логическими методами. Основным математическим аппаратом ИСАПР является аппарат алгебры логики в условиях детерминированной, нечеткой или вероятностной информации. Тогда применяют язык Пролог.

5. ИСАПР ориентированы не на процедуры моделирования и анализа, а на процедуры синтеза (переход от общего к частному: декомпозиции, поиск знаний по образцу, составление документации), т.е. дедуктивного или индуктивного вывода (композиция, агрегирование, обобщение (целое из частей), формализация эвристических знаний, построение понятий).

6. Входная и выходная информация представляется с использованием принципов когнитивной психологии [приемы визуального отображения информации (иерархические меню, многооконный гра-

фический интерфейс, различные способы образного отображения действий пользователя – мультипликация)]. Это примеры графического интеллектуального интерфейса.

7 ИСАПР имеют ЭС для консультации пользователя, методики проектирования. Это и диагностика ошибок, сбоев работы ИСАПР, выдача советов, обучение.

8. ИСАПР представляют объект проектирования многоаспектно, даны связи с другими объектами, правила модификации свойств объекта, правила взаимодействия с другими объектами с целью его эквивалентного представления. Такое описание значительно облегчает многоуровневое проектирование объекта.

Количественные и качественные характеристики интеллектуальных САПР

Количественные характеристики имеют следующие величины:

• число входных, внутренних и выходных языков;

• число слов (понятий) в каждом из языков;

• глубина дерева диалога «ИСАПР – пользователь»;

• среднее время реакции ИСАПР на вопрос пользователя;

• число уровней иерархии представления объекта;

• число ЭС (или число проблем, по которым пользователь может получить консультацию);

• объем базы знаний (число правил, фреймов).

Качественные характеристики содержат следующие показатели:

– функциональные возможности (моделирование, расчет, оптимизация, параметрический и структурный синтез);

– тип структуры ИСАПР: с внешней инструментальной интеллектуализацией; с внутренней локализованной интеллектуализацией; с внутренней распределенной интеллектуализацией; с внешней специализированной интеллектуализацией;

– тип интеллектуального интерфейса: языковой (ограниченный естественный язык, предметно-ориентированный); графический (иерархическое или простое меню, графический ввод, графический вывод, многооконный интерфейс);

– способ представления знаний (правила продукций, фреймы, семантические сети, предикаты, парадигмы);

– тип логического вывода (прямой или обратный логический вывод, с поиском в глубину или ширину, комбинированный);

– аппарат логической обработки знаний (метод резолюции, логический вывод);

– тип обрабатываемых знаний (четкие, нечеткие, вероятностные).

Наряду со специфическими характеристиками ИСАПР имеются и традиционные: тип языков программирования, объем памяти, тип ОС, тип технических средств.

При *внешней интеллектуализации* (рис. 4.14, *а*) к САПР присоединяется готовая инструментальная интеллектуальная система, наполняемая знаниями. Достоинствами такой системы являются высокая скорость и простота разработки. Недостаток состоит в низкой эффективности из-за избыточности памяти и нерациональной организации работы системы.

Внешняя специализированная интеллектуализация заключается в том, что ЭС специально создается под данную САПР (рис. 4.14, *б*). Такую систему называют *гибридной.*

Развитием предыдущей системы будет *внутренняя локализованная интеллектуализация,* где имеются хорошо отработанные блоки лингвистического транслятора, логического вывода, представления знаний. Характеристики традиционной САПР значительно улучшаются, так как используются по необходимости отдельные блоки, а не весь ИИ (рис. 4.14, *в*).

При *внутренней распределенной интеллектуализации* каждый алгоритм САПР и принцип ее работы в целом основаны на использовании не данных, а знаний (рис. 4.14, *г*). На основе знаний строится ИСАПР — моделирующая (интеллектуальные аналоги традиционных САПР) и синтезирующая.

Моделирующая интеллектуальная САПР. Моделирующая ИСАПР выполняет описание объекта и основной задачи моделирования (задачи расчета, анализа, оптимизации) на предметно-ориентированном языке (рис. 4.15).

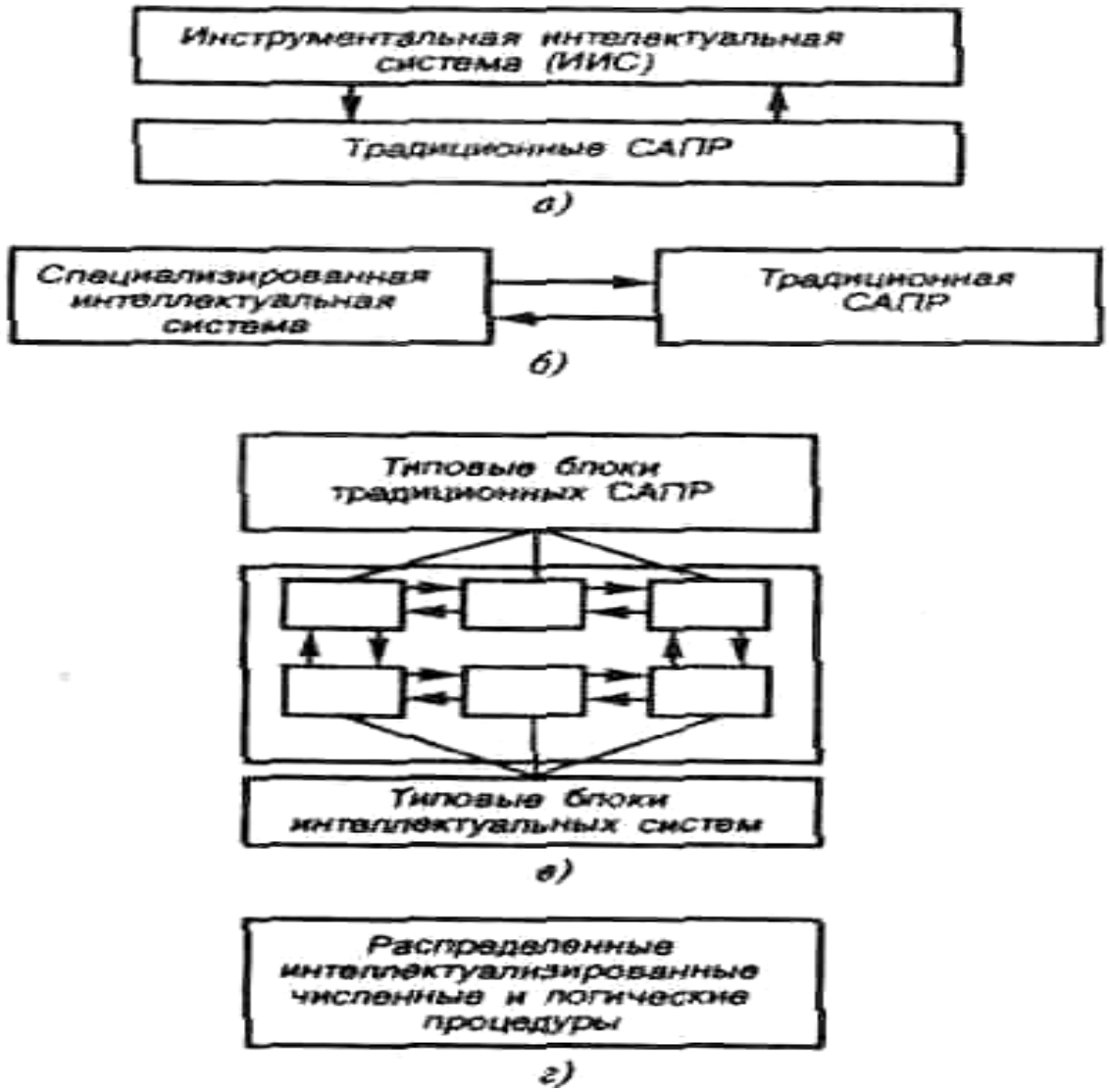


Рис. 4.14. Типы структуры САПР:
а – внешняя интеллектуализация; *б* – внешняя специализированная;
в – внутренняя локализованная; *г* – внутренняя распределенная

Структура и начальные параметры считаются известными, задача состоит в выявлении конечных параметров и характеристик заданного объекта. Интеллект моделирующей ИСАПР заключается в адаптации работы ИСАПР к особенностям объекта: при составлении математической модели объекта база знаний, используя покомпонентное описание объекта, выдает те модели его компонента, которые наиболее соответствуют требованиям точности и скорости моделирования с учетом сложности всего объекта, а база знаний инициирует в соответствии с этими требованиями способ составления математической модели объекта. Моделирование состоит в том, что база знаний на основе сведений, поступающих от «датчиков», анализирующих параметры и качество моделирующей процедуры (анализ, оптимизация, размещение, трассировка), управляет ходом моделиро-

вания — изменяет параметры моделирующей процедуры или меняет алгоритмы.

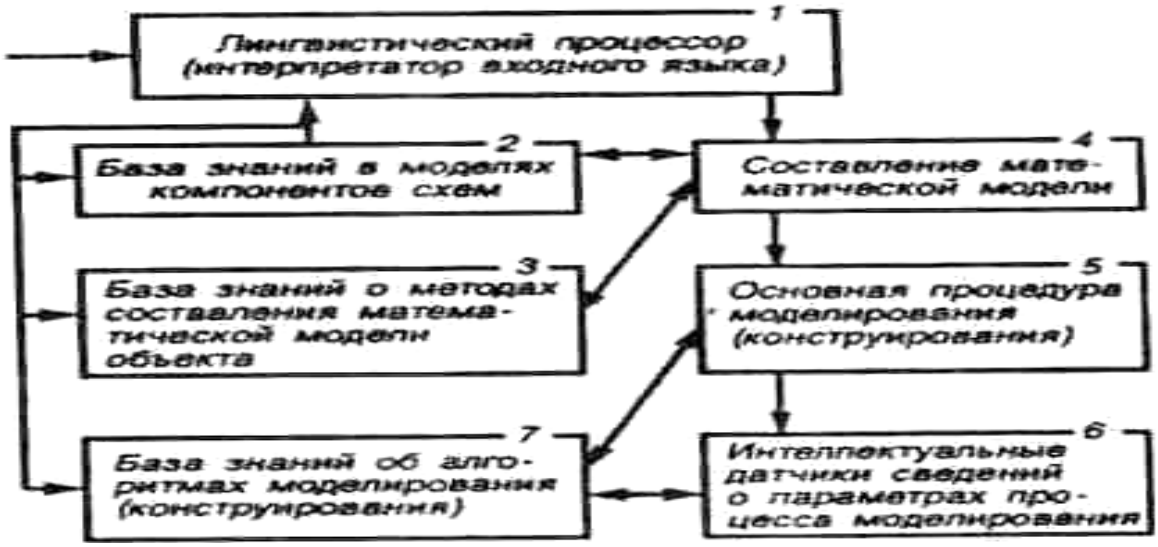


Рис. 4.15. Структура адаптивной моделирующей ИСАПР

Основным отличием баз знаний в этой ИСАПР от БД в традиционной САПР является активный характер баз знаний, так как кроме декларативных знаний (параметры компонентов, алгоритмов, оптимальные условия их применения) в этих базах хранятся и знания процедур — алгоритмы и методы, инициируемые знаниями, содержащимися в описании объекта и задачи, либо знаниями, извлекаемыми с помощью интеллектуальных датчиков из самого процесса моделирования.

В состав моделирующей адаптивной ИСАПР дополнительно включаются подсистемы диагностики, указывающие пользователю как на его ошибки (на семантическом уровне) при составлении описания объекта, так и на ошибки системы (на естественном языке).

Для проектирования вычислительных систем решается задача фрагментации объектов, т. е. моделирование по частям и агрегирования результатов моделирования.

Синтезирующая интеллектуальная САПР. Она решает задачи структурного и параметрического синтеза (рис. 4.16). Предполагается, что эти задачи можно решать отдельно и последовательно.

В блоке логического вывода может частично выполняться и параметризация синтезируемой структуры, если она имеет важное значение для функционирования структуры. Синтез в такой ИСАПР со-

стоит в построении дерева декомпозиции задачи на подзадачи (в качестве задачи может выступать объект или процесс) в блоке *2* и поиске варианта декомпозиции в блоке логического вывода *3,* удовлетворяющего заданным требованиям (блок *4*). В качестве дерева декомпозиции можно использовать И—ИЛИ дерево или дерево состояний и свойств. Вместо блоков *2* и *3* можно использовать любой эвристический прием синтеза.

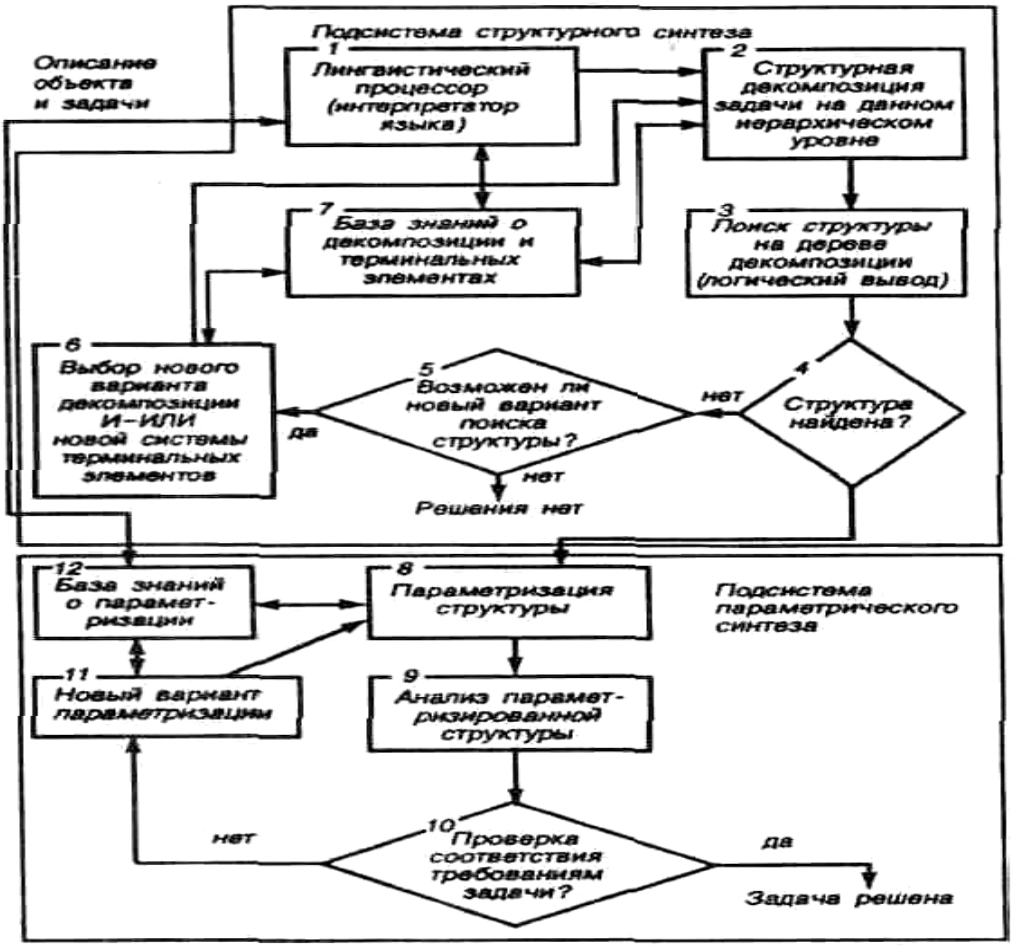


Рис. 4.16. Структура синтезирующей ИСАПР

Если на выбранном базисе терминальных элементов дерева декомпозиции структура не синтезируется, нужно сменить базис, а если синтезируется, то дальше выполняется ее параметризация. Параметризация может выполняться различными способами — либо эвристически, на основе знаний экспертов, либо формально, с помощью оптимизирующей системы, аналогичной адаптивной моделирующей ИСАПР, но с использованием в качестве основной процедуры программы оптимизации. Описанная процедура синтеза может быть применена на каждом иерархическом уровне.

4.5. Методы структурного и параметрического синтеза

Общая характеристика методов синтеза. В задачах синтеза наиболее эффективно используются методы ИИ. Особенностью этих задач является необходимость многократного принятия решений в процессе их реализации. Поскольку эти решения неоднозначны, результаты синтеза тоже неоднозначны. Существуют различные типы синтеза (рис. 4.17).

Рис. 4.17. Основные типы синтеза

Оптимизационный синтез состоит в том, что сначала разработчик сам создает начальный вариант проектируемого объекта, т. е. задает его структуру и параметры. Если после расчета и анализа математической модели этого начального варианта окажется, что характеристики объекта отличаются от требуемых, то изменяются сначала параметры, а если это не поможет, то и структура объекта в соответствии с каким-либо методом оптимизации.

Каждый шаг оптимизации требует повторного моделирования, расчета и анализа объекта, так как для оптимизации по нескольким переменным требуется обычно не менее 100 шагов, а приемлемое время оптимизации составляет не более нескольких минут. Это требование ограничивает применение оптимизационного синтеза параметров и особенно структуры объектов. Обычно оптимизационный

синтез используется при расчете параметров (классические непрерывные оптимизации). Задачи структурного синтеза решаются методами дискретной оптимизации (направленный перебор).

Генерационный синтез, в отличие от оптимизационного, состоит не в последовательном улучшении первоначального варианта объекта, а в создании (генерации) сразу модели работоспособного объекта. Выделяют две разновидности генерационного синтеза. Первая — синтез по теоретически строго обоснованным соотношениям, определяющим структуру или чаще параметры объекта проектирования (*теоретический синтез*). Эти соотношения отражают необходимые условия работоспособности объекта. Примером строгого теоретического синтеза является структурный синтез цифровых устройств (ЭВМ).

Теоретический синтез возможен при строгой формализации постановки задачи — получении заданной логической функции на выходе цифрового устройства. Но такие случаи редки, и формулировки задач синтеза трудны из-за разнообразия и сложности математизации. Поэтому трудно иметь строгий общий теоретический аппарат и приходится прибегать к прошлому опыту, выраженному в виде методик, отношений, зависимостей, просто к здравому смыслу и творчеству.

Вторая разновидность генерационного синтеза — *эвристический синтез*. Большинство задач (технологических) вычислительных процессов решается именно этим способом.

В целом, если задача синтеза формулируется как задача поиска параметризированной структуры, то ее решение можно искать на путях теоретического синтеза (хотя результаты могут быть неточными). Например, преобразование множества X во множество Y или сигнала X в сигнал Y. Если таким образом задачу сформулировать нельзя, то прибегают к эвристическому синтезу.

Различия методов теоретического и эвристического синтеза – следующие:

• теоретический синтез реализуется с помощью алгоритмов, обладающих свойством определенности, т. е. его результат однозначен и не зависит от точности исполнителя;

• эвристический синтез допускает множество альтернативных решений, так как в нем важную роль играет выбор исполнителем того или иного способа синтеза, включая выбор типа элементов, из которых должен состоять объект, способов их соединения и расчета их параметров.

Методы структурного синтеза. Синтез ВС и вычислительных процессов (ВП) включает три этапа:

- выбор типовых элементов структуры (синтез базиса);
- построение структуры (структурный синтез);
- параметризация элементов структуры (параметрический синтез).

Выбор типовых элементов структуры выполняется эмпирически. Для формализации используют матрицу «элементы — свойства» (или, что то же самое, матрицу «средства — цели»). Строкам этой матрицы соответствуют различные альтернативные элементы, выполняющие одни и те же функции, а столбцам — свойства этих элементов, важные для проектируемой ВС (ВП). В каждой клетке a_{ij} матрицы по балльной системе проставляется экспертная оценка i-го свойства у i-го элемента. Анализ этой матрицы проектировщиком позволит составить наглядное и достаточно полное представление о приемлемости того или иного элемента.

Структурный синтез редко бывает полностью независим от параметрического, обычно выбор структуры и определение значений параметров ее элементов тесно связано, составляя вместе структурно-параметрический синтез. Тем не менее структурный синтез можно выполнять независимо от параметрического в тех случаях, когда формирование определенной структуры объекта является необходимым условием его работоспособности, а параметры лишь обеспечивают более высокое качество функционирования. К таким случаям относятся, например, структурный синтез цифровых устройств, вычислительных процессов.

Используют следующие типовые приемы структурного синтеза.

1. *Выбор из готовых структур-прототипов.*

Этот прием предполагает наличие библиотеки готовых структур. Основной недостаток — необходимость прямого перебора всех структур. Если подобрать полностью подходящую структуру не удается, выбирают наиболее близкую и модифицируют ее под заданные требования путем удаления или добавления новых элементов, введения дополнительных или исключения ненужных связей и т. д.

2. *Построение частной структуры из общей.*

В данном случае сначала создают структуру с максимальной избыточностью, являющуюся обобщением всех известных структур объекта данного типа. Нужная структура синтезируется путем удаления лишних элементов и связей обобщенной структуры. Этот прием

традиционно используется при синтезе вычислительных процессов, когда обобщенный процесс представляет собой цепочку всех вычислительных операций, применяемых для изготовления объекта данного класса, а синтез состоит в выборе из этой цепочки только тех операций, которые нужны в каждом конкретном случае. Так как выбор выполняется самим разработчиком на основе своего опыта, этот прием, как и предыдущий, относится к методам эмпирического синтеза.

3. *Направленный поиск по И—ИЛИ дереву.*

Данный прием можно рассматривать как специальный случай построения частной структуры из общей. В качестве общей структуры выступает заранее составленное И—ИЛИ дерево (рис. 4.18), в котором каждая группа путей от корневой вершины через вершины И, ИЛИ до терминальных вершин соответствует одной частной структуре.

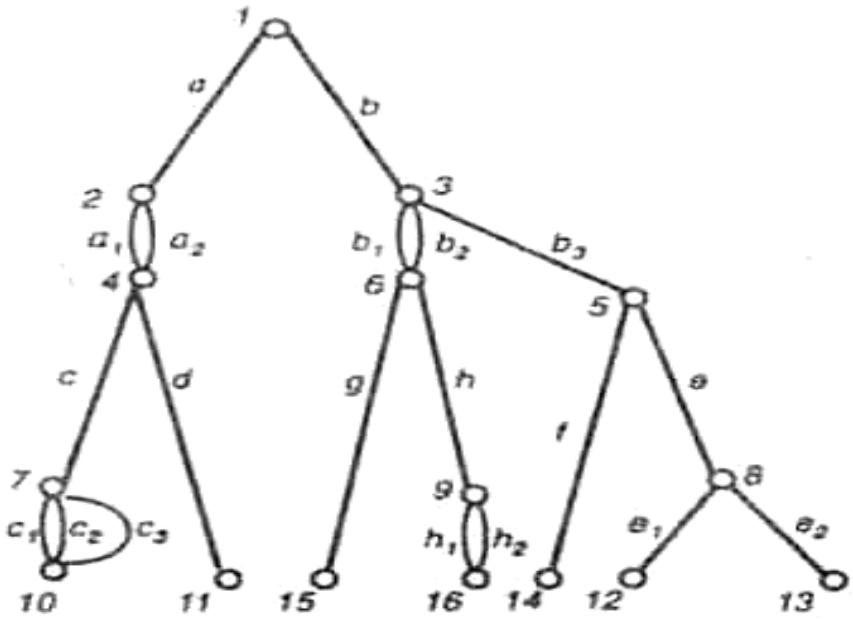


Рис. 4.18. Структура И–ИЛИ дерева

Синтез по И–ИЛИ дереву удобно применять в тех случаях, когда объект легко декомпозируется на составляющие его части, которые декомпозируются на еще более мелкие и т. д., каждая декомпозиция порождает вершину типа И, а каждый уровень декомпозиции — ярус дерева из вершин И. Альтернативные варианты реализации каждой части объекта порождают вершину типа ИЛИ. Ветви дерева, выходящие из вершин ИЛИ, – имена способов реализации этих частей или их свойств, играющих определенную роль при синтезе.

В качестве примера на рис. 4.18 приведено дерево И–ИЛИ для некоторого абстрактного объекта, который декомпозируется на части *a, b,* затем каждая из них соответственно на части *c, d* и *e, f, g, h.*

Если дерево состоит только из вершин И, то оно описывает одну структуру объекта. Альтернативный выбор структуры определяется вершинами ИЛИ. Обход дерева из вершин ИЛИ возможен либо в глубину, либо в ширину. При этом с каждой альтернативной реализацией (a_1, или a_2, c_1, c_2 или c_3 и т. д.) связывается оператор перехода, разрешающий переход в следующую вершину И только при выполнении определенных ограничений в вершине ИЛИ. Ограничения могут быть глобальными, относящимися ко всему объекту (общая масса или общая стоимость ВС, общее время вычислительного процесса и т. д.), и локальными, относящимися к данной реализуемой части объекта (условия стыковки с другими ЭВМ, ВС; ограничения на параметры части ВС).

Условием успешного окончания синтеза является прохождение по всем вершинам И до терминальных вершин. Практически строить дерево И–ИЛИ не обязательно, достаточно иметь дерево декомпозиции И, в каждой вершине которого нужно программно проверять альтернативные возможности реализации, задаваемые списком этих реализаций и списком ограничений. Проверка этих ограничений означает, что синтез носит структурно-параметрический характер, поскольку в процессе синтеза отбираются элементы с определенными значениями параметров.

4. *Направленный поиск по дереву состояний и свойств.*

Структурно-параметрический синтез можно выполнить не только на основе И–ИЛИ дерева, но и с помощью дерева состояний и свойств (рис. 4.19).

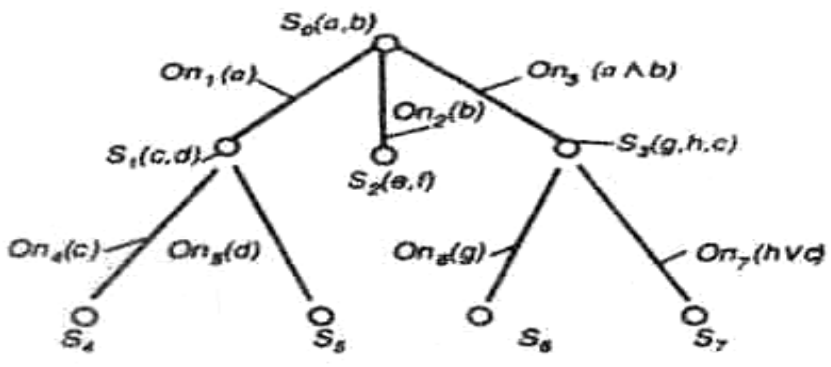


Рис. 4.19. Направленный поиск по дереву состояний и свойств

Каждая вершина $S_i(a_1 \ldots a_n)$ в этом дереве трактуется как задача, возможное решение которой обладает свойствами $a_1 \ldots a_n$, а каждый оператор перехода $On_i(a)$ имеет смысл: *«Если* решение задачи S_{i-1} должно обладать свойством *a,* то перейти к решению задачи *S*». Таким образом, нужно решить задачу S_3. Пусть известно, что решение должно одновременно удовлетворять свойствам *(a, b, h, c)*. Нужно найти терминальную вершину, имеющую указанные свойства.

Приведенный пример позволяет иллюстрировать некоторые характерные особенности обработки знаний в ИИ при использовании дерева состояний:

1) *Наследование свойств.*

Из рис. 4.19 видно, что состояние $S_1(c, d)$ на самом деле, кроме свойств *c*, *d*, должно также обладать и свойством *d* предыдущей вершины S_0, состояние $S_2(e, f)$ — свойством *b.* Обеспечиваемое последовательным прохождением вершин при поиске по дереву передача свойств отцовской вершины к дочерним называется в ИИ *наследованием свойств.* Механизм автоматического наследования свойств позволяет упростить запись и реализацию операторов перехода, не учитывая в них свойств предыдущих состояний. Он «по умолчанию» обеспечивает присутствие в последующих состояниях свойств предыдущих.

Однако наследование свойств в дереве поиска необязательно, и его можно отменить, если условиться, что «отцовское» и «дочернее» состояния по своим свойствам друг с другом не связаны, например, относятся к понятиям с разной семантикой (S_0 – рабочий, S_1 – станок). Если же состояния в дереве относятся к одному классу, то механизм наследования свойств позволяет «наращивать» свойства состояний в процессе последовательного раскрытия вершин дерева. Это особенно важно при решении задач классификации и диагностики.

2) *Наследование условий перехода.*

По аналогии с наследованием свойств состояний можно говорить о наследовании условий перехода в операторах перехода *On*. Это позволяет объяснить результат поиска. Например, переход к состоянию S_7 вызван наличием свойства $b \wedge c$ в S_3 и одновременно свойства $a \wedge b$ в S_0, что легко установить, анализируя условия перехода в On_7 и On_3. Обычно процесс последовательного применения операторов перехода к раскрытию вершин дерева состояний запоминается в программе поиска и может быть легко воспроизведен в виде

трассы поиска, вследствие чего процедуру объяснения результатов часто называют *трассировкой поиска*.

3) *Поиск правил по образцу.*

Этот часто используемый в работах по ИИ термин означает, что левая часть правила «если S_{i-1} обладает свойством a, то переход к S_i» сравнивается с образцами, т.е. с перечнем фактически имеющихся имен состояний и их свойств. Совпадение левой части правила с каким-либо образцом вызывает активизацию правила, т.е. срабатывание его правой части.

Кроме описанных существует большое число других методов синтеза – метод морфологического ящика, весьма близкий к поиску по дереву состояний и И–ИЛИ дереву, методы «мозгового штурма» и контрольных вопросов, относящиеся к методам решения изобретательских задач, однако в САПР они не получили распространения из-за сложности их формализации.

Параметрический синтез. После синтеза структуры (если она не сопровождалась синтезом параметров) выполняется параметрический синтез. Вариант синтеза путем решения задачи оптимизации широко описан в литературе, поэтому рассмотрим генерационный параметрический синтез, т. е. расчет параметров по системе формул или уравнений без оптимизации, обеспечивающей получение работоспособного варианта объекта.

Задача ставится таким образом. Пусть функционирование ВС

обеспечивается при выполнении системы соотношений (формул)

$$Y_i = f_i(x_1 \ldots x_m),\ i = 1, m,$$

где $x_1, \ldots, x_m$ — параметры ВС.

Необходимо определить последовательность применения формул расчета при задании некоторой части параметров, т. е. нужно синтезировать методику расчета по известным формулам.

Решение состоит в определении такой цепочки причинно-следственной связи в формулах (ранжирование формул), в которой расчет по первой формуле в этой цепочке позволил бы найти величины, необходимые для расчета по второй формуле и т. д. По существу данная задача состоит в планировании вычислений. Алгоритмы и программы, решающие эту задачу, в теории ИИ получили название *концептуальных решателей или планировщиков.*

Суть алгоритма планирования заключается в следующем. Найдем сначала формулу с минимальным числом неизвестных, например с одной неизвестной, и вычислим ее. Далее снова найдем новую формулу с новой неизвестной и также определим ее и т. д. Легко заметить, что если построить матрицу, столбцы которой соответствуют неизвестным *m* (часть из них задана), а строки — номерам формул, то при отыскании искомой последовательности расчета эта матрица будет иметь строго трапециевидную структуру:

а) расчет только по формулам;

б) расчет с решением системы уравнений относительно X_6, X_7. На рис. 4.20 показаны случаи: *а* –переопределения (число формул больше неизвестных); *б* –недоопределения; 1, 7, 6, 4, 3 – номера формул, когда X_1 рассчитывается по формуле на основе известных X_2, X_5, а для расчета X_6, X_7 нужно решить систему из двух уравнений, после чего X_3, X_4 опять можно рассчитать по формулам.

Для автоматизации планирования можно каждую формулу заменить фреймовой системой инструкций. Например, для формулы $X_1 = f_1(X_2, X_5)$ можно записать:

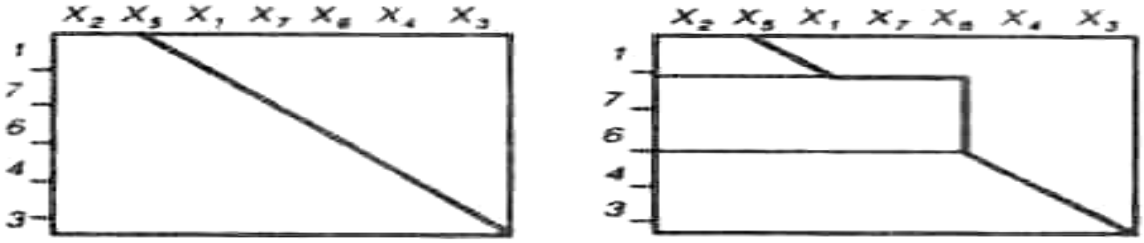
– если известны X_5, X_2, то известно X_1;

– если известны X_1, X_2, то известно X_5;

– если известны X_5, X_1, то известно X_2;

– если известны X_5, X_2, X_1, то неизвестных нет;

– если известны X_1, то неизвестны X_2, X_5 и т. д.



а *б*

Рис. 4.20 **.** Построение матрицы по алгоритму планирования

Представляя каждую формулу в указанном виде (этот процесс легко автоматизировать, записывая и вводя в программу через дисплей лишь исходную формулу) и последовательно многократно просматривая их, можно определять порядок активизации каждой формулы, т. е. последовательность их использования в синтезируемой методике расчета. Но данный формализм правил соответствует фор-

мированию семантической вычислительной сети. Однако возможны и другие алгоритмы прямого определения структуры матриц, не связанные с построением семантических сетей (рис. 4.21).

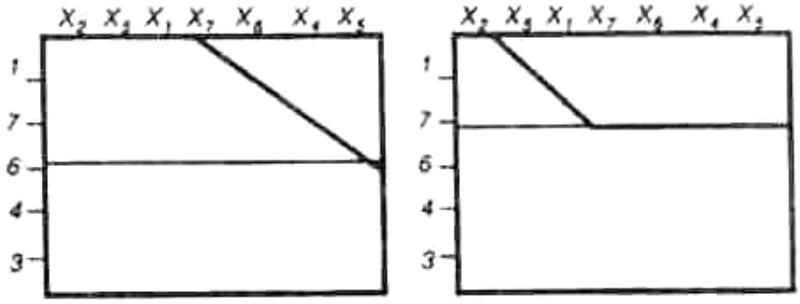


Рис. 4.21. Алгоритм прямого определения структуры матрицы

4.6. Нейрокомпьютинг и его особенности

Нейрокомпьютинг – это научное направление, занимающееся разработкой вычислительных систем – нейрокомпьютеров, которые состоят из большого числа параллельно работающих простых вычислительных элементов (нейронов).

В отличие от цифровых систем, представляющих собой комбинации процессорных и запоминающих блоков, нейропроцессоры содержат память, распределённую в связях между очень простыми процессорами, которые часто могут быть описаны как формальные нейроны или блоки из однотипных формальных нейронов. Тем самым основная нагрузка на выполнение конкретных функций процессорами ложится на архитектуру системы, детали которой в свою очередь определяются межнейронными связями. Подход, основанный на представлении как памяти данных, так и алгоритмов системой связей (и их весами), называется *коннекционизм.*

Три основных преимущества нейрокомпьютеров:

- Все алгоритмы нейроинформатики высокопараллельны, а это уже залог высокого быстродействия.
- Нейросистемы можно легко сделать очень устойчивыми к помехам и разрушениям.
- Устойчивые и надёжные нейросистемы могут создаваться и из ненадёжных элементов, имеющих значительный разброс параметров.

Разработчики нейрокомпьютеров стремятся объединить устойчивость, быстродействие и параллелизм аналоговых вычислительных машин с универсальностью современных компьютеров.

Биологический нейрон и его математическая модель

Искусственный нейрон (математический нейрон Маккалока — Питтса, формальный нейрон) — узел искусственной нейронной сети, являющийся упрощённой моделью естественного нейрона. Математически искусственный нейрон представляют как некоторую нелинейную функцию от единственного аргумента — линейной комбинации всех входных сигналов. Данную функцию называют функцией активации или функцией срабатывания, передаточной функцией [2]. Полученный результат посылается на единственный выход. Такие искусственные нейроны объединяют в сети — соединяют выходы одних нейронов со входами других. Искусственные нейроны и сети являются основными элементами идеального нейрокомпьютера.

Математически нейрон представляет собой взвешенный сумматор, единственный выход которого определяется через его входы и матрицу весов следующим образом:

$$u = \sum_{i=1}^{n} w_i x_i + w_0 x_0 \text{, } y = f(u)$$

где x_i и w_i — соответственно сигналы на входах нейрона и веса входов;

u – индуцированное локальное поле;

f(u) — передаточная функция.

Возможные значения сигналов на входах нейрона считают заданными в интервале $[0, 1]$. Они могут быть либо дискретными (0 или 1), либо аналоговыми. Дополнительный вход x_0 и соответствующий ему вес w_0 используются для инициализации нейрона. Под *инициализацией* подразумевается смещение активационной функции нейрона по горизонтальной оси, т.е. формирование порога чувствительности нейрона [5]. Кроме того, иногда к выходу нейрона специально добавляют некую случайную величину, называемую сдвигом. Сдвиг можно рассматривать как сигнал на дополнительном, всегда нагруженном синапсе.

Множество математических моделей нейрона может быть построено на базе простой концепции строения нейрона (рис. 4.22). Так называемая суммирующая функция объединяет все входные сигналы X_i, которые поступают от нейронов-отправителей. Значением такого объединения является взвешенная сумма, где веса w_i представляют собой синаптические мощности. Возбуждающие синапсы имеют положительные веса, а тормозящие синапсы –

отрицательные веса. Для выражения нижнего уровня активации нейрона к взвешенной сумме прибавляется компенсация (смещение) Q.

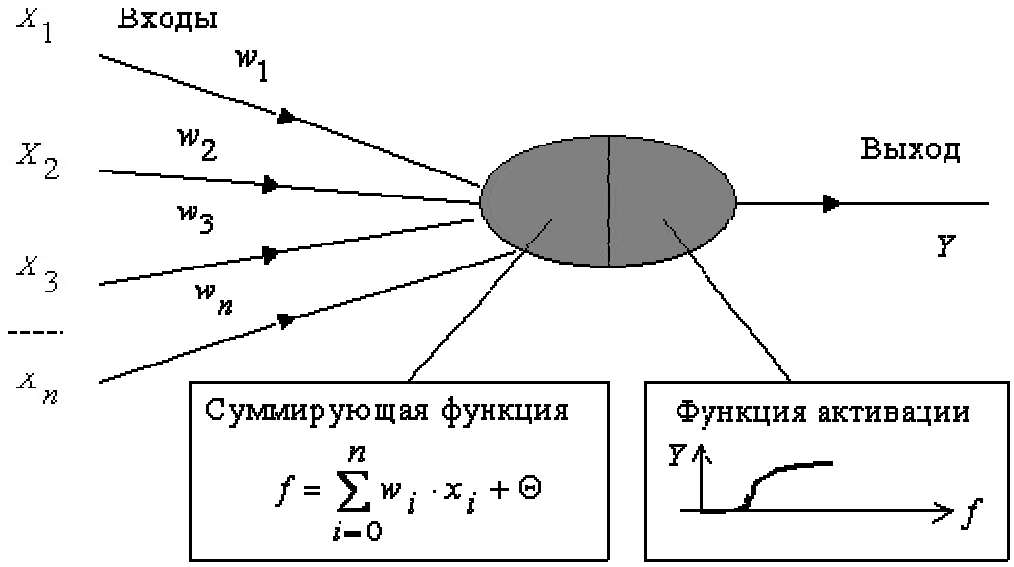
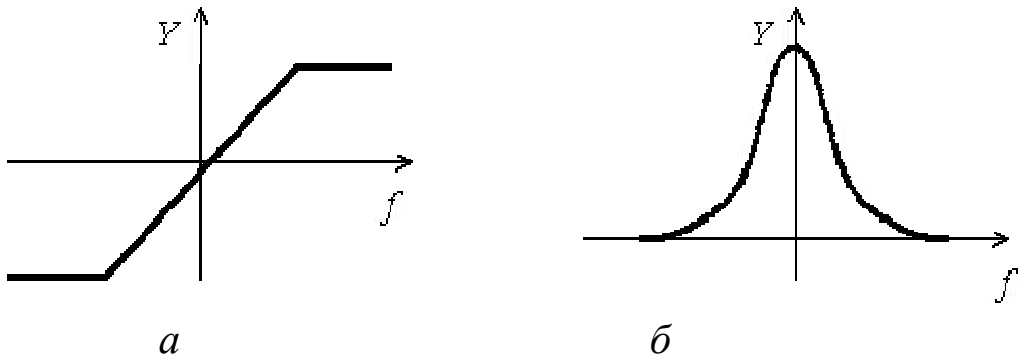


Рис. 4.22. Простая математическая модель нейрона

Так называемая функция активации рассчитывает выходной сигнал нейрона Y по уровню активности f. Функция активации обычно является сигмоидной, как показано в правой нижней рамке на рис. 4.22. Другими возможными видами функций активации являются линейная и радиально-симметричная функции (рис. 4.23).



а *б*

Рис. 4.23. Функции активации нейронов:
а – линейная; *б* – радиально-симметрическая

Персептрон Розенблатта

Одной из первых искусственных сетей, способных к перцепции (восприятию) и формированию реакции на воспринятый стимул, явился персептрон Розенблатта (1957). Персептрон рассматривался его автором не как конкретное техническое вычислительное устрой-

ство, а как модель работы мозга. Нужно заметить, что после нескольких десятилетий исследований современные работы по искусственным нейронным сетям редко преследуют такую цель.

Простейший классический персептрон содержит нейроподобные элементы трех типов (рис. 4.24), назначение которых в целом соответствует нейронам рефлекторной нейронной сети. *S*-элементы формируют сетчатку сенсорных клеток, принимающих двоичные сигналы от внешнего мира. Далее сигналы поступают в слой ассоциативных или *A*-элементов (для упрощения изображения часть связей от входных *S*-клеток к *A*-клеткам не показана). Только ассоциативные элементы, представляющие собой формальные нейроны, выполняют нелинейную обработку информации и имеют изменяемые веса связей. *R*-элементы с фиксированными весами формируют сигнал реакции персептрона на входной стимул.

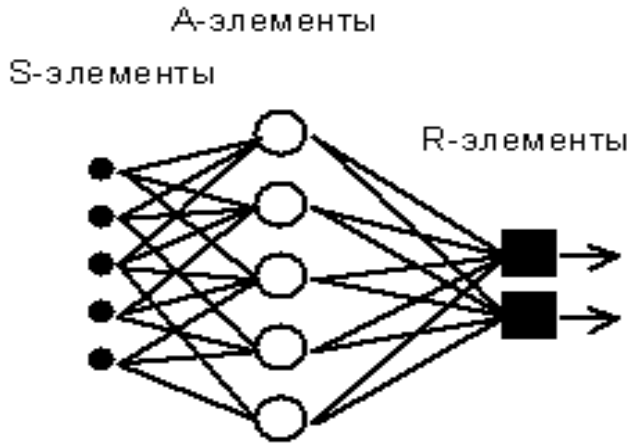


Рис. 4.24. Элементарный персептрон Розенблатта

Ф. Розенблатт называл такую нейронную сеть трехслойной, однако по современной терминологии представленная сеть называется *однослойной*, так как имеет только один слой нейропроцессорных элементов. Однослойный персептрон характеризуется матрицей синаптических связей *W* от *S*- к *A*-элементам. Элемент матрицы W_{ij} отвечает связи, ведущей от *i*-го *S*-элемента к *j*-му *A*-элементу.

В Корнельской авиационной лаборатории была разработана электротехническая модель персептрона MARK-1, которая содержала восемь выходных *R*-элементов и 512 *A*-элементов, которые можно было соединять в различных комбинациях. На этом персептроне была проведена серия экспериментов по распознаванию букв алфавита и геометрических образов.

В работах Розенблатта было сделано заключение о том, что нейронная сеть рассмотренной архитектуры будет способна к воспроизведению любой логической функции, однако, как было показано позднее американскими учеными М. Минским и С. Пейпертом (1969), этот вывод оказался неточным. Были выявлены принципиальные неустранимые ограничения однослойных персептронов, и в последствии стал в основном рассматриваться многослойный вариант персептрона, в котором имеется несколько слоев процессорных элементов.

С сегодняшних позиций однослойный персептрон представляет скорее исторический интерес, однако на его примере могут быть изучены основные понятия и простые алгоритмы обучения нейронных сетей.

Теорема об обучении персептрона

Обучение сети состоит в подстройке весовых коэффициентов каждого нейрона. Пусть имеется набор пар векторов (x^α, y^α), $\alpha = 1 \ldots p$, называемый *обучающей выборкой*. Будем называть нейронную сеть обученной на данной обучающей выборке, если при подаче на входы сети каждого вектора x^α на выходах всякий раз получается соответствующий вектор y^α.

Предложенный Ф. Розенблаттом метод обучения состоит в итерационной подстройке матрицы весов, последовательно уменьшающей ошибку в выходных векторах. Алгоритм включает несколько шагов:

Шаг 0. Начальные значения весов всех нейронов $W(t = 0)$ полагаются случайными.

Шаг 1. Сети предъявляется входной образ x^α, в результате формируется выходной образ $\mathbf{y}^\alpha \neq \mathbf{y}^\alpha$.

Шаг 2. Вычисляется вектор ошибки $\delta^\alpha = \left(\mathbf{y}^\alpha - \mathbf{y}^\alpha\right)$, делаемой сетью на выходе. Дальнейшая идея состоит в том, что изменение вектора весовых коэффициентов в области малых ошибок должно быть пропорционально ошибке на выходе и равно нулю, если ошибка равна нулю.

Шаг 3. Вектор весов модифицируется по следующей формуле: $W(t + \Delta t) = W(t) + \eta x^a (\delta^a)^T$. Здесь $0 < \eta < 1$ – темп обучения.

Шаг 4. Шаги 1–3 повторяются для всех обучающих векторов. Один цикл последовательного предъявления всей выборки называется *эпохой*. Обучение завершается по истечении нескольких эпох:

а) когда итерации сойдутся, т.е. вектор весов перестает изменяться;

б) когда полная просуммированная по всем векторам абсолютная ошибка станет меньше некоторого малого значения.

Используемая на шаге 3 формула учитывает следующие обстоятельства:

а) модифицируются только компоненты матрицы весов, отвечающие ненулевым значениям входов;

б) знак приращения веса соответствует знаку ошибки, т.е. положительная ошибка ($\delta > 0$, значение выхода меньше требуемого) приводит к усилению связи;

в) обучение каждого нейрона происходит независимо от обучения остальных нейронов, что соответствует важному с биологической точки зрения принципу *локальности* обучения.

Данный метод обучения был назван Ф. Розенблаттом «*методом коррекции с обратной передачей сигнала ошибки*». Позднее более широко стало известно название «δ-правило». Представленный алгоритм относится к широкому классу алгоритмов обучения с учителем, поскольку известны как входные векторы, так и требуемые значения выходных векторов (имеется учитель, способный оценить правильность ответа ученика).

Доказанная Розенблаттом теорема о сходимости обучения по δ-правилу говорит о том, что персептрон способен обучиться любому обучающему набору, который он способен представить.

Понятие линейной разделимости и персептронной представляемости. Каждый нейрон персептрона является формальным пороговым элементом, принимающим единичные значения в случае, если суммарный взвешенный вход больше некоторого порогового значения:

$$y_j = \begin{cases} 1, & \sum_i W_{ij} x_i > \theta_j \\ 0, & \sum_i W_{ij} x_i \le \theta_j \end{cases}$$

Таким образом, при заданных значениях весов и порогов нейрон имеет определенное значение выходной активности для каждого возможного вектора входов. Множество входных векторов, при которых нейрон активен ($y = 1$), отделено от множества векторов, на которых нейрон пассивен ($y = 0$) *гиперплоскостью*, уравнение которой таково:

$$\sum_i W_{ij} x_i - \theta_j = 0$$

Следовательно, нейрон способен отделить (иметь различный выход) только такие два множества векторов входов, для которых имеется гиперплоскость, отсекающая одно множество от другого. Такие множества называют *линейно разделимыми*. Проиллюстрируем это понятие на примере.

Пусть имеется нейрон, для которого входной вектор содержит только две булевые компоненты (X_1, X_2), определяющие плоскость (рис. 4.25). На данной плоскости возможные значения векторов отвечают вершинам единичного квадрата. В каждой вершине определено требуемое значение активности нейрона 0 (белая точка) или 1 (черная точка). Требуется определить, существует ли такой набор весов и порогов нейрона, при котором этот нейрон сможет отделить точки разного цвета?

На рис 4.25 представлена одна из ситуаций, когда этого сделать нельзя вследствие линейной неразделимости множеств белых и черных точек.

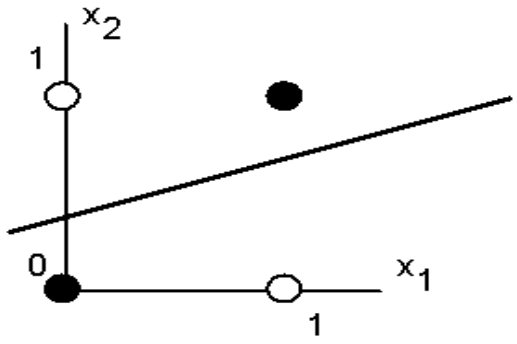


Рис. 4.25. Белые точки не могут быть отделены одной прямой от черных

Требуемая активность нейрона для этого рисунка определяется табл. 4.2, в которой нетрудно узнать задание логической функции «исключающее ИЛИ».

Таблица 4.2

Активность нейрона

X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	0

Линейная неразделимость множества аргументов, отвечающих различным значениям функции, означает, что функция «исключающее ИЛИ», столь широко использующаяся в логических устройствах, не может быть представлена формальным нейроном. Столь скромные возможности нейрона и послужили основой для критики персептронного направления Ф. Розенблатта со стороны М. Минского и С. Пейперта.

При возрастании числа аргументов ситуация еще более катастрофична: относительное число функций, которые обладают свойством линейной разделимости, резко уменьшается. А значит, и резко сужается класс функций, который может быть реализован персептроном (так называемый класс функций, обладающий свойством персептронной представляемости). Соответствующие данные приведены в табл. 4.3.

Таблица 4.3

Зависимость число возможных логических функций от числа переменных

Число переменных N	Полное число возможных логических функций $(= 2^{2^N})$	Из них линейно разделимых функций
1	4	4
2	16	14
3	256	104
4	65536	1882
5	> 1000000000	94572

Видно, что однослойный персептрон крайне ограничен в своих возможностях точно представить наперед заданную логическую функцию. Позднее, в начале 70-х годов, это ограничение было преодолено путем введения нескольких слоев нейронов, однако критическое отношение к классическому персептрону сильно заморозило

общий круг интереса и научных исследований в области искусственных нейронных сетей.

Задача обучения нейронной сети на примерах. Классификация и категоризация

Задача обучения нейронной сети на примерах. По своей организации и функциональному назначению искусственная нейронная сеть с несколькими входами и выходами выполняет некоторое преобразование входных стимулов – сенсорной информации о внешнем мире – в выходные управляющие сигналы. Число преобразуемых стимулов равно n – числу входов сети, а число выходных сигналов соответствует числу выходов m. Совокупность всевозможных входных векторов размерности n образует векторное пространство X, которое будем называть *признаковым пространством.*

При рассмотрении соответствующих пространств предполагается использование обычных векторных операций сложения и умножения на скаляр. Аналогично выходные векторы также формируют признаковое пространство, которое будет обозначаться Y. Теперь нейронную сеть можно мыслить как некоторую многомерную функцию $F: X \to Y$, аргумент которой принадлежит признаковому пространству входов, а значение – выходному признаковому пространству.

При произвольном значении синаптических весовых коэффициентов нейронов сети функция, реализуемая сетью, также произвольна.

Для получения требуемой функции необходим специфический выбор весов. Упорядоченная совокупность всех весовых коэффициентов всех нейронов может быть представлена как вектор W. Множество всех таких векторов также формирует векторное пространство, называемое *пространством состояний* или *конфигурационным (фазовым) пространством W*. Термин «фазовое пространство» пришел из статистической физики систем многих частиц, где под ним понимается совокупность координат и импульсов всех частиц, составляющих систему.

Задание вектора в конфигурационном пространстве полностью определяет все синаптические веса и тем самым состояние сети. Состояние, при котором нейронная сеть выполняет требуемую функцию, называют *обученным состоянием* сети W^*. Отметим, что для заданной функции обученное состояние может не существовать

или быть не единственным. Задача обучения теперь формально эквивалентна построению процесса перехода в конфигурационном пространстве от некоторого произвольного состояния W^0 к обученному состоянию.

Требуемая функция однозначно описывается путем задания соответствия каждому вектору признакового пространства X некоторого вектора из пространства Y. В случае сети из одного нейрона в задаче детектирования границы полное описание требуемой функции достигается заданием всего четырех пар векторов. Однако в общем случае, как, например, при работе с видеоизображением, признаковые пространства могут иметь высокую размерность, поэтому даже в случае булевых векторов однозначное определение функции становится весьма громоздким (при условии, конечно, если функция не задана явно, например формулой; однако для явно заданных функций обычно не возникает потребности представлять их нейросетевыми моделями).

Во многих практических случаях значения требуемых функций для заданных значений аргумента получаются из эксперимента или наблюдений и, следовательно, известны лишь для ограниченной совокупности векторов. Кроме того, известные значения функции могут содержать погрешности, а отдельные данные могут даже частично противоречить друг другу. По этим причинам перед нейронной сетью обычно ставится задача *приближенного представления функции по имеющимся примерам*.

Имеющиеся в распоряжении исследователя примеры соответствий между векторами либо специально отобранные из всех примеров наиболее представительные данные называют *обучающей выборкой*. Обучающая выборка определяется заданием пар векторов, причем в каждой паре один вектор соответствует стимулу, а второй – требуемой реакции. Обучение нейронной сети состоит в приведении всех векторов стимулов из обучающей выборки требуемым реакциям путем выбора весовых коэффициентов нейронов.

Общая проблема кибернетики, заключающаяся в построении искусственной системы с заданным функциональным поведением, в контексте нейроных сетей понимается как задача *синтеза* требуемой искусственной сети. Она может включать в себя следующие подзадачи:

1) выбор существенных для решаемой задачи признаков и формирование признаковых пространств;

2) выбор или разработка архитектуры нейронной сети, адекватной решаемой задаче;

3) получение обучающей выборки из наиболее представительных, по мнению эксперта, векторов признаковых пространств;

4) обучение нейронной сети на обучающей выборке.

Следует отметить, что подзадачи 1–3 во многом требуют экспертного опыта работы с нейронными сетями, и здесь нет исчерпывающих формальных рекомендаций.

Классификация и категоризация. В случае, когда выходное признаковое пространство представляет собой дискретный перечень из двух или более групп данных, задачей нейронной сети является отнесение входных векторов к одной из этих групп. В этом случае говорят, что нейросетевая система выполняет *классификацию* или *категоризацию* данных.

Эти две интеллектуальные задачи, по-видимому, следует отличать друг от друга. Термин *класс* можно определить как совокупность предметов или понятий (образов), выделенных и сгруппированных по определенным признакам или правилам. Под *классификацией* будем понимать отнесение некоторого образа к классу, выполняемое по этим формальным правилам по совокупности признаков. *Категория* же (если отвлечься от специфического философского характера этого понятия) определяет лишь некоторые общие свойства образов и связи между ними.

Задача *категоризации*, т.е. определения отношения данного образа к некоторой категории, гораздо менее определена, чем задача отношения к классу. Границы различных категорий являются нечеткими, расплывчатыми, и обычно сама категория понимается не через формальное определение, а только в сравнении с другими категориями. Границы классов, напротив, определены достаточно точно – образ относится к данному классу, если известно, что он обладает необходимым числом признаков, характерных для этого класса.

Итак, задача систем-классификаторов – это установление принадлежности образа к одному из формально определенных классов. Примерами такой задачи являются задача классификации растений в ботанике, классификация химических веществ по их свойствам и типам возможных реакций, в которые они вступают, и др. Формальные признаки могут быть определены посредством правил типа «если…, то…», а системы, оперирующие с такими

правилами, получили название *экспертных систем*. Традиционной областью применения классификаторов на нейронных сетях является экспериментальная физика высоких энергий, где одной из актуальных задач выступает выделение среди множества зарегистрированных в эксперименте событий с элементарными частицами событий, представляющих интерес для данного эксперимента.

Проблема категоризации находится на ступеньку выше по сложности в сравнении с классификацией. Особенность ее заключается в том, что помимо отнесения образа к какой-либо группе требуется определить сами эти группы, т.е. сформировать категории.

При обучении с учителем (например, в персептроне) формирование категорий происходит методом проб и ошибок на основе примеров с известными ответами, представляемыми экспертом. Формирование категорий напоминает процесс обучения у живых организмов, поэтому эксперта называют «супервизором» или учителем. Учитель управляет обучением при помощи изменения параметров связей и реже – самой топологии сети.

Задача системы-категоризатора состоит в формировании обобщающих признаков в совокупности примеров. При увеличении числа примеров несущественные, случайные признаки сглаживаются, а часто встречающиеся – усиливаются, при этом происходит постепенное уточнение границ категорий. Хорошо обученная нейросетевая система способна извлекать признаки из новых примеров, ранее неизвестных системе, и принимать на их основе приемлемые решения.

Важно отметить различие в характере неявных «знаний», запомненных искусственной нейронной сетью, и явных, формальных «знаний», заложенных в экспертных системах (табл. 4.4).

Таблица 4.4

Сравнение экспертных и нейросетевых систем

Параметр	Экспертные системы (ЭкС)	Нейросетевые системы (НС)
1	2	3
Источник знаний	Формализованный опыт эксперта, выраженный в виде логических утверждений – правил и фактов, безусловно принимаемых системой	Совокупный опыт эксперта-учителя, отбирающего примеры для обучения + индивидуальный опыт обучающейся на этих примерах нейронной сети

Окончание табл. 4.4

1	2	3
Характер знаний	Формально-логическое «левополушарное» знание в виде правил	Ассоциативное «правополушарное» знание в виде связей между нейронами сети
Развитие знаний	В форме расширения совокупности правил и фактов (базы знаний)	В форме дообучения на дополнительной последовательности примеров, с уточнением границ категорий и формированием новых категорий
Роль эксперта	Задает на основе правил полный объем знаний экспертной системы	Отбирает характерные примеры, не формулируя специально обоснование своего выбора
Роль искусственной системы	Поиск цепочки фактов и правил для доказательства суждения	Формирование индивидуального опыта в форме категорий, получаемых на основе примеров, и категоризация образов

Различия в характере экспертных и нейросетевых систем обуславливают и различия в сферах их применения. Экспертные системы применяются в узких предметных областях с хорошо структурированными знаниями, например, в классификации неисправностей конкретного типа оборудования, фармокологии, анализе химсостава проб и т.д. Нейронные сети используются, кроме перечисленных областей, и в задачах с плохо структурированной информацией, например, при распознавании образов, рукописного текста, анализе речи и т.д.

Обучение нейронной сети с учителем как задача многофакторной оптимизации

Понятие о задаче оптимизации. Возможность применения теории оптимизации к обучению нейронных сетей крайне привлекательна, так как существует множество хорошо опробованных методов оптимизации, доведенных до стандартных компьютерных программ. Сопоставление процесса обучения с процессом поиска некоторого оптимума не лишено и биологических оснований, если рассматривать элементы адаптации организма к окружающим условиям в виде оптимального количества пищи, оптимального расходования энергии и т.п.

Функция одной действительной переменной $f(x)$ достигает локального минимума в некоторой точке x_0, если существует такая

δ-окрестность этой точки, что для всех x из этой окрестности, т.е. таких, что $|x - x_0| < \delta$, имеет место $f(x) > f(x_0)$.

Без дополнительных предположений о свойствах гладкости функции выяснить, является ли некоторая точка достоверной точкой минимума, используя данное определение, невозможно, поскольку любая окрестность содержит континуум точек. При применении численных методов для приближенного поиска минимума исследователь может столкнуться с несколькими проблемами. Во-первых, минимум функции может быть не единственным. Во-вторых, на практике часто необходимо найти глобальный, а не локальный минимум, однако обычно не ясно, нет ли у функции еще одного, более глубокого, чем найденный, минимума.

Математическое определение локального минимума функции в многомерном пространстве имеет тот же вид, если заменить точки x и x_0 на векторы, а вместо модуля использовать норму. Поиск минимума для функции многих переменных (многих факторов) является существенно более сложной задачей, чем для одной переменной. Это связано прежде всего с тем, что локальное направ-ление уменьшения значения функции может не соответствовать направлению движения к точке минимума. Кроме того, с ростом размерности быстро возрастают затраты на вычисление функции.

Решение задачи оптимизации во многом является искусством, общих, заведомо работающих и эффективных в любой ситуации методов нет. Среди часто используемых методов можно рекомендовать симплекс-метод Нелдера, некоторые градиентные методы, а также методы случайного поиска.

В случае если независимые переменные являются дискретными и могут принимать одно значение из некоторого фиксированного набора, задача многомерной оптимизации несколько упрощается. При этом множество точек поиска становится конечным, а следовательно, задача может быть, хотя бы в принципе, решена методом полного перебора. Будем называть оптимизационные задачи с конечным множеством поиска задачами *комбинаторной оптимизации*.

Для комбинаторных задач также существуют методы поиска приближенного решения, предлагающие некоторую стратегию перебора точек, сокращающую объем вычислительной работы. Имитация отжига и генетический алгоритм также применимы и к комбинаторной оптимизации.

Постановка задачи оптимизации при обучении нейронной сети. Пусть имеется нейронная сеть, выполняющая преобразование

F: $X \to Y$ векторов X из признакового пространства входов X в векторы Y выходного пространства Y. Сеть находится в состоянии W из пространства состояний W. Пусть далее имеется обучающая выборка ($X\alpha$,$Y\alpha$), $\alpha = 1...p$. Рассмотрим полную ошибку E, делаемую сетью в состоянии W:

$$E = E(W) = \sum_{\alpha} \left\| F(X^{\alpha};W) - Y^{\alpha} \right\| = \sum_{\alpha} \sum_{i} [F_i(X^{\alpha};W) - Y_i^{\alpha}]^2$$

Отметим два свойства полной ошибки. Во-первых, ошибка $E = E(W)$ является *функцией состояния* W, определенной на пространстве состояний. По определению она принимает неотрицательные значения. Во-вторых, в некотором обученном состоянии W^*, в котором сеть не делает ошибок на обучающей выборке, данная функция принимает нулевое значение. Следовательно, обученные состояния являются *точками минимума* введенной функции $E(W)$.

Таким образом, задача обучения нейронной сети является задачей поиска минимума функции ошибки в пространстве состояний, и, следовательно, для ее решения могут применяться стандартные методы теории оптимизации. Эта задача относится к классу многофакторных задач, так, например, для однослойного персептрона с N входами и M выходами речь идет о поиске минимума в $N \times M$-мерном пространстве.

На практике могут использоваться нейронные сети в состояниях с некоторым малым значением ошибки, не являющихся в точности минимумами функции ошибки. Другими словами, в качестве решения принимается некоторое состояние из окрестности обученного состояния W^*. При этом допустимый уровень ошибки определяется особенностями конкретной прикладной задачи, а также приемлемым для пользователя объемом затрат на обучение.

Необходимость иерархической организации нейросетевых архитектур. Многослойный персептрон

Необходимость иерархической организации нейросетевых архитектур. Особенности строения биологических сетей подталкивают исследователя к использованию более сложных, в частности, иерархических архитектур. Идея относительно проста – на низших уровнях иерархии классы преобразуются таким образом, чтобы сформировать линейно разделимые множества, которые в свою

очередь будут успешно распознаваться нейронами на следующих (высших) уровнях иерархии.

Однако основной проблемой, традиционно ограничивающей возможные сетевые топологии простейшими структурами, является проблема обучения. На этапе обучения сети предъявляются некоторые входные образы, называемые обучающей выборкой, и исследуются получаемые выходные реакции. Цель обучения состоит в приведении наблюдаемых реакций на заданной обучающей выборке к требуемым (адекватным) реакциям путем изменения состояний синаптических связей. Сеть считается обученной, если все реакции на заданном наборе стимулов являются адекватными.

Данная классическая схема обучения с учителем требует явного знания ошибок при функционировании каждого нейрона, что затруднено для иерархических систем, где непосредственно контролируются только входы и выходы. Кроме того, необходимая избыточность в иерархических сетях приводит к тому, что состояние обучения может быть реализовано многими способами, что делает само понятие «ошибка, делаемая данным нейроном» весьма неопределенным.

Наличие таких серьезных трудностей в значительной мере сдерживало прогресс в области нейронных сетей вплоть до середины 80-х годов прошлого века, когда были получены эффективные алгоритмы обучения иерархических сетей.

Многослойный персептрон. Рассмотрим иерархическую сетевую структуру, в которой связанные между собой нейроны (узлы сети) объединены в несколько слоев (рис. 4.26).

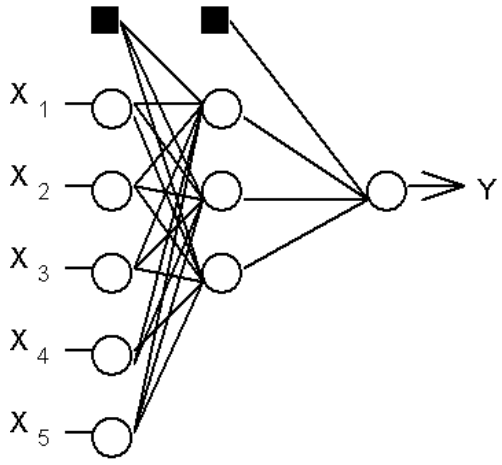


Рис. 4.26. Структура многослойного персептрона с пятью входами, тремя нейронами в скрытом слое и одним нейроном выходного слоя

На возможность построения таких архитектур указал еще Ф. Розенблатт, однако им не была решена проблема обучения.

Межнейронные синаптические связи сети устроены таким образом, что каждый нейрон на данном уровне иерархии принимает и обрабатывает сигналы от каждого нейрона более низкого уровня. Таким образом, в данной сети имеется выделенное направление распространения нейроимпульсов – от входного слоя через один или несколько скрытых слоев к выходному слою нейронов. Нейросеть такой топологии будем называть обобщенным многослойным персептроном или просто персептроном.

Персептрон представляет собой сеть, состоящую из нескольких последовательно соединенных слоев формальных нейронов МакКаллока и Питтса. На низшем уровне иерархии находится входной слой, состоящий из сенсорных элементов, задачей которого является только прием и распространение по сети входной информации. Далее имеется один или, реже, несколько скрытых слоев. Каждый нейрон на скрытом слое имеет несколько входов, соединенных с выходами нейронов предыдущего слоя или непосредственно со входными сенсорами $X_1..X_n$, и один выход. Нейрон характеризуется уникальным вектором весовых коэффициентов W. Веса всех нейронов слоя формируют матрицу, которую будем обозначать V или W. Функция нейрона состоит в вычислении взвешенной суммы его входов с дальнейшим нелинейным преобразованием ее в выходной сигнал:

$$y = 1 \Big/ \left(1 + \exp(-[\sum_i W_i x_i - \Theta]) \right)$$

Выходы нейронов последнего, выходного слоя описывают результат классификации $Y = Y(X)$. Особенности работы персептрона состоят в следующем. Каждый нейрон суммирует поступающие к нему сигналы от нейронов предыдущего уровня иерархии с весами, определяемыми состояниями синапсов, и формирует ответный сигнал (переходит в возбужденное состояние), если полученная сумма выше порогового значения. Персептрон переводит входной образ, определяющий степени возбуждения нейронов самого нижнего уровня иерахии, в выходной образ, определяемый нейронами самого верхнего уровня. Число последних обычно сравнительно невелико. Состояние возбуждения нейрона на верхнем уровне говорит о принадлежности входного образа к той или иной категории.

Традиционно рассматривается аналоговая логика, при которой допустимые состояния синаптических связей определяются произвольными действительными числами, а степени активности нейронов – действительными числами между 0 и 1. Иногда исследуются также

модели с дискретной арифметикой, в которой синапс характеризуется двумя булевыми переменными: активностью (0 или 1) и полярностью (-1 или +1), что соответствует трехзначной логике. Состояния нейронов могут при этом описываться одной булевой переменной. Данный дискретный подход делает конфигурационное пространство состояний нейронной сети конечным (не говоря уже о преимуществах при аппаратной реализации).

Обучение методом обратного распространения ошибок. Для обучения многослойной сети в 1986 г. Д.Е. Руммельхартом и Г.Е. Хинтоном был предложен алгоритм обратного распространения ошибок (error back propagation). Многочисленные публикации о промышленных применениях многослойных сетей с этим алгоритмом обучения подтвердили его принципиальную работоспособность на практике.

Вначале возникает резонный вопрос: почему для обучения многослойного персептрона нельзя применить уже известное δ-правило Розенблатта. Ответ состоит в том, что для применения метода Розенблатта необходимо знать не только текущие выходы нейронов y, но и требуемые *правильные* значения Y. В случае многослойной сети эти правильные значения имеются только для нейронов выходного слоя. Требуемые значения выходов для нейронов скрытых слоев неизвестны, что и ограничивает применение δ-правила.

Основная идея обратного распространения состоит в том, как получить оценку ошибки для нейронов скрытых слоев. Известные ошибки, делаемые нейронами выходного слоя, возникают вследствие неизвестных пока ошибок нейронов скрытых слоев. Чем больше значение синаптической связи между нейроном скрытого слоя и выходным нейроном, тем сильнее ошибка первого влияет на ошибку второго. Следовательно, оценку ошибки элементов скрытых слоев можно получить как взвешенную сумму ошибок последующих слоев. При обучении информация распространяется от низших слоев иерархии к высшим, а оценки ошибок, делаемые сетью, – в обратном направлении, что и отражено в названии метода.

Контрольные вопросы

1. Основные архитектуры вычислительных сетей, используемых в САПР.
2. Понятие базы данных. Типовая организация СУБД. Объектно-ориентированные СУБД.
3. Основные модели данных в базах данных.
4. Какие данные называются структурированными?
5. Дайте определение и опишите назначение базы данных.

6. Дайте определение и опишите назначение системы управления базой данных.

7. В чем заключается различие архитектур баз данных, организованных по принципу «клиент – сервер» и «файл – сервер»?

8. Назовите и поясните взаимосвязь структурных элементов базы данных.

9. Дайте понятие ключа. Какие виды ключей вы знаете?

10. Какие характеристики указываются при описании структуры базы данных и каково назначение такого описания?

11. Данные каких типов могут храниться в полях базы данных?

12. Какие модели данных вы знаете?

13. Поясните назначение ключевых полей в реляционной базе данных.

14. Что называется инфологической моделью предметной области?

15. Какие виды связей между объектами вам известны?

16. В чем заключается принцип нормализации отношений?

17. Каким требованиям должны отвечать отношения, находящиеся в первой, второй и третьей нормальных формах?

18. Каковы основные функциональные возможности СУБД?

19. Основные виды информационного обеспечения САПР.

20. Интеллектуальные САПР в проектировании электронных средств.

21. Понятие структурного синтеза. Параметрический синтез.

22. Синтезирующая интеллектуальная САПР.

23. Принцип работы персептрона.

24. Многослойный персептрон: обучение методом обратного распространения ошибок.

25. Обучение нейронной сети с учителем как задача многофакторной оптимизации.

26. Теорема об обучении персептрона.

Задания для самостоятельной работы

1. Перечислите команды для выполнения типовых операций в среде СУБД.

2. Назовите и охарактеризуйте основные этапы технологического процесса обработки информации с использованием СУБД.

СПИСОК СОКРАЩЕНИЙ

CASE – Computer-Aided Software/System Engineering

SADT – Structured Analysis and Design Technique

АРМ – автоматизированное рабочее место

АСНИ – автоматизированная система научных исследований

АСТПП – автоматизированная система технологической подготовки производства

АСУ ТП – автоматизированная система управления технологическими процессами

БД – база данных

ВП – вычислительный процесс

ВС – вычислительная сеть

ГАП – гибкие автоматизированные производства

ГАПС – гибкие автоматизированные производственные системы

ГПС – гибкие производственные системы

ДРП – дискретное рабочее поле

ЕСТД – Единая система технологической документации

ЕСТПП – Единая система технологической подготовки производства

ЖЦ – жизненный цикл

ЗПР – задача принятия решения

ЗУ – запоминающее устройство

ИИ – искусственный интеллект

ИМД – иерархическая модель данных

ИО – информационное обеспечение

ИС – информационная система

ИСАПР – интеллектуальные системы автоматизированного проектирования

ИСИИ – инструментальная система искусственного интеллекта

КМ – концептуальная модель

КП – коммутационное поле

КТС – комплекс технических средств

ЛВС – локальные вычислительные сети

ЛМ – логическая модель

МД – модель данных

МК – маршрутная карта

ММ – математическая модель

МО – математическое обеспечение

НИР – научно-исследовательские работы
НС – нейросетевые системы
ОКР – опытно-конструкторские работы
ПО – программное обеспечение
ПОЯ – проблемно-ориентированные языки
ПП – печатная плата
ППП – пакет прикладных программ
РМД – реляционная модель данных
РЭС – радиоэлектронное средство
САПР – система автоматизированного проектирования
СБИС – сверхбольшая интегральная схема
СД – словарь данных
СДС – средняя длина связи
СМД – сетевая модель данных
СПД – система передачи данных
СУБД – система управления базой данных
ТЗ – техническое задание
ТО – технологическая операция
ТП – технологический процесс
ТПП – технологическая подготовка производства
ТхО – техническое обеспечение
ТЭЗ – типовой элемент замены
УВГИ – устройство ввода графической информации
ЭкС – экспертные системы
ЭС – электронные средства
ЯМД – язык манипулирования данными
ЯОД – язык описания данных

ЛИТЕРАТУРА

1. ГОСТ 23501.101-87. Системы автоматизированного проектирования. Основные положения. – М.: Изд-во стандартов, 1988.

2. ГОСТ 23501.108-85. Системы автоматизированного проектирования. Классификация и обозначение. – М.: Изд-во стандартов, 1985.

3. Андреев, А.М. Внутренний мир объектно-ориентированных СУБД / А.М. Андреев, Д.В. Березкин. – М.: Финансы и статистика, 2001. – 147 с.

4. Асаи, К. Прикладные нечеткие системы: [пер. с япон.] / К. Асаи, Д. Ватада, С. Иваи [и др.]; под ред. Т. Тэрано, К. Асаи, М. Сугэно. – М.: Мир, 1993. – 368 с.

5. Боггс, У. UML и Rational Rose 2002: [пер. с англ.] / У. Боггс, М. Боггс. – М.: Лори, 2004. – 510 с.

6. Бойко, В. В. Проектирование баз данных информационных систем / В.В. Бойко, В. М. Савинков. — М.: Финансы и статистика, 1989. – 320 с.

7. Бусленко, Н.П. Моделирование сложных систем / Н.П. Бусленко. – М.: Наука, 1978. – 400 с.

8. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ / Г. Буч.– 3-е изд. – СПб.: Вильямс, 2008. – 721 с.

9. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем / А.М. Вендров. – М.: Финансы и статистика, 2003. – 176 с.

10. Вендров, А.М. Один из подходов к выбору средств проектирования баз данных и приложений / А.М. Вендров // СУБД. – №3. – 1995. – С. 45–52.

11. Волкова, В.Н. Основы теории систем и системного анализа: учебник для студентов вузов, обучающихся по специальности «Системный анализ и управление» / В.Н. Волкова, А.А. Денисов. – СПб.: Из-во СПбГТУ,1997. – 570 с.

12. Грабер, М. Введение в SQL: [пер. с англ.]./ М. Грабер. – М.: Лори, 1996. – 382 с.

13. Дейт, К. Введение в системы баз данных: [пер. с англ.] / К. Дейт. – М.: Наука, 1980. – 464 с.

14. Диго, С. М. Проектирование и использование баз данных: учебник / С. М. Диго. – М.: Финансы и статистика, 1995. – 208 с.

15. Заде, Л. Понятие лингвистической переменной и его применение к принятию приближенных решений / Л. Заде. – М.: Мир, 1976. – 167 с.

16. Калянов, Г.Н. CASE. Структурный системный анализ (автоматизация и применение) / Г.Н. Калянов. – М.: Лори, 1996. – 275 с.

17. Конструирование функциональных узлов ЭВМ на интегральных схемах / Б.Н. Ермолаев [и др.]. – М.: Радио и связь, 1998. – 200 с.

18. Корячко, В.П. Теоретические основы САПР / В.П. Корячко, В.М. Курейчик, И.П. Норенков. – М.: Энергоатомиздат, 1987. – 400 с.

19. Кузнецов, О.П. Дискретная математика для инженера / О.П. Кузнецов, Г.М. Адельсон-Вельский. – 2-е изд. – М.: Энергоатомиздат, 1988. – 480 с.

20. Курейчик, В.М. Математическое обеспечение конструкторского и технологического проектирования с применением САПР / В.М. Курейчик. – М.: Радио и связь, 1990. – 352 с.

21. Куроуз, Дж. Компьютерные сети / Дж. Куроуз, К. Росс. – 4-е изд. – СПб.: Питер, 2004. – 765 с.

22. Лобанова, В.А. Системы автоматизации технологических комплексов с транспортным запаздыванием (проблематика построения и современные информационные технологии их моделирования) / В.А. Лобанова, А.И. Суздальцев. – Орел: ОрелГТУ, 2004. – 134 с.

23. Спортак, М. Компьютерные сети и сетевые технологии / М. Спортак, Ф. Паппас [и др.]. – Киев: ТИД «ДС», 2002. – 736 с.

24. Марка, Д.А. Методология структурного анализа и проектирования / Д.А. Марка, К. МакГоуэн. – М.: МетаТехнология, 1993. – 240 с.

25. Милославская, Н. Г Интрасети: доступ в Internet, защита: учебное пособие для вузов / Н.Г. Милославская [и др.]. – М.: ЮНИТИ, 1999. – 468 с.

26. Мейер, М. Теория реляционных баз данных / М. Мейер. – М.: Мир, 1987. – 608 с.

27. Информационные технологии проектирования радиоэлектронных средств: учебное пособие [Электронный ресурс] / Д.Ю. Муромцев [и др.]. – СПб.: Лань, 2018. – 412 с. – Режим доступа: https://e.lanbook.com/book/109618.

28. Норенков, И.П. Информационная поддержка наукоемких изделий [Электронный ресурс] / И.П. Норенков, П.К. Кузьмик. — М.:

МГТУ им. Н.Э. Баумана, 2002. – Режим доступа: https://bookini.ru/informatsionnaya-podderzhka-naukoyomkih-izdelij-cals-tehnologii/

29. Муромцев, Ю.Л. Задачи трассировки печатных плат : лабораторные работы [Электронный ресурс] / Ю.Л. Муромцев, В.В. Трейгер, В.Н. Грошев. – Тамбов : ТИХМ, 1990. – 32 с. – Режим доступа: http://crems.jesby.tstu.ru/files/e-books/

30. Муромцев, Ю.Л. Задачи размещения радиоэлектронной аппаратуры : лабораторные работы [Электронный ресурс] / Ю.Л. Муромцев, В.В. Трейгер, В.Н. Грошев. – Тамбов : ТИХМ, 1988. – 32 с. – Режим доступа: http://crems.jesby.tstu.ru/files/e-books/

31. Норенков, И. П. Основы автоматизированного проектирования: учебник для вузов [Электронный ресурс] / И. П. Норенков. – М.: МГТУ им. Баумана, 2006. – 448 с. – Режим доступа: https://lib-bkm.ru/load/19-1-0-196

32. Ризкин, И.Х. Машинный анализ и проектирование технических систем / И.Х. Ризкин. – М.: Наука, 1985. – 160 с.

33. Росс, Д. Структурный анализ (SA): язык для передачи понимания [пер. с англ.] / Д. Росс. – М.: Мир, 1984. – 284 с.

34. Саати, Т. Принятие решений. Метод анализа иерархий / Т. Саати. – М.: Радио и связь, 1993. – 320 с.

35. Советов, Б.Я. Моделирование систем: учебник для вузов по спец. «Автоматизированные системы управления» / Б.Я. Советов, С.А. Яковлев. – М.: Высшая школа, 2002. – 271с.

36. Тиори, Т. Проектирование структур баз данных: в 2 кн. Кн. 1. [пер. с англ.] / Т. Тиори, Дж. Фрай. – М.: Мир, 1985. – 287 с.

37. Уемов, Л.И. Системный подход и общая теория систем / Л.И. Уемов. – М.: Мысль, 1978. – 272 с.

38. Ульман, Дж. Основы систем баз данных [пер. с англ.] / Дж. Ульман; под ред. М. Р. Когаловского. – М.: Финансы и статистика, 1983. – 334 с.

39. Щербаков, А.Ю. Введение в теорию и практику компьютерной безопасности / А.Ю. Щербаков. – М.: Мир, 2001. – 351 с.

40. Шиндер, Д. Основы компьютерных сетей [пер. с англ.] / Д. Шиндер. – М.: Издательский дом «Вильямс», 2003. – 656 с.

41. Пат. 2147036 Российская Федерация, МПК С14В1/28. Способ управления клеймением параметров движущихся кож / А.И. Суздальцев, В.А. Лобанова. – Опубл. 27.03.2000, Бюл. № 1.

Учебное издание

Лобанова Валентина Андреевна
Воронина Оксана Александровна

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ЭЛЕКТРОННЫХ СРЕДСТВ

Учебное пособие

Редактор Т.Д. Васильева
Технический редактор Т.П. Прокудина

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Орловский государственный университет имени И.С. Тургенева»

Подписано к печати 23.04.2020 г. Формат 60×90 1/16.
Усл. печ. л. 14,9. Тираж 100 экз.
Заказ №_____

Отпечатано с готового оригинал-макета
на полиграфической базе ОГУ имени И.С. Тургенева
302026, г. Орел, ул. Комсомольская, 95.