САНКТ-ПЕТЕРБУРГ
МОСКВА
КРАСНОДАР
2016
ЛАНЬ®

Б. Я. СОВЕТОВ, В. В. ЦЕХАНОВСКИЙ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ:
теоретические основы

ДОПУЩЕНО
УМО вузов РФ по университетскому политехническому образованию
в качестве учебного пособия для студентов вузов, обучающихся по направлению подготовки бакалавра «Информационные системы и технологии»

· САНКТ-ПЕТЕРБУРГ ·
· МОСКВА · КРАСНОДАР ·
· 2016 ·

ББК 32.81я73
С 56

Советов Б. Я., Цехановский В. В.

С 56 Информационные технологии: теоретические основы: Учебное пособие. — СПб.: Издательство «Лань», 2016. — 448 с.: ил. — (Учебники для вузов. Специальная литература).

ISBN 978-5-8114-1912-8

В учебном пособии на основе современных тенденций развития информатики рассмотрены вопросы становления и развития информационных технологий.

Информационные технологии рассматриваются как единая система, базирующаяся на основных информационных процессах, базовых информационных технологиях, поддерживаемых соответствующей инструментальной стратой.

Представленный материал формирует у студентов представление об информационных технологиях в контексте промышленных методов и средств работы с информацией в различных сферах человеческой деятельности, обеспечивающих рациональное и эффективное ее использование.

Для бакалавров учреждений высшего профессионального образования, обучающихся по укрупненной группе специальностей «Информатика и вычислительная техника».

ББК 32.81я73

Рецензенты:

Р. М. Юсупов — доктор технических наук, профессор, директор Санкт-Петербургского института информатики и автоматизации Российской академии наук, член-корреспондент РАН, заслуженный деятель науки и техники РФ;

Н. П. Меткин — доктор технических наук, профессор, ген. директор ООО «Санкт-Петербургская ассоциация предприятий радиоэлектроники».

Обложка
Е. А. ВЛАСОВА

ВВЕДЕНИЕ

Информационные технологии являются составной частью научного направления «Информатика» и базируются на ее достижениях. Информатизация как процесс перехода к информационному обществу сопровождается возникновением новых и интенсивным развитием существующих информационных технологий. Информация превращается в коммерческий ресурс, способствуя получению прибыли при внедрении информационных технологий во многие сферы человеческой деятельности. Возникают информационная экономика, новая информационная инфраструктура промышленности и социальной сферы, формируется информационная культура. Достижения информатики могут позитивно воздействовать на развитие общества при приоритетном развитии образования. Существует мнение, что преодоление современного кризиса мировой образовательной системы возможно на основе информатизации.

Зарубежный и российский опыт внедрения информационных технологий подтверждает высокую экономическую эффективность для многих сфер применения.

В своем становлении любая отрасль, в том числе и информационная, проходила стадии от кустарного ремесленного производства к производству, основанному на высоких технологиях.

Информационные технологии обеспечивают переход от рутинных к промышленным методам и средствам работы с информацией в различных сферах человеческой

деятельности, обеспечивая рациональное и эффективное ее использование.

В данном пособии с позиций системного подхода, теории информации, теории моделирования, искусственного интеллекта и других прикладных наук информатики предлагается подход к рассмотрению информационных технологий как науки о промышленных способах ее переработки, преобразования и использования.

Необходимость создания данного пособия связана с широким развитием информационных технологий и использованием их в различных предметных областях. Овладение основами информационных технологий позволяет принципиально на новых позициях решать проблемы использования информационных ресурсов. Однако содержание, модели, методы и средства современных информационных технологий слабо раскрыты в технической литературе, доступ к которой ограничен. Авторами предпринята попытка системного изложения вопросов теории и практического применения информационных технологий в различных приложениях с использованием современных компьютерных средств.

В результате освоения дисциплины студент должен

1) знать:

- структуру, состав и свойства информационных процессов, систем и технологий, методы анализа информационных систем, модели представления проектных решений, конфигурации информационных систем;
- состав, структуру, принципы реализации и функционирования информационных технологий, используемых при создании информационных систем, базовые и прикладные информационные технологии, инструментальные средства информационных технологий;
- классификацию информационных систем, структуры, конфигурации информационных систем, общую характеристику процесса проектирования информационных систем;
- принципы, базовые концепции технологий программирования, основные этапы и принципы создания программного продукта;

- основные положения теории баз данных, хранилищ данных, витрин данных, баз знаний, концептуальные, логические и физические модели данных;
- основные виды и процедуры обработки информации, модели и методы решения задач обработки информации;
- теорию технологий искусственного интеллекта (математическое описание экспертной системы, логический вывод, искусственные нейронные сети, расчетно-логические системы, системы с генетическими алгоритмами, мультиагентные системы);
- состав и структуру инструментальных средств, тенденции их развития (операционные системы, языки программирования, технические средства);
- модели и структуры информационных сетей; информационные ресурсы сетей; теоретические основы современных информационных сетей;
- основные этапы, методологию, технологию и средства проектирования информационных систем;

2) уметь:

- разрабатывать информационно-логическую, функциональную и объектно-ориентированную модели информационной системы, модели данных информационных систем;
- применять информационные технологии при проектировании информационных систем;
- использовать архитектурные и детализированные решения при проектировании систем;
- осуществлять математическую и информационную постановку задач по обработке информации, использовать алгоритмы обработки информации для различных приложений;
- проводить предпроектное обследование (инжиниринг) объекта проектирования, системный анализ предметной области, их взаимосвязей, проводить выбор исходных данных для проектирования информационных систем, проводить сборку информационной системы из готовых компонентов, адаптировать приложения к изменяющимся условиям функционирования;

3) владеть:

- методами и средствами представления данных и знаний о предметной области, методами и средствами анализа информационных систем, технологиями реализации, внедрения проекта информационной системы;
- методологией использования информационных технологий при создании информационных систем;
- моделями и средствами разработки архитектуры информационных систем;
- навыками владения одной из технологий программирования;
- инструментальными средствами обработки информации;
- информационными технологиями поиска информации и способами их реализации — технологиями интеллектуального анализа данных, интеллектуальными технологиями поддержки принятия решений (на основе хранилищ данных, оперативной аналитической обработки информации и интеллектуального анализа данных);
- технологиями построения и сопровождения инфокоммуникационных систем и сетей;
- методами и средствами проектирования, модернизации и модификации информационных систем.

Учебное пособие включает шесть разделов, объединяющих 29 тем.

В первой главе последовательно рассматриваются понятия, виды и свойства информации, даются фундаментальные определения информации, ее количественные и качественные оценки, ставится проблема превращения информации в ресурс. Информационные технологии рассматриваются авторами как система. Определяются основные понятия и задачи информационной технологии, приводятся этапы эволюции. Во второй главе раскрываются базовые информационные процессы, входящие в состав информационных технологий. Для каждого из рассматриваемых процессов, таких как извлечение информации, транспортирование, обработка, хранение,

представление и использование информации, дается подробная характеристика с раскрытием моделей и современного состояния. Непосредственно содержанию информационных технологий посвящена третья глава книги, в которой детально раскрываются базовые информационные технологии, к которым отнесены: мультимедиатехнологии, геоинформационные, технологии защиты информации, CASE-технологии, телекоммуникационные, технологии искусственного интеллекта, технологии программирования, облачные технологии, технология больших данных.

В четвертой главе авторами приводится анализ прикладных информационных технологий. Ввиду многообразия предметных технологий, основное внимание уделено технологиям корпоративного управления, которые инвариантны к сфере применения. Особое внимание уделяется современным и перспективным типовым стандартным технологическим и инструментальным средствам, которые полезно применять в различных сферах деятельности. Непрерывное совершенствование существующих и возникновение новых информационных технологий невозможно без развития их инструментальной базы, чему и посвящена пятая глава. В учебном пособии дается анализ и приводятся рекомендации по использованию программных, технических и методических средств информационных технологий. Это позволит читателю сориентироваться на сложившемся рынке вычислительной и сетевой продукции с учетом изложенных в книге мировых стандартов, а также самостоятельно выявить тенденции развития и перспективы информационных технологий и информатики в целом. Шестая глава посвящена информационной технологии построения систем, что особенно актуально для формирования профессионалов-разработчиков. Приводятся основы системного подхода, интерпретированные к задачам построения информационных систем. Раскрываются стадии их разработки. Приводится методика построения информационных систем с оценкой их качества.

Учебное пособие предназначено для бакалавров учреждений высшего образования, обучающихся по

укрупненной группе специальностей «Информатика и вычислительная техника». По своему содержанию учебное пособие является вводным ко всем дисциплинам профессионального цикла и направлено на формирование единой концепции информационных технологий, которые раскрываются в других дисциплинах.

ГЛАВА 1

ВОЗНИКНОВЕНИЕ И ЭТАПЫ СТАНОВЛЕНИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

В результате освоения данной темы студент должен

1) знать:

- понятия информации и знаний, виды информации, формы ее представления, роль в процессе управления;
- свойства информации;
- количественные и качественные характеристики информации, возможности их применения;
- тенденции превращения информации в ресурс;
- определение и задачи информационных технологий, уровни их рассмотрения;
- состав и структуру информационных технологий;
- этапы эволюции информационных технологий;

2) уметь:

- различать информацию по видам и формам ее проявления;
- выделять свойства информации для конкретных предметных областей;
- учитывать специфику конкретной предметной области в специализированных прикладных информационных технологиях;
- выделять информационные процессы для формирования структуры информационных технологий;

3) владеть:

- навыками анализа информации о конкретной предметной области;
- навыками анализа структуры и состава информационных процессов и технологий.

1.1. ПОНЯТИЕ ИНФОРМАЦИИ, ВИДЫ ИНФОРМАЦИИ

Процесс движения мирового сообщества сопровождается чередой различных социальных и промышленных революций, что наглядно видно на рисунке 1.1 [1]. Каждый такой переход, помимо основательного изменения политического и экономического устройства, приводил к значительному прорыву в индустриальном развитии.

На рубеже 1970–1980-х гг. достижения науки привели к созданию высокотехнологичного промышленного производства. Внедрение ЭВМ и сетей передачи данных способствовали революционным процессам в области информационных технологий. В процессе своего развития человечество в любой сфере деятельности последовательно проходило стадии от ручного кустарного труда до высокотехнологичного промышленного производства. В первую очередь усилия были направлены на облегчение физического труда, а информационная сфера долгие годы была уделом умственного труда человека и с каждым годом требовала большего количества трудовых ресурсов. Революционные процессы в области информатизации позволили перейти на промышленный уровень технологий и инструментальных средств.

Отличительной чертой человеческого общества является то, что в течение длительного времени основным предметом труда оставались материальные объекты. Воздействуя на них, человек добывал себе средства к существованию, и на протяжении многих веков решалась задача усиления мускульных возможностей человека с помощью различных инструментов, агрегатов и машин. На

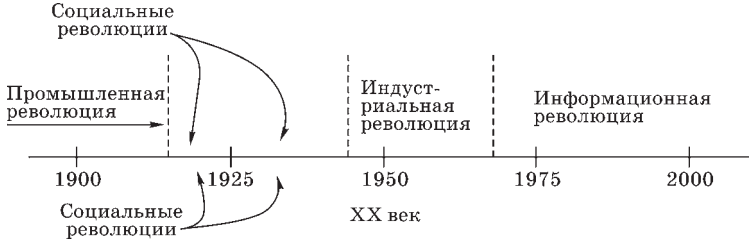


Рис. 1.1
Последовательность развития мирового сообщества

это была направлена механизация производства, которая стала интенсивно внедряться в начале XX в. Развитие человеческого общества практически на всех этапах проходило на основе технического прогресса. Это и овладение огнем, и использование паровых машин, и проникновение в тайны атомной энергии, и т. п. Повышению производительности труда способствовала автоматизация. В процессе формирования трудовых коллективов возникла необходимость обмена знаниями. Первоначально знания передавались устно из поколения в поколение; появление письменности позволило по-новому показать накопленные знания — представить их в виде информации. Предполагают, что между первыми инструментами обработки материальных объектов и средствами отображения информационных образов существовал временной интервал около миллиона лет. Таким образом, появление информации является естественным следствием развития человеческого общества.

В настоящее время информация является одним из самых дорогих видов ресурсов. Это проявляется в тенденции стремительного перекачивания трудовых ресурсов из сферы материального производства в информационную. Например, в США в конце XIX в. свыше 95% трудоспособного населения было занято физическим трудом и только менее 5% — работой по обработке информации. Сегодня мы наблюдаем картину соотношения трудовых ресурсов с точностью до наоборот. В 1940-х гг. экстенсивный фактор как средство преодоления разрыва между потребностями и возможностями обработки информации себя исчерпал. Это явилось толчком к созданию новых средств обработки информации — ЭВМ и переходу к интенсивному развитию информационной индустрии и, как следствие, к становлению научного направления — теории информации.

Теория информации быстро и прочно внедряется в самые различные области человеческого знания: физику, химию, биологию, медицину, философию, лингвистику, педагогику, экономику, логику, технические науки, эстетику и др.

Чрезвычайно большое значение приобрел термин «информация» в связи с развитием информационных технологий.

Информация — это абстрактное понятие, если относить ее к определенному классу закономерностей материального мира и процессу отражения его в человеческом сознании. Существуют различные определения. Н. Винером указывалось, что информация — это обозначение содержания, полученного из внешнего мира. К. Шеннон дал ее определение как передаваемые сообщения, которые уменьшают неопределенность у получателя информации. У. Эшби определил информацию как передачу разнообразия. А. М. Яглом полагал, что информация — это вероятность выбора. Л. Бриллюен определил ее как отрицание энтропии. Энтропийные и негэнтропийные оценки информации оказались перспективными.

Понятие энтропии в теории информации впервые было введено К. Шенноном как мера количества информации, вырабатываемой источником, пропускаемой каналом или попадающей к получателю (в пересчете на символ или секунду). В более общем плане энтропия является показателем неопределенности, беспорядка, разнообразия, хаоса, равновесия в системе. Негэнтропия, ошибочно рассматриваемая как энтропия с отрицательным знаком, является мерой порядка, упорядоченности, внутренней структуры, связанной информации.

Отсутствие строго научного определения информации во многом связано с разным смыслом, вкладываемым в это понятие в зависимости от предметной области. Восприятие информации математиком, химиком, биологом во многом отличается. Структура, форма представления, содержание информации для разных групп пользователей приводит к ее разным понятиям. Даже на бытовом уровне наблюдается расслоение в восприятии информации различными группами пользователей. Неизменным остается необходимость извлечения, передачи, хранения, обработки и представления информации. Все это и является основными задачами теории информационных технологий.

Никакая информация, никакое знание не появляется сразу — этому предшествует этап накопления, осмысления, систематизации опытных данных, взглядов. Знание — продукт такого процесса. Мышление — необходимый атрибут такого процесса.

Информация может существовать в пассивной (неактуализированной) и активной (актуализированной) форме [2].

Информацию можно дифференцировать по различным признакам. Нас в первую очередь будет интересовать классификация (табл. 1.1) по критериям и свойствам, относящимся к информационным системам и технологиям. Более подробная классификация видов информации в общем плане приведена в [3].

В философском аспекте информацию можно разделить на мировоззренческую; эстетическую; религиозную; научную; бытовую; техническую; экономическую; технологическую.

Особой категорией информации являются знания. Информология — область знаний, изучающая проявление информации, ее представление, измерение.

Знания и информация обладают специфическими свойствами. Например, известно высказывание Б. Шоу: «Если у тебя и меня имеется по одному яблоку, и мы ими обменялись, то у каждого из нас осталось по одному яблоку; если у тебя и меня имеется по одной идее и мы ими обменялись, то у каждого из нас будет по две идеи».

Подытоживая сказанное выше, отметим, что в настоящее время существуют три точки зрения при обсуждении проблем информации. Первая точка зрения отождествляет понятие информации со знанием. Хотя данный подход широко критикуется в отечественной литературе, во многих научных трудах он имеет место. Вторая точка зрения ограничивает предметную область понятия информации социальными и биологическими процессами, отвергая существование информационных процессов в неорганической природе. Третья точка зрения, широко используемая в настоящее время, связана с атрибутивным понятием информации. Впервые атрибутивное понятие информации

Таблица 1.1

Классификация видов информации

Классификационный признак	Виды информации		
	Вид		Характеристика
Отношение к внешней среде	Входная		Информация, воспринимаемая от окружающей среды
	Выходная		Информация, производимая системой и выдаваемая в окружающую среду
	Внутренняя		Информация, производимая и хранимая в системе
Форма представления для обработки	Дискретная		Последовательность дискретных сигналов
	Аналоговая		Непрерывная величина
Форма представления для пользователя	Визуальная	Символьная	Совокупность условных символов
		Графическая	Совокупность изображений (графики, рисунки, диаграммы)
		Текстовая	Совокупность букв и цифр
	Звуковая		Совокупность звуковых сигналов
	Мультимедийная		Комбинированная форма представления
Стадия обработки	Первичная		Информация до начала процесса обработки
	Вторичная		Информация после окончания процесса обработки
	Промежуточная		Информация в качестве исходных данных для последующей обработки
	Результативная		Информация, полученная в ходе решения задачи
По изменению во времени	Постоянная		Неизменная и многократно используемая информация в течение всего времени решения задачи
	Переменная		Отражает фактические количественные и качественные изменения в предметной области
	Условно-постоянная		Неизменная и многократно используемая информация на установленном отрезке времени
По отношению к достижению цели системы	Синтаксическая		Определяет способ представления информации
	Семантическая		Оценивает смысл передаваемой информации
	Прагматическая		Определяет возможность достижения поставленной цели
По форме проявления	Активная		Изменяет состояние предметной области
	Пассивная		Не влияет на состояние предметной области

было сформулировано Н. Винером, полагавшим, что все явления в природе охватываются тремя основными понятиями: вещество, энергия, информация. В отличие от Н. Винера, не рассматривавшего взаимосвязь этих компонентов, многие современные авторы тесно увязывают их и рассматривают как единую систему.

Понятие информации должно быть связано с определенным объектом, свойства которого она отражает. Кроме того, наблюдается относительная независимость информации от носителя, поскольку возможны ее преобразование и передача по различным физическим средам с помощью разнообразных физических сигналов безотносительно к ее содержанию, т. е. к семантике, что и явилось центральным вопросом многих исследований, в том числе и в философской науке. Информация о любом материальном объекте может быть получена путем наблюдения, натурного либо вычислительного эксперимента, а также на основе логического вывода. Поэтому говорят о доопытной, или априорной, информации и послеопытной, т. е. апостериорной, полученной в итоге эксперимента.

При обмене информацией имеют место источник в виде объекта материального мира и приемник — человек либо какой-то материальный объект. Информация возникает за счет отражения, которое является свойством всей материи, любой материальной системы. Свойство отражения совершенствуется по мере развития материи от элементарного отражения до высшей его формы — сознания. Процесс отражения означает взаимодействие объектов материального мира.

Информация — результат отражения. Информация отображает некоторый образ реального мира, который в дальнейшем может существовать независимо от материального объекта. Действительно, для описания естественных либо искусственно созданных объектов используют информационные модели, которые далее могут быть исходным материалом для разработки систем. Очень важно, чтобы эти модели были адекватны реальным объектам. Любое исследование сопровождается большим объемом информации, которая требует обработки, представления

и использования зачастую в реальном масштабе времени. Таким образом, понятие информации предполагает наличие двух объектов — источника информации и потребителя. Важно, чтобы информация для потребителя имела смысл. Потребитель информации может оценивать ее в зависимости от того, где и для какой конкретной задачи информация используется. Поэтому выделяют такие аспекты информации, как прагматический, семантический и синтаксический.

Прагматический аспект связан с возможностью достижения поставленной цели с использованием получаемой информации. Этот аспект информации влияет на поведение потребителя. Если информация была эффективной, то поведение потребителя меняется в желаемом направлении, т. е. информация имеет прагматическое содержание. Таким образом, этот аспект характеризует поведенческую сторону проблемы.

Семантический аспект позволяет оценить смысл передаваемой информации. Определяется семантическими связями между словами или другими смысловыми элементами языка.

Синтаксический аспект информации связан со способом ее представления. В зависимости от реального процесса, в котором участвует информация (осуществляется ее сбор, передача, преобразование, отражение, представление, ввод или вывод), она представляется в виде специальных знаков, символов.

Виды информации

Все виды деятельности человека по преобразованию природы и общества сопровождались получением новой информации. Логическая информация, адекватно отображающая объективные закономерности природы, общества и мышления, получила название научной информации. Ее делят по областям получения или использования на следующие виды: политическую, техническую, биологическую, химическую, физическую и т. д.; по назначению — на массовую и специальную. Часть информации, которая занесена на бумажный носитель,

получила название документальной информации. Любое производство при функционировании требует перемещения документов, т. е. возникает документооборот. Наряду с научной информацией в сфере техники при решении производственных задач используется техническая информация. Она сопровождает разработку новых изделий, материалов, конструкций, агрегатов, технологических процессов. Научную и техническую информацию объединяют термином «научно-техническая информация».

Верхним уровнем информации как результата отражения окружающей действительности (результата мышления) являются знания. Знания возникают как итог теоретической и практической деятельности. Информация в виде знаний отличается высокой структуризацией. Это позволяет выделить полезную информацию при анализе окружающих нас физических, химических и прочих процессов и явлений. На основе структуризации информации формируется информационная модель объекта. По мере развития общества информация как совокупность научно-технических данных и знаний превращается в базу системы информационного обслуживания научно-технической деятельности общества.

С развитием общества возникает необходимость целесообразной организации информационного ресурса, т. е. концентрации имеющихся фактов, данных и знаний по направлениям науки и техники. Признание информации как ресурса и появление понятия «информационный ресурс» дало толчок развитию нового научного направления — информатики. Информатика как область науки и техники связана со сбором и переработкой больших объемов информации на основе современных программно-аппаратных средств вычислительной техники и техники связи.

Важным аспектом информации является ее главенствующая роль в процессе управления.

Круг объектов управления чрезвычайно широк и разнообразен: экономика, территория, социальная сфера, производство, научный эксперимент, образование и др.

При анализе процесса управления ввиду сложности объекта производят его расчленение на части по различным признакам. Одним из главных признаков является вид иерархии. Характерны следующие виды иерархии: временная, пространственная, функциональная, ситуационная и информационная. Следует отметить, что деление какой-либо системы на части не может быть однозначным, так как выделение границ между частями является всегда в какой-либо мере субъективным. Выбор того или иного принципа выделения составных частей должен удовлетворять следующим основным условиям: обеспечивать их максимальную автономность; учитывать необходимость координации их действий для достижения общей цели функционирования, а также совместимость отдельных частей.

Временная иерархия. Признаком деления здесь является интервал времени от момента поступления информации о состоянии объекта управления до выдачи управляющего воздействия. Чем больше интервал, тем выше уровень (ранг) элемента. Управление может осуществляться в реальном времени, с интервалом сутки, декада, месяц, квартал и т. д. Причем управляющий интервал выбирается не произвольно, а исходя из критериев, определяющих устойчивость и эффективность функционирования всей системы.

Пространственная иерархия. Признаком деления здесь является площадь, занимаемая объектом управления. Чем больше площадь объекта, тем выше его ранг. Данный признак является субъективным, так как не всегда площадь, занимаемая объектом, соответствует его значимости, и его можно использовать в случае аналогичности параметров элементов одного уровня.

Функциональная иерархия. В основе лежит функциональная зависимость (подчиненность) элементов системы. Такое разделение также является субъективным, так как в этом случае трудно выделить границы между элементами системы.

Ситуационная иерархия. Деление на уровни в данном случае производится в зависимости от эффекта, вызывае-

мого той или иной ситуацией, например от ущерба, возникающего в результате аварии или выхода из строя оборудования.

Информационная иерархия. В настоящее время этот вид иерархии является очень существенным в связи с возросшим значением информации для управления. В основе деления на уровни лежит оперативность и обновляемость информации. Именно через эти характеристики прослеживается иерархия информации по уровням управления предприятием.

На первом уровне хранится и обрабатывается повторяющаяся, часто обновляющаяся информация, необходимая для повседневной деятельности, т. е. для оперативного управления. Следующий уровень составляет информация более обобщенная, чем оперативная, и используемая не так часто. Информация группируется по функциональным областям и применяется для поддержки принятия решения по управлению производством. На верхнем уровне хранится и обрабатывается стратегическая информация для долгосрочного планирования. Для нее характерны высокая степень обобщенности, неповторяемость, непредсказуемость и редкое использование.

В общем виде функциональная модель процесса управления представлена на рисунке 1.2. Учет информации об объекте управления состоит в регистрации, классификации и идентификации. На основе разнообразных математических моделей, описывающих реальное и требуемое состояние объекта, и критериев оптимальности анализируют информацию о состоянии объекта управления. Окончательная модель прогнозируемого состояния объекта управления формируется в виде плана. Возни-

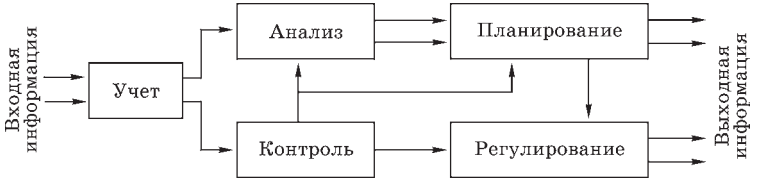


Рис. 1.2
Функциональная модель процесса управления

кающие за счет внешних воздействий отклонения от плана корректируют путем сравнения учетной и плановой информации, нового анализа и формирования управляющих воздействий (регулирования).

В большинстве случаев при информационном анализе процесса управления обычно рассматривают пассивную форму проявления информации, отражающую свойства внешней среды, объекта управления и самой управляющей системы. Однако не менее важное значение имеет и активная форма информации, являющаяся причиной изменения состояния управляемого объекта.

Принято выделять следующие качественно различимые формы проявления информации: осведомляющую — $I_{ос}$, преобразующую — $I_{п}$, принятия решения — $I_{пр}$ и управляющую — $I_{у}$.

К осведомляющей относят информацию о состоянии внешней среды, объекта управления и управляющей системы. Преобразующая включает информацию, содержащуюся в алгоритмах управления. Информация принятия решения является отражением образов и целей на конечное множество принимаемых решений. Управляющей является информация, вызывающая целенаправленное изменение состояния объекта управления.

В любой системе управления можно выделить два информационных канала: целевой и рабочий. В целевом канале на основе информационных процессов происходит выбор цели и принятие решения по выбору управляюще-

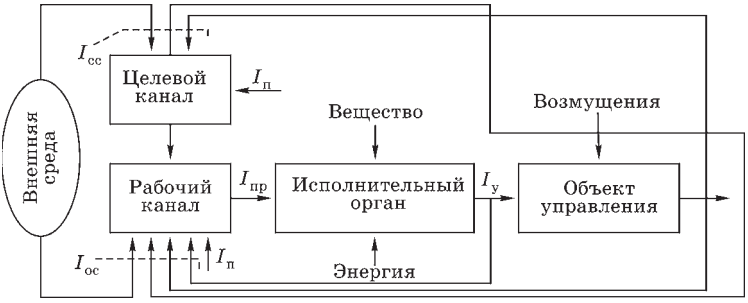


Рис. 1.3
Информационная структура системы управления

го воздействия. В рабочем канале формируется информация, реализуемая исполнительным органом, осуществляющим целенаправленное изменение состояния объекта управления через вещественно-энергетические характеристики. Целевой канал может находиться как на одном уровне иерархии с рабочим, так и на более высоком. На рисунке 1.3 представлена информационная структура системы управления, где выделены целевой и рабочий каналы, а также приведены основные формы проявления информации.

1.2. СВОЙСТВА ИНФОРМАЦИИ

Информация многогранна в своих проявлениях, как со стороны материальных форм существования, так и с точки зрения воздействия на информационную систему и окружающую среду. Все это разнообразие проявляется в статических и динамических свойствах информации. Статические описывают информацию в определенный момент времени, а динамические — изменение и развитие информации во времени.

Динамические свойства информации характеризуют изменение информации во времени.

Подробно основные свойства информации изложены в [2], [3]. Мы остановимся на свойствах, имеющих непосредственное отношение к информационным системам и технологиям.

Адекватность информации — это степень соответствия реальному объективному состоянию дел.

Адекватность информации может выражаться в трех формах: семантической, синтаксической, прагматической.

Синтаксическая адекватность отображает формально-структурные характеристики информации и не затрагивает ее смыслового содержания. На синтаксическом уровне учитываются тип носителя и способ представления информации, скорость передачи и обработки, размеры кодов представления информации, надежность и точность преобразования этих кодов и т. п.

Эта форма способствует восприятию внешних структурных характеристик, т. е. синтаксической стороны информации.

Семантическая (смысловая) адекватность. Эта форма определяет степень соответствия образа объекта и самого объекта и предполагает учет смыслового содержания информации. На этом уровне анализируются те сведения, которые отражает информация, рассматриваются смысловые связи.

Прагматическая (потребительская) адекватность отражает отношение информации и ее потребителя, соответствие информации цели управления, которая на ее основе реализуется.

Актуальность — это степень соответствия информации текущему моменту времени информации, определяется степенью сохранения ценности информации для управления в момент ее использования и зависит от динамики изменения ее характеристик и от интервала времени, прошедшего с момента возникновения данной информации.

Достоверность информации — свойство отражать реально существующие объекты с необходимой точностью. Измеряется достоверность информации доверительной вероятностью необходимой точности, т. е. вероятностью того, что отображаемое информацией значение параметра отличается от истинного значения этого параметра в пределах необходимой точности.

Информация достоверна, если она отражает истинное положение дел. Объективная информация всегда достоверна, но достоверная информация может быть как объективной, так и субъективной. Достоверная информация помогает принять нам правильное решение. Недостоверной информация может быть по следующим причинам:

- преднамеренное искажение (дезинформация) или непреднамеренное искажение субъективного свойства;
- искажение в результате воздействия помех и недостаточно точных средств ее фиксации.

Достаточность (*полнота*) — характеризует качество информации и определяет минимальный, но достаточный

набор данных для принятия решений или создания новых данных на основе имеющихся. Понятие полноты информации связано с ее смысловым содержанием (семантикой) и прагматикой.

Доступность информации. Возможность получения информации, необходимой пользователю. На степень доступности информации влияют одновременно как форма представления и методы интерпретации, так и ограничения использования.

Форма представления и методы интерпретации информации реализуются путем согласования ее семантической формы с тезаурусом пользователя, а также интерфейсом пользователя.

Ограничения использования подразделяют информацию на следующие виды:

- секретная, имеющая существенные ограничения использования;
- конфиденциальная (для служебного пользования), отражающая интересы общества или отдельных групп людей в форме накладываемых ограничений на использование;
- публичная (открытая) информация, не имеющая ограничений использования.

Реализация ограничений осуществляется применением методов и средств защиты информации от несанкционированного доступа.

Краткость. Представление информации в наиболее сжатой форме без потери ее содержания.

Кумулятивность. Характеризует накопление и хранение информации, способность ее к обобщенному и компактному изложению.

Концентрация информации. Проявляется в тенденции к объединению и представлению в укрупненной форме.

Массовость. Предусматривают два аспекта: качественный аспект раскрывает массовость информации как информации общественной, общей для всех; количественный — как информации, распространяемой для широкой сети потребителей, пользователей информации.

Мобильность в пространстве и во времени без потери актуальности.

Неассоциативность и некоммутативность информации. Информация не может рассматриваться как арифметическая сумма составляющих ее элементов, и эти элементы нельзя использовать в другой последовательности.

Неисчерпаемость информации — возможность ее многоразового и многоцелевого использования, неотчуждение при обмене или продаже.

Объективность и субъективность информации. Понятие объективности информации является относительным. Это понятно, если учесть, что методы являются субъективными. Более объективной принято считать ту информацию, в которую методы вносят меньший субъективный элемент. В ходе информационного процесса степень объективности информации всегда понижается.

Понятность — свойство информации, основанное на том, что информация становится понятной, если она выражена языком, на котором говорят те, кому предназначена эта информация.

Преемственность информации. Характеризует возможность повторного использования информациии в процессе обработки.

Рассеяние информации. Свойство, противоположное концентрации информации. По мере концентрации информации в одном месте, она становится менее значимой в другом.

Репрезентативность информации связана с правильностью ее отбора и формирования в целях адекватного отражения свойств объекта. Важнейшее значение здесь имеют:

- правильность концепции, на базе которой сформулировано исходное понятие;
- обоснованность отбора существенных признаков и связей отображаемого явления.

Нарушение репрезентативности информации приводит нередко к существенным ее погрешностям.

Самовозрастание. Постоянный рост объема в процессе информационного производства, независимость формы представления и конкретного носителя.

Своевременность информации означает ее поступление не позже заранее назначенного момента времени, согласованного с временем решения поставленной задачи.

Смысл и новизна. Это свойство характеризует перемещение информации в социальных коммуникациях и выделяет ту ее часть, которая нова для потребителя.

Содержательность информации отражает семантическую составляющую, определяемую отношением количества семантической информации в сообщении к объему обрабатываемых данных, т. е.

$$C = \frac{I_{\text{с}}}{V_{\text{д}}}.$$

С увеличением содержательности информации растет семантическая пропускная способность информационной системы, так как для получения одних и тех же сведений требуется преобразовать меньший объем данных.

Наряду с коэффициентом содержательности C, отражающим семантический аспект, можно использовать и коэффициент информативности, характеризующийся отношением количества синтаксической информации (по Шеннону) к объему данных:

$$Y = \frac{I}{V_{\text{д}}}.$$

Старение информации. Процесс снижения ее ценности для пользователя. Зависит от частоты внесения изменений и их соотвествия реальному состоянию предметной области.

Точность информации определяется степенью соответствия получаемой информации к реальному состоянию объекта, процесса, явления и т. п.

Трансформируемость информации означает независимость содержания информации от формы фиксации и способа предъявления.

Универсальность информации — содержание информации может быть предназначено для любого пользователя.

Целостность (*устойчивость*) — свойство информации сохранять форму и содержание при различных на нее воздействиях.

Ценность — свойство информации, зависящее от целевого назначения, т. е. предметной области распространения информации.

На ценность информации косвенно влияют следущие факторы: вероятность ее своевременного появления или получения, важность, доступность, полнота, достоверность, актуальность.

Ценность является субъективной характеристикой, зависящей от требований пользователя.

Эмерджентность информации. Свойство информационных систем, порождаемое взаимодействием элементов и ненаблюдаемое ни в одном из элементов, если они рассматриваются отдельно.

1.3. КОЛИЧЕСТВЕННЫЕ И КАЧЕСТВЕННЫЕ ХАРАКТЕРИСТИКИ ИНФОРМАЦИИ

Возможен ряд подходов к оценке качества информации. Наиболее существенными из них являются синтаксический, семантический, прагматический и алгоритмический. На рисунке 1.4 представлена классификация мер информации.

На *синтаксическом уровне* для определения количества информации используются два подхода: объемный и статистический (вероятностный, энтропийный). В объемном подходе синтаксическая мера количества информации оперирует с обезличенной информацией, не выражающей смыслового отношения к объекту. При этом учитываются тип носителя и способ представления информации, скорость передачи и обработки, размеры кодов представления информации.

Объем данных понимается в техническом смысле этого слова как информационный объем сообщения или как

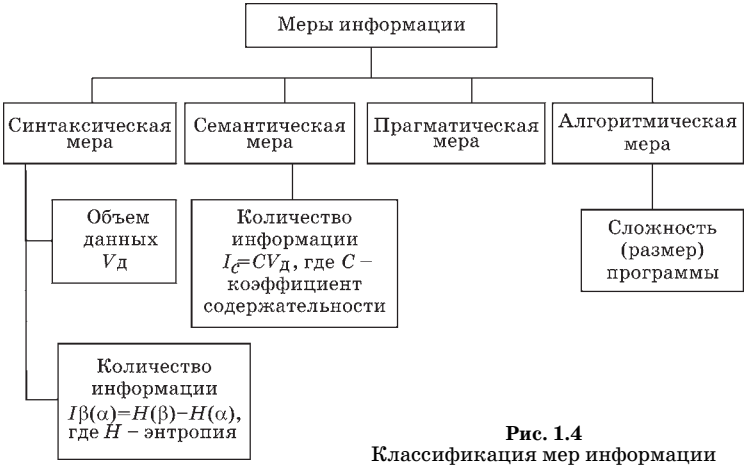


Рис. 1.4
Классификация мер информации

объем памяти, необходимый для хранения сообщения без каких-либо изменений. При этом прагматический аспект не рассматривается.

Информационный объем сообщения измеряется в *битах* и равен количеству двоичных цифр (0 и 1), которыми закодировано сообщение.

В компьютерной практике понятие «бит» используется также как единица измерения объема памяти. Ячейка памяти размером в 1 бит может находиться в двух состояниях (включено и выключено), и в нее может быть записана одна двоичная цифра (0 или 1). Так как бит — слишком маленькая единица измерения информации, пользуются кратными ей величинами. Основной единицей измерения информации является *байт*. 1 байт равен 8 битам. В ячейку размером в 1 байт можно поместить 8 двоичных цифр, т. е. в одном байте можно хранить $256 = 2^8$ различных чисел. Для измерения еще бóльших объемов информации используются величины, представленные в таблице 1.2.

Количество информации на синтаксическом уровне определяется также на основе статистического (вероятностного) подхода через понятие энтропии системы.

Статистический подход представлен в обширном разделе кибернетики — теории информации, которая занимается математическим описанием и оценкой методов

Таблица 1.2

Единицы измерения объемов информации

Измерение в байтах						
десятичная приставка			двоичная приставка			
название	символ	степень	название	символ		степень
				МЭК	ГОСТ	
Килобайт	KB	10^3	Кибибит	KiB	Килобит	2^{10}
Мегабайт	MB	10^6	Мебибит	MiB	Мегабит	2^{20}
Гигабайт	GB	10^9	Гибибит	GiB	Гигабит	2^{30}
Терабайт	TB	10^{12}	Тебибит	TiB	Терабит	2^{40}
Петабайт	PB	10^{15}	Пебибит	PiB		2^{50}
Эксбайт	EB	10^{18}	Эксбибит	EiB		2^{60}
Зеттабайт	ZB	10^{21}	Зебиабит	ZiB		2^{70}
Йоттабайт	YB	10^{24}	Йобибит	YiB		2^{80}

передачи, хранения, извлечения и классификации информации. Теория информации в математической основе использует методы теории вероятности, математической статистики, линейной алгебры и др. В статистической теории основное внимание обращается на распределение вероятности появления отдельных событий и построение на его основе обобщенных характеристик, позволяющих оценить количество информации в одном событии либо в их совокупности. Количественной мерой информации стала энтропия. Чтобы возник процесс передачи, должны иметь место источник информации и потребитель. Источник выдает сообщение, потребитель, принимая сообщение, принимает при этом информацию о состоянии источника. В статистической теории, как выше указывалось, не изучается содержание информации. Предполагается, что до получения информации имела место некоторая неопределенность. С получением информации эта неопределенность снимается. Таким образом, статистическая количественная характеристика информации — это мера снимаемой в процессе получения информации неопределенности системы.

В рамках статистического подхода наиболее широкое распространение получили два подхода: формулы Хартли и Шеннона.

Американский инженер Р. Хартли процесс получения информации рассматривал как выбор одного сообщения из конечного наперед заданного множества из N равновероятных сообщений, а количество информации I, содержащееся в выбранном сообщении, определял как двоичный логарифм N:

$$I = \log_2 N.$$

Американский ученый К. Шеннон предложил в 1948 г. другую формулу определения количества информации, учитывающую возможную неодинаковую вероятность сообщений в наборе:

$$I = -(p_1 \log_2 p_1 + p_2 \log_2 p_2 + \ldots + p_N \log_2 p_N),$$

где p_i — вероятность того, что именно i-е сообщение выделено в наборе из N сообщений.

Если вероятности $p_1, \ldots, p_N$ равны, то каждая из них равна $1/N$, и формула Шеннона превращается в формулу Хартли.

Следует отметить, что понятие энтропии исторически использовалось для оценки меры неопределенности состояния любой системы. Чем больше энтропия системы, тем больше неопределенность ее состояния и тем большую информацию получаем, когда эта неопределенность снимается. Энтропия как количественная мера информации обладает следующими свойствами:

1) функция энтропии является непрерывной относительно вероятности возникновения событий и для дискретных событий имеет наибольшее значение при равной вероятности их появления. Если возможно появление лишь одного события, то априорной неопределенности нет, поэтому количество информации и энтропия равны нулю;

2) при равновероятных событиях функция энтропии возрастает с увеличением числа событий в ансамбле, а поэтому для повышения информативности символов необходимо увеличивать основание системы счисления используемого кода;

3) функция энтропии не зависит от пути выбора событий. Это свойство вытекает из аддитивности статической

меры информации и, как следствие, аддитивности функции энтропии.

Таким образом, статистическая теория позволяет дать плодотворные оценки количества информации для такого важного этапа информационного процесса в системе, как передача. Заложенные еще К. Шенноном принципы количественной оценки на основе функции энтропии сохраняют свою значимость до настоящего времени и являются полезными при определении информативности символов и сообщений, при оценке оптимальности построения кода на основе критериев избыточности.

В современных системах обработки информации и управления существенное место занимает подготовка информации для принятия решения и сам процесс принятия решения в системе. Здесь существенную помощь может оказать семантическая теория, позволяющая вскрыть смысл и содержание информации, выражаемой на естественном либо близком ему языке. С увеличением объема производства и его сложности количество информации, необходимое для принятия безошибочного решения, непрерывно возрастает. В этих условиях необходимо осуществлять отбор информации по некоторым критериям, т. е. предоставлять руководителю либо лицу, принимающему решение, своевременную и полезную информацию. С учетом ошибок, которые могут возникать в информации в связи с действиями оператора, отказами технических средств и др., избыточность допускается лишь как средство борьбы с ошибками. В этом смысле можно считать, что избыточность способствует сохранению ценности информации, обеспечивая требуемую верность. В рамках семантического подхода ценность информации можно задать через функцию потерь. Если в процессе подготовки информации исходная величина x отображается через величину y, то минимум потерь можно установить как

$$\Pi_{\min} = \min_{Y} \sum_{X} \Pi(x / y) P(x),$$

где $P(x)$ — распределение входной величины x; $\Pi(x/y)$ — потери при преобразовании входной величины x в величину y.

Отсюда ценность информации определяется как

$$Ц = \max_{P(x/y)} [П_{\min} - M\{П(x / y\}],$$

где $M\{П(x, y)\}$ — математическое ожидание потерь при отходе от входной величины x к величине y.

Следует отметить, что данная интерпретация ценности имеет сугубо технический характер. Конструктивным выходом из нее является такое разбиение входной величины x, при котором удается максимизировать ценность. В общем случае ценность информации, поступающей от материального объекта, является функцией времени. Анализ информации, используемой для принятия решения в реальных системах, позволил найти функции ценности. Эти функции задают предельные временные интервалы, в течение которых имеет смысл использовать данную информацию. При принятии решения обычно используется не только информация о материальном объекте, но и информация об условных распределениях критериальных оценок последствий различных альтернативных решений. В этом случае резко уменьшается число предпочтительных альтернатив и удается принять решение, базируясь на качественно неполной информации. В ряде практических случаев решение принимается с использованием субъективных критериев, при этом приходится применять большой объем информации, ужесточать требования к согласованности и непротиворечивости исходной информации. Принцип принятия решений по своей методологии требует сохранения содержания качественных понятий на всех этапах использования информации при общей оценке альтернативных решений. Кроме того, исключается сложная информация, при которой лицо, принимающее решение, должно иметь дело с громоздкими задачами. Используют замкнутые процедуры выявления предпочтений, т. е. процедуры, в которых имеется возможность проверить предпочтение на непротиворечивость и транзитивность. Можно отметить, что семантическая теория требует дальнейшей серьезной проработки, од-

нако уже сейчас при принятии решений существует ряд методов, позволяющих оценивать смысловое содержание информации.

Семантический подход базируется на смысловом содержании информации. Термин «семантика» исторически применялся в металогике и семиотике. В металогике под семантикой понимают изучение связей между знакосочетаниями, входящими в состав какого-либо формализованного языка, и их интерпретациями (истолкованиями) в терминах той системы понятия и представлений, формализацией которого служит данный язык. В более узком смысле под семантикой подразумевают совокупность правил соответствия между формальными выражениями и их интерпретацией. Под семиотикой понимают комплекс научных теорий, изучающих свойства знаковых систем, т. е. систем конкретных или абстрактных объектов, с каждым из которых сопоставлено некоторое значение. Примерами знаковых систем являются естественные языки, а также искусственные языки, такие как алгоритмические, языки программирования, информационные и др.

Семантическая мера информации. Для измерения смыслового содержания информации, т. е. ее количества на семантическом уровне, наибольшее признание получила тезаурусная мера (предложена Ю. И. Шрейдером), которая связывает семантические свойства информации со способностью пользователя принимать поступившее сообщение. Для этого используется понятие «тезаурус пользователя» [4].

Тезаурус — это совокупность сведений, которыми располагает пользователь или система.

В зависимости от соотношений между смысловым содержанием информации S и тезаурусом пользователя S_p изменяется количество семантической информации I_c, воспринимаемой пользователем и включаемой им в дальнейшем в свой тезаурус. Характер такой зависимости показан на рисунке 1.5. Рассмотрим два предельных случая, когда количество семантической информации I_c равно нулю:

- при $S_p \to 0$ пользователь не воспринимает, не понимает поступающую информацию;
- при $S_p \to 1$ пользователь все знает, и поступающая информация ему не нужна.

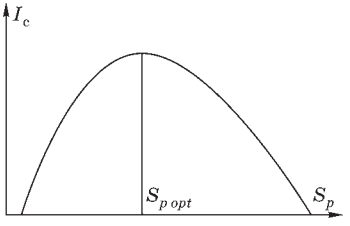


Рис. 1.5
Зависимость количества семантической информации, воспринимаемой потребителем, от его тезауруса

Максимальное количество семантической информации I_c потребитель приобретает при согласовании ее смыслового содержания S со своим тезаурусом S_p ($S_p = S_{p\,opt}$), когда поступающая информация понятна пользователю и несет ему ранее не известные (отсутствующие в его тезаурусе) сведения. Следовательно, количество семантической информации в сообщении, количество новых знаний, получаемых пользователем, является величиной относительной. Одно и то же сообщение может иметь смысловое содержание для компетентного пользователя и быть бессмысленным для пользователя некомпетентного. Относительной мерой количества семантической информации может служить коэффициент содержательности C, рассмотренный выше.

Прагматический (*аксиологический*) *подход* к информации базируется на анализе ее ценности с точки зрения потребителя. Например, информация, имеющая несомненную ценность для биолога, будет иметь ценность, близкую к нулевой, для программиста. Ценность информации связывают со временем, поскольку с течением времени она стареет и ценность ее, а следовательно, и «количество» уменьшаются. Таким образом, прагматический подход оценивает содержательный аспект информации. Он имеет особое значение при использовании информации для управления, поскольку ее количество тесно связано с эффективностью управления в системе.

Прагматическая мера информации определяет полезность информации (ценность) для достижения пользователем поставленной цепи. Эта мера также величина отно-

сительная, обусловленная особенностями использования этой информации в той или иной системе.

Ценность информации целесообразно измерять в тех же самых единицах (или близких к ним), в которых измеряется целевая функция.

Алгоритмический подход связан с желанием внедрения универсальной меры информации. Количественная характеристика, отражающая сложность (размер) программы, позволяющая произвести какое-либо сообщение, была предложена А. Н. Колмогоровым.

Так как существуют разные способы задания и реализации алгоритма с использованием различных вычислительных машин и языков программирования, то для определенности задается некоторая конкретная машина, например машина Тьюринга. В этом случае в качестве количественной характеристики сообщения можно взять минимальное число внутренних состояний машины, требующихся для воспроизведения данного сообщения.

Разные подходы к оценке количества информации заставляют, с одной стороны, использовать разнотипные единицы информации для характеристики различных информационных процессов, а с другой стороны, увязывать эти единицы между собой как на логическом, так и на физическом уровнях. Например, процесс передачи информации, измеряемой в одних единицах, сопрягается с процессом хранения информации, где она измеряется в других единицах, и т. д., а поэтому выбор единицы информации является весьма актуальной задачей.

В таблице 1.3 сопоставлены меры информации.

Таблица 1.3

Сопоставление мер информации

Мера информации	Единицы измерения	Примеры (для компьютерной области)
Синтаксическая: объемный подход; вероятностный, шенноновский подход	Степень уменьшения неопределенности. Единицы представления информации	Вероятность события. Бит, байт, Кбайт и т. д.

Продолжение табл. 1.3

Мера информации	Единицы измерения	Примеры (для компьютерной области)
Семантическая	Тезаурус. Экономические показатели	Пакет прикладных программ, персональный компьютер, компьютерные сети и т. д. Рентабельность, производительность, коэффициент амортизации и т. д.
Прагматическая	Ценность использования	Емкость памяти, производительность компьютера, скорость передачи данных и т. д. Денежное выражение. Время обработки информации и принятия решений
Алгоритмическая	Сложность (размер) программы	Минимальное число внутренних состояний ЭВМ

1.4. ПРЕВРАЩЕНИЕ ИНФОРМАЦИИ В РЕСУРС

Ресурсное обеспечение любого вида деятельности составляют финансы, материальные ресурсы, штаты и информационные ресурсы.

Если первые три вида ресурсов можно рассматривать обособленно, то информационные ресурсы тесно взаимосвязаны с каждым из них и по уровню иерархии стоят выше, так как используются при управлении остальными.

Информацию как вид ресурса можно создавать, передавать, искать, принимать, копировать (в той или иной форме), обрабатывать, разрушать. Информационные образы могут создаваться в самых разнообразных формах: в форме световых, звуковых или радиоволн, электрического тока или напряжения, магнитных полей, знаков на бумажных носителях. Важность информации как экономической категории составляет одну из главнейших характеристик постиндустриальной эпохи.

Информационный ресурс — концентрация имеющихся фактов, документов, данных и знаний, отражающих реальное, изменяющееся во времени, состояние общества и используемых при подготовке кадров, в научных исследованиях и материальном производстве [5].

Факты — результат наблюдения за состоянием предметной области.

Документы — часть информации, определенным образом структурированная и занесенная на бумажный носитель.

Данные — вид информации, отличающийся высокой степенью форматированности, в отличие от более свободных структур, характерных для речевой, текстовой и визуальной информации.

Знания — итог теоретической и практической деятельности человека, отражающий накопление предыдущего опыта и отличающийся высокой степенью структуризации.

В знаниях можно выделить три основные составные части:

- декларативные (факторальные) знания, представляющие общее описание объекта, что не позволяет их использовать без предварительной структуризации в конкретной предметной области;
- понятийные (системные) знания, содержащие, помимо первой части, взаимосвязи между понятиями и свойства понятий;
- процедурные (алгоритмические) знания, позволяющие получить алгоритм решения.

Долгие годы информационный ресурс рассматривался как вспомогательный в развитии финансовых и материальных ресурсов. Однако в последние десятилетия все более явной становится тенденция к превращению информационного ресурса в главный фактор экономического развития. Это проявляется в следующих направлениях:

- информационный ресурс становится самостоятельным фактором экономики, выступая как товар, включая и информационные услуги;
- информационный ресурс становится основным компонентом материальных ресурсов (доля затрат на информацию в себестоимости продукции сравнивается или превышает остальные статьи затрат);
- возрастание роли информационных ресурсов в средствах труда, что проявляется в автоматизации процес-

сов материального производства и концентрации трудовых усилий в сфере интеллектуального производства.

Таким образом, четко прослеживается тенденция в сторону интеллектуального информационного производства. Так, один из ведущих производителей в сфере инструментальных средств вычислительной техники корпорация IBM в последние годы сделала резкий поворот в сторону когнитивных наук и направленности в сторону создания средств аналитики больших данных. В рамках этой инициативы с бюджетом в 17 млн евро, финансируемой Евросоюзом и названной RESERVOIR — Resources and Services Virtualization without Barriers (виртуализация ресурсов и сервисов без ограничений) — будут изучены возможности разработки и управления ИТ-сервисами в масштабах административных доменов (групп серверов, маршрутизаторов и сетей, управляемых одной организацией), информационно-технологических платформ и географических регионов. Цель этого проекта, охватывающего сферу cloud computing (облачные решения), как новой парадигмы вычислений — разработка технологий для поддержки онлайновой экономики, основанной на сервисах, где ресурсы и сервисы инициализируются и управляются в непрерывном и «прозрачном» для конечных пользователей режиме. Эксперты любой компании могут иметь доступ для анализа ситуации, после этого специалисты предоставляют свои рекомендации о путях более эффективного использования ресурсов и возможностей, показывают новые направления [6].

Функционирование информационного производства невозможно без исходного сырья — информации и накопленных знаний. Как вид ресурса, информация обладает рядом специфических особенностей, отличающих ее от традиционных факторов производства — природных, трудовых и финасовых ресурсов. Наиболее значимыми свойствами информации являются: виртуальный характер использования, нечеткое определение полезности, неявная зависимость между исходным объемов знаний и объемом нового созданного знания, высокая мобильность в физическом и интеллектуальном пространствах.

Повышение значимости и повсеместное распространение в качестве экономических ресурсов информации и знаний ведут не только к разнообразным положительным эффектам, в частности к экономии ресурсов, снижению нагрузки на окружающую среду, расширению возможностей людей, существуют и различные проблемы, свойственные экономике, в которой информация и знания становятся важными ресурсами. Так, ускорение темпов научно-технического прогресса приводит к усилению давления на общество, поскольку социальные, равно как и экономические институты, не успевают адаптироваться к изменениям. Информационная нагрузка может оказывать деструктивное влияние на людей, тем более что происходит все более жесткое и целенаправленное использование методов информационного воздействия.

1.5. ОПРЕДЕЛЕНИЕ И ЗАДАЧИ ИНФОРМАЦИОННОЙ ТЕХНОЛОГИИ

Теоретической базой для информационных технологий является информатика. Цель информатики — изучение структуры и общих свойств научной информации с выявлением закономерностей процессов коммуникации. В современном понимании информатика — это область науки и техники, изучающая информационные процессы и методы их автоматизации. Пользователю она предоставляет методологические основы построения информационной модели объекта. Примером такой модели является концептуальная модель, которая отражает реальное содержание конкретной предметной области.

В информатике можно выделить три уровня. Физический (нижний) уровень представляет собой средства вычислительной техники и техники связи. Их развитие оказывает решающее влияние на возможности и направление использования информатики. Логический (средний) уровень составляют информационные технологии. Прикладной (верхний) уровень определяет идеологию применения информационных технологий для проектирования различных систем.

Информатика как научное направление имеет ряд определений [47]. Это объясняется тем, что основным объектом изучения информатики является информация, точного определения которой нет до настоящего времени.

Теория информации, кибернетика и синергетика внесли значительный вклад в развитие информатики, однако не в состоянии описать и дать научное объяснение разнообразным информационным процессам, имеющим место в природе и обществе. Новое научное направление — инфодинамика — связывает воедино массу, энергию и негэнтропию.

Поскольку однозначного понимания научного направления «информатика» не существует, целесообразно говорить не об истории, а об ее задачах на современном этапе. Так как информация является отражением, то в информационном обществе мы имеем дело с приближенными моделями реального мира. В связи с этим главной задачей информатики должно быть методологическое обоснование построения информационной модели объекта, явления, процесса. Использование этой модели для целенаправленной деятельности в любых сферах человеческого общества осуществляется на основе реализации информационных процессов и соответствующих им технологий.

Таким образом, для современного состояния информационных технологий необходим переход от информационного описания предметной области к представлению на уровне данных, осуществляемому на основе декомпозиции, абстракции, агрегирования.

Декомпозиция — это разбиение системы (программы, задачи) на компоненты, объединение которых позволяет решить данную задачу.

Абстракция позволяет правильно выбрать нужные компоненты для декомпозиции. Абстракция представляет собой эффективный способ декомпозиции, осуществляемый посредством изменения списка декомпозиции.

Процесс абстракции может быть рассмотрен как некоторое обобщение. Он позволяет забыть о различиях и рассматривать предметы и явления так, как если бы они были эквивалентны. Так как выделение общего у процес-

сов и явлений есть основа классификации, то иерархия абстракций представляет собой фактически схему классификации.

Выделяют два основных способа абстрагирования:

- абстракция через параметризацию;
- абстракция через спецификацию.

Абстракция через параметризацию — выделение формальных параметров с возможностью их замены на фактические в различных контекстах. Выделение формальных параметров позволяет абстрагироваться от конкретного приложения и базируется на общности определенных свойств конкретных приложений.

Абстракция через спецификацию позволяет абстрагироваться от внутренней структуры до уровня знания свойств внешних проявлений (результата). Внешние свойства компонента указываются путем описания внешних связей, требований и эффектов.

Внешние связи — это связи различной природы данного компонента с окружением.

Требования (requires) — это условия, которые должны быть выполнены для правильного использования компонента.

Эффекты (effects) — это условия, которым удовлетворяют внешние проявления (результаты) компонента.

С точки зрения конкретных приложений выделяют следующие виды абстракций:

- процедурная абстракция (ПА);
- абстракция данных (АД);
- абстракция через итерацию (АИ).

Процедурная (функциональная) абстракция позволяет расширить некоторую виртуальную машину новой операцией.

Абстракция данных состоит из набора объектов и набора операций, характеризующих поведение этих объектов.

Абстракция через итерацию дает возможность не рассматривать информацию, не имеющую прямого отношения к управляющему потоку или циклу.

При построении модели данных предметной области наряду с естественным процессом декомпозиции исполь-

зуется и агрегирование. Это связано с необходимостью интеграции информационных ресурсов в силу их разнородности для ряда предметных областей.

Агрегирование — процесс объединения предметов в некоторую группу как в целях классификации, так и для обеспечения взаимодействия компонентов информационной системы.

В настоящее время при проектировании информационных систем используются два подхода: функционально-структурный и объектно-ориентированный.

Функционально-структурный подход (структурный) использует принцип алгоритмической декомпозиции с выделением функциональных элементов предметной области и установлением строгого порядка выполняемых действий. Недостатком данного способа является неизбежность продвижения информации в одну сторону («вниз по течению»), что в случае ошибки при проектировании приводит к деформированию системы.

По мере постепенного превращения компьютеров из счетных устройств в универсальные машины для обработки данных, примерно после 1970 г., стали появляться новые термины: данные как продукты (data product); инструменты для работы с данными (data tool); приложения, реализуемые посредством соответствующей организации (data application); наука о данных (data science); ученые, работающие с данными (data scientist), и даже журналисты, которые доносят сведения, содержащиеся в данных, до широкой публики (data journalist).

Технология (от *др.-греч.* τέχνη — искусство, мастерство, умение; λόγος —мысль, причина; методика, способ производства) — в широком смысле — совокупность методов, процессов и материалов, используемых в какой-либо отрасли деятельности, а также научное описание способов технического производства; в узком — комплекс организационных мер, операций и приемов, направленных на изготовление, обслуживание, ремонт и/или эксплуатацию изделия с номинальным качеством и оптимальными затратами и обусловленных текущим уровнем развития науки, техники и общества в целом.

При этом:

- под термином *изделие* следует понимать любой конечный продукт труда (материальный, интеллектуальный, моральный, политический и т. п.);
- под термином *номинальное качество* следует понимать прогнозируемое или заранее заданное качество, например оговоренное техническим заданием и согласованное техническим предложением.

Современные технологии основаны на достижениях научно-технического прогресса и ориентированы на производство продукта: материальная технология создает материальный продукт, информационная технология (ИТ) —информационный продукт. Технология — это также научная дисциплина, разрабатывающая и совершенствующая способы и инструменты производства. Технологией или *технологическим процессом* часто называют и сами операции добычи, транспортировки и переработки, которые являются основой производственного процесса.

Таким образом, термин «технология» имеет множество толкований. В широком смысле под технологией понимают науку о законах производства материальных благ, вкладывая в нее три основные части: идеологию, т. е. принципы производства; орудия труда, т. е. станки, машины, агрегаты; кадры, владеющие профессиональными навыками. Эти составляющие называются соответственно информационной, инструментальной и социальной. Для конкретного производства технологию понимают в узком смысле как совокупность приемов и методов, определяющих последовательность действий для реализации производственного процесса. Уровень технологий связан с научно-техническим прогрессом общества и влияет на его социальную структуру, культуру и идеологию. Для любой технологии могут быть выделены цель, предмет и средства. Целью технологии в промышленном производстве является повышение качества продукции, сокращение сроков ее изготовления и снижение себестоимости.

Методология любой технологии включает в себя: декомпозицию производственного процесса на отдельные взаимосвязанные и подчиненные составляющие (стадии,

этапы, фазы, операции); реализацию определенной последовательности выполнения операций, фаз, этапов и стадий производственного процесса в соответствии с целью технологии; технологическую документацию, формализующую выполнение всех составляющих.

Производство информации направлено на целесообразное использование информационных ресурсов и снабжение ими всех элементов организационной структуры и реализуется путем создания информационной системы. Информационные ресурсы являются исходным «сырьем» для системы управления любой организационной структуры, конечным продуктом является принятое решение. Принятие решения в большинстве случаев осуществляется в условиях недостатка информации, поэтому степень использования информационных ресурсов во многом определяет эффективность работы организации.

В своем становлении любая отрасль, в том числе и информационная, проходила стадии от кустарного ремесленного производства к производству, основанному на высоких технологиях.

Информационные технологии обеспечивают переход от рутинных к промышленным методам и средствам работы с информацией в различных сферах человеческой деятельности, обеспечивая рациональное и эффективное ее использование.

В развитии технологии выделяют два принципиально разных этапа: один этап характеризуется непрерывным совершенствованием установившейся базисной технологии и достижением верхнего предельного уровня, когда дальнейшее улучшение является неоправданным из-за больших экономических вложений; другой этап отличается отказом от существующей технологии и переходом к принципиально другой, развивающейся по законам первого этапа.

Информационная технология — совокупность методов и способов получения, обработки, представления информации, направленных на изменение ее состояния, свойств, формы, содержания и осуществляемых в интересах пользователей.

Можно выделить три уровня рассмотрения информационных технологий.

1-й уровень — теоретический. Основная задача — создание комплекса взаимосвязанных моделей информационных процессов, совместимых параметрически и критериально.

2-й уровень — исследовательский. Основная задача — разработка методов, позволяющих автоматизированно конструировать оптимальные конкретные информационные технологии.

3-й уровень — прикладной, который целесообразно разделить на две страты: инструментальную и предметную.

Инструментальная страта (аналог — оборудование, станки, инструмент) определяет пути и средства реализации информационных технологий, которые можно разделить на методические, информационные, математические, алгоритмические, технические и программные.

Предметная страта связана со спецификой конкретной предметной области и находит отражение в специализированных информационных технологиях, например организационное управление, управление технологическими процессами, автоматизированное проектирование, обучение и др.

Основным критерием оценки информационных технологий является их эффективность, особенно экономическая эффективность. Традиционный расчет прибыли производится с учетом исчисляемых расходов и доходов.

Зачастую степень эффективности определяют исходя из того, насколько выгодны решения, принимаемые с точки зрения функционирования систем и устройств, государства, права и бизнеса, образования, культуры и т. д. При этом не самым главным критерием является расчет финансовых вложений. Так, например, если информационные службы высокоэффективны, то не только бизнес, но и культура и образование выигрывают от их деятельности. В этом случае весьма затруднительно обосновать необходимые капиталовложения в информационные технологии. Здесь не работают расчеты, сделанные только

с учетом экономической эффективности, а обычно учитывается, каких результатов можно достигнуть при создании новой системы или модернизации существующей.

Успех применения информационной технологии может определяться эффективностью решения основных задач. Некоторые специалисты считают, что обоснование полезности — это искусство маркетинга. Для этого предлагается использовать разработанный Робертом Бенсоном метод «информационной экономики». Такой метод рассматривается как надежный способ анализа экономической эффективности, позволяющий учитывать качественные выгоды, величина которых определяется методом финансового прогноза с учетом возможных рисков.

Успешное внедрение информационных технологий связано с возможностью их типизации. Конкретная информационная технология обладает комплексным составом компонентов, поэтому целесообразно определить структуру и состав информационной технологии (рис. 1.6).

Технологический процесс — часть информационного процесса, содержащая действия (физические, механические и др.) по изменению состояния информации.

Рис. 1.6
Состав информационной технологии

Информационные технологии базируются на информации о предметной области, информационных процессах и являются основой реализации процессов информатизации и построения информационных систем на основе инструментального обеспечения (hardware + software), математического обеспечения (braineware) и методического обеспечения (orgware).

Конкретная информационная технология определяется в результате компиляции и синтеза базовых технологических операций, специализированных технологий и средств реализации.

Информационная технология базируется на реализации информационных процессов, разнообразие которых требует выделения базовых, характерных для любой информационной технологии.

Базовый технологический процесс основан на использовании стандартных моделей и инструментальных средств и может быть использован в качестве составной части информационной технологии. К их числу можно отнести: операции извлечения, транспортировки, хранения, обработки и представления информации.

Среди базовых технологических процессов выделим:

- извлечение информации;
- транспортирование информации;
- обработка информации;
- хранение информации;
- представление и использование информации.

Процесс извлечения информации связан с переходом от реального представления предметной области к его описанию в формальном виде и данных, которые отражают это представление.

Процесс транспортировки осуществляет передачу информации на расстояние для ускоренного обмена и организации быстрого доступа к ней, используя при этом различные способы преобразования.

Процесс обработки информации состоит в получении одних «информационных объектов» из других «информационных объектов» посредством выполнения некоторых алгоритмов; является одной из основных операций, вы-

полняемых над информацией, и главным средством увеличения ее объема и разнообразия.

Процесс хранения связан с необходимостью накопления и долговременного хранения данных с обеспечением их актуальности, целостности, безопасности, доступности.

Процесс представления и использования информации направлен на решение задачи доступа к информации в удобной для пользователя форме.

Базовые технологические процессы строятся на основе базовых технологических операций, но кроме этого включают ряд специфических моделей и инструментальных средств. Этот вид технологий ориентирован на решение определенного класса задач и используется в конкретных технологиях в виде отдельной компоненты. Среди них можно выделить:

- облачные технологии;
- мультимедиатехнологии;
- геоинформационные технологии;
- технологии защиты информации;
- CASE-технологии;
- телекоммуникационные технологии;
- технологии искусственного интеллекта.

Специфика конкретной предметной области находит отражение в специализированных прикладных информационных технологиях, например организационное управление, управление технологическими процессами, автоматизированное проектирование, обучение и др. Среди них наиболее продвинутыми являются следующие:

- информационные технологии организационного управления (корпоративные информационные технологии);
- информационные технологии в промышленности и экономике;
- информационные технологии в образовании;
- информационные технологии автоматизированного проектирования.

Аналогом инструментальной базы (оборудование, станки, инструмент) являются средства реализации ин-

формационных технологий, которые можно разделить на методические, информационные, математические, алгоритмические, технические и программные.

CASE-технология (Computer Aided Software Engineering — компьютерное автоматизированное проектирование программного обеспечения) является своеобразной «технологической оснасткой», позволяющей осуществить автоматизированное проектирование информационных технологий.

Методические средства определяют требования при разработке, внедрении и эксплуатации информационных технологий, обеспечивая информационную, программную и техническую совместимость. Наиболее важными из них являются требования по стандартизации.

Информационные средства обеспечивают эффективное представление предметной области, к их числу относятся информационные модели, системы классификации и кодирования информации (общероссийские, отраслевые) и др.

Математические средства включают в себя модели решения функциональных задач и модели организации информационных процессов, обеспечивающие эффективное принятие решения. Математические средства автоматически переходят в алгоритмические, обеспечивающие их реализацию.

Технические и программные средства задают уровень реализации информационных технологий как при их создании, так и при реализации.

Таким образом, конкретная информационная технология определяется в результате компиляции и синтеза базовых технологических операций, «отраслевых технологий» и средств реализации.

Эволюция информационных технологий наиболее ярко прослеживается на процессах хранения информации, транспортирования и обработки информации.

В управлении данными, объединяющем задачи получения, хранения, управления, анализа и визуализации данных, выделяют шесть временных фаз (поколений) [7], которые представлены на рисунке 1.7. В начале дан-

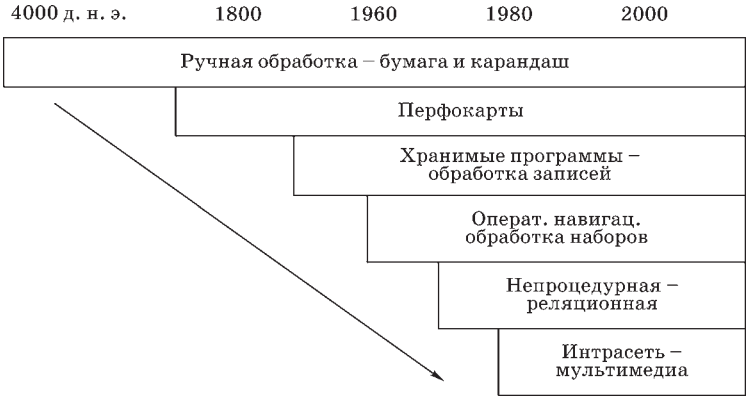


Рис. 1.7
Временные фазы развития управления данными

ные обрабатывались вручную. На следующем шаге использовались оборудование с перфокартами и электромеханические машины для сортировки и табулирования миллионов записей. В третьей фазе данные хранились на магнитных лентах и сохраняемые программы выполняли пакетную обработку последовательных файлов. Четвертая фаза ввела понятия схемы базы данных и оперативного навигационного доступа к данным. В пятой фазе был обеспечен автоматический доступ к реляционным базам данным и внедрена распределенная и клиент-серверная обработка. Теперь мы находимся в начале шестого поколения систем, которые хранят более богатые типы данных, в особенности документы, графические, звуковые и видеообразы. Эти системы шестого поколения представляют собой базовые средства хранения для появляющихся приложений Интернет и Интранет.

Для нулевого поколения (менеджеры записей, 4000 г. до н. э. — 1900) в течение шести тысяч лет наблюдалась эволюция от глиняных таблиц к папирусу, затем к пергаменту и, наконец, к бумаге. Имелось много новшеств в представлении данных: фонетические алфавиты, сочинения, книги, библиотеки, бумажные и печатные издания. Это были большие достижения, но обработка информации в эту эпоху производилась вручную.

Первое поколение (менеджеры записей, 1900–1955) связано с технологией перфокарт, когда запись данных представлялась в виде двоичных структур на перфокарте. Процветание компании IBM в период с 1915 до 1960 г. связано с производством электромеханического оборудования для записи данных на карты, сортировки и составления таблиц. Громоздкость оборудования, необходимость хранения громадного количества перфокарт предсказывали появления новой технологии, которая должна была вытеснить электромеханические компьютеры.

Второе поколение (программируемое оборудование обработки записей, 1955–1980) связано с появлением технологии магнитных лент, каждая из которых могла хранить информацию десяти тысяч перфокарт. Для обработки информации были разработаны электронные компьютеры с хранимыми программами, которые могли обрабатывать сотни записей в секунду. Ключевым моментом этой новой технологии было программное обеспечение, с помощью которого сравнительно легко можно было программировать и использовать компьютеры.

Программное обеспечение этого времени поддерживало модель обработки записей на основе файлов. Типичные программы последовательно читали несколько входных файлов и производили на выходе новые файлы. Для облегчения определения этих ориентированных на записи последовательных задач были созданы COBOL и несколько других языков программирования. Операционные системы обеспечивали абстракцию файла для хранения этих записей, язык управления заданиями для выполнения заданий и планировщик заданий для управления потоком работ.

Системы пакетной обработки транзакций сохраняли транзакции на картах или лентах и собирали их в пакеты для последующей обработки. Раз в день эти пакеты транзакций сортировались. Отсортированные транзакции сливались с хранимой на ленте намного большей по размерам базой данных (основным файлом) для производства нового основного файла. На основе этого основного файла также производился отчет, который использовался как

гроссбух на следующий бизнес-день. Пакетная обработка позволяла очень эффективно использовать компьютеры, но обладала двумя серьезными ограничениями: невозможность распознавания ошибки до обработки основного файла и отсутствие оперативного знания о текущей информации.

Третье поколение (оперативные базы данных, 1965–1980) связано с внедрением оперативного доступа к данным в интерактивном режиме, основанном на использовании систем баз данных с оперативными транзакциями.

Технические средства для подключения к компьютеру интерактивных компьютерных терминалов прошли путь развития от телетайпов к простым алфавитно-цифровым дисплеям и, наконец, к сегодняшним интеллектуальным терминалам, основанным на технологии персональных компьютеров.

Оперативные базы данных хранились на магнитных дисках или барабанах, которые обеспечивали доступ к любому элементу данных за доли секунды. Эти устройства и программное обеспечение управления данными давали возможность программам считывать несколько записей, изменять их и затем возвращать новые значения оперативному пользователю. Вначале системы обеспечивали простой поиск данных: либо прямой поиск по номеру записи, либо ассоциативный поиск по ключу.

Простые индексно-последовательные организации записей быстро развились к более мощной модели записей, ориентированной на наборы. Эволюция моделей данных прошла путь от иерархических и сетевых моделей к реляционным.

В этих ранних базах данных поддерживались три вида схем данных:

- логическая схема, которая определяет глобальный логический проект записей базы данных и связей между записями;
- физическая схема, описывающая физическое размещение записей базы данных на устройствах памяти и в файлах, а также индексы, нужные для поддержки логических связей;

- предоставляемая каждому приложению подсхема, раскрывающая только часть логической схемы, которую использует программа. Механизм логических, физических и подсхем обеспечивал независимость данных. И на самом деле многие программы, написанные в ту эпоху, все еще работают сегодня с использованием той же самой подсхемы, с которой все начиналось, хотя логическая и физическая схемы абсолютно изменились.

К 1980 г. сетевые (и иерархические) модели данных, ориентированные на наборы записей, стали очень популярны. Однако навигационный программный интерфейс был низкого уровня, что послужило толчком к дальнейшему совершенствованию информационных технологий.

Четвертое поколение (реляционные базы данных: архитектура «клиент-сервер», 1980–1995) являлась альтернативой низкоуровневому интерфейсу. Идея реляционной модели состоит в единообразном представлении сущности и связи. Реляционная модель данных обладает унифицированным языком для определения данных, навигации по данным и манипулирования данными. Работы в этом направлении породили язык, названный SQL.

Сегодня почти все системы баз данных обеспечивают интерфейс SQL. Кроме того, во всех системах поддерживаются собственные расширения, выходящие за рамки этого стандарта.

Реляционная модель обладает некоторыми неожиданными преимуществами, кроме повышения продуктивности и простоты использования. Реляционная модель оказалась хорошо пригодной к использованию в архитектуре «клиент-сервер», параллельной обработке и графическим пользовательским интерфейсам. Приложение «клиент-сервер» разбивается на две части. Клиентская часть отвечает за поддержку ввода и представление выводных данных для пользователя или клиентского устройства. Сервер отвечает за хранение базы данных, обработку клиентских запросов к базе данных, возврат клиенту общего ответа. Реляционный интерфейс особенно удобен для использования в архитектуре «клиент-сервер», поскольку приво-

дит к обмену высокоуровневыми запросами и ответами. Высокоуровневый интерфейс SQL минимизирует коммуникации между клиентом и сервером. Сегодня многие клиент-серверные средства строятся на основе протокола Open Database Connectivity (ODBC) , который обеспечивает для клиента стандартный механизм запросов высокого уровня к серверу. Парадигма «клиент-сервер» продолжает развиваться. Как разъясняется в следующем разделе, имеется возрастающая тенденция интеграции процедур в серверах баз данных. В частности, такие процедурные языки, как BASIC и Java, были добавлены к серверам, чтобы клиенты могли вызывать прикладные процедуры, выполняемые на сервере.

Параллельная обработка баз данных была вторым неожиданным преимуществом реляционной модели. Отношения являются однородными множествами записей. Реляционная модель включает набор операций, замкнутых по композиции: каждая операция получает отношения на входе и производит отношение как результат. Поэтому реляционные операции естественным образом предоставляют возможности конвейерного параллелизма путем направления вывода одной операции на вход следующей.

Реляционные данные также хорошо приспособлены к графическим пользовательским интерфейсам (GUI). Пользователи легко могут создавать отношения в виде электронных таблиц и визуально манипулировать ими.

Между тем файловые системы и системы, ориентированные на наборы, оставались рабочими лошадками многих корпораций. С годами эти корпорации построили громадные приложения и не могли легко перейти к использованию реляционных систем. Реляционные системы скорее стали ключевым средством для новых клиент-серверных приложений.

Пятое поколение (мультимедийные базы данных, 1995 г. — по настоящее время) связано с переходом от традиционных баз данных, хранящих числа и символы, к объектно-реляционным базам данных, содержащих данные со сложным поведением. Например, географам следует иметь возможность реализации карт, специали-

стам в области текстов имеет смысл реализовывать индексацию и выборку текстов, специалистам по графическим образам стоило бы реализовать библиотеки типов для работы с образами. Конкретным примером может служить распространенный объективный тип временных рядов. Вместо встраивания этого объекта в систему баз рекомендуется реализация соответствующего типа в виде библиотеки классов с методами для создания, обновления и удаления временных рядов.

Быстрое развитие Интернета усиливает эти дебаты. Клиенты и серверы Интернета строятся с использованием апплетов и «хелперов», которые сохраняют, обрабатывают и отображают данные того или иного типа. Пользователи вставляют эти апплеты в браузер или сервер. Общераспространенные апплеты управляют звуком, графикой, видео, электронными таблицами, графами. Для каждого из ассоциированных с этими апплетами типов данных имеется библиотека классов. Настольные компьютеры и веб-браузеры являются распространенными источниками и приемниками большей части данных. Поэтому типы и объектные модели, используемые в настольных компьютерах, будут диктовать, какие библиотеки классов должны поддерживаться на серверах баз данных.

Подытожим: базы данных призваны хранить больше, чем только числа и текстовые строки. Они используются для хранения многих видов объектов, что мы видим в World Wide Web, и связей между этими объектами. Различие между базой данных и остальной частью Web становится неясным. Каждый поставщик баз данных обещает «универсальный сервер», который будет хранить и анализировать все формы данных (все библиотеки классов и соответствующие объекты).

Чтобы приблизиться к современному состоянию технологии управления данными, имеет смысл описать два крупных проекта управления данными, в которых используются предельные возможности сегодняшней технологии [7]. Система Earth Observation System/Data Information System (EOS/DIS) разрабатывается агент-

ством NASA и его подрядчиками для хранения всех спутниковых данных, которые начали поступать со спутников серии «Миссия к планете Земля» в 1977 г. Объем базы данных, включающей данные от удаленных сенсорных датчиков, будет расти на 5 Тбайт в день (Тбайт — это 1 млн Гбайт). К 2007 г. размер базы данных вырос до 15 петабайт. Это в тысячу раз больше объема самых больших сегодняшних оперативных баз данных. NFSA желает, чтобы эта база данных была доступна каждому в любом месте в любое время. Любой человек сможет производить поиск, анализ и визуализацию данных из этой базы данных. Для построения EOS/DIS потребуются наиболее развитые методы хранения, поиска и визуализации данных. Большая часть данных будет обладать пространственными и временными характеристиками, так что для системы потребуются существенное развитие технологии хранения данных этих типов, а также библиотеки классов для различных научных наборов данных. Например, для этого приложения потребуется библиотека для определения снежного покрова, каталога растительных форм, анализа облачности и других физических свойств образов LandSat. Эта библиотека классов должна легко подключаться к менеджеру данных SOS/DIS.

Другим впечатляющим примером базы данных является возникающая всемирная библиотека. Многие ведомственные библиотеки открывают доступ к своим хранилищам в режиме онлайн. Новая научная литература публикуется в режиме онлайн. Такой вид публикации поднимает трудные социальные вопросы по поводу авторских прав и интеллектуальной собственности и заставляет решать сложные технические проблемы. Пугают размеры и многообразие информации. Информация появляется на многих языках, во многих форматах данных и в громадных объемах. При применении традиционных подходов к организации такой информации (автор, тема, название) не используются мощности компьютеров для поиска информации по содержимому, для связывания документов и группирования сходных документов. Обнаружение информации, нахождение требуемой информации в море до-

кументов, карт, фотографий, звука и видео представляет собой захватывающую и трудную проблему.

Быстрое развитие технологий хранения информации, коммуникаций и обработки позволяет переместить всю информацию в киберпространство. Программное обеспечение для определения, поиска и визуализации оперативно доступной информации — ключ к созданию и доступу к такой информации. Основные задачи, которые необходимо решить:

- определение моделей данных для новых типов (например, пространственных, темпоральных, графических) и их интеграция с традиционными системами баз данных;
- масштабирование баз данных по размеру (до петабайт), пространственному размещению (распределенные) и многообразию (неоднородные);
- автоматическое обнаружение тенденций данных, структур и аномалий (добывание данных, анализ данных);
- интеграция (комбинирование) данных из нескольких источников;
- создание сценариев и управление потоком работ (процессом) и данными в организациях;.
- автоматизация проектирования и администрирования базами данных.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем заключается понятие информации?
2. Какие существуют виды иерархии информации?
3. В чем суть информационного подхода к процессу управления?
4. Дайте сравнительную характеристику свойств информации.
5. Какими факторами определяется ценность информации?
6. Каковы основные аспекты количественной оценки информации?
7. Укажите основные меры информации.

8. В чем сущность статистической меры количества информации?
9. Укажите основные свойства энтропии.
10. В чем отличие энтропии источника от энтропии сообщения?
11. На чем базируется семантический подход к оценке содержания информации?
12. Что такое тезаурус?
13. Дайте сравнительную характеристику семантического и прагматического подходов к оценке информации.
14. Что такое информационный ресурс?
15. Назовите основные составные части знаний.
16. В чем коммерческая сущность информации?
17. Укажите основные уровни информатики.
18. В чем суть декомпозиции информации?
19. Поясните суть понятия информации.
20. Что такое абстрагирование информации и каковы его основные способы?
21. Что такое агрегирование информации?
22. Дайте определение информационной технологии и поясните ее содержание.
23. От чего зависит эффективность информационных технологий?
24. Перечислите основные уровни рассмотрения информационных технологий.
25. Что такое базовый технологический процесс? Перечислите их.
26. Раскройте содержание прикладного уровня информационных технологий.
27. Выделите основные фазы (поколения) эволюции информационных технологий.

ГЛАВА 2

БАЗОВЫЕ ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ, ИХ ХАРАКТЕРИСТИКА И МОДЕЛИ

В результате освоения данной темы студент должен

1) знать:

- основные фазы извлечения информации, формы и методы исследования данных, способы и методы извлечения знаний;
- способы и методы поиска информации в сети Интернет, в том числе на основе нейронных сетей и онтологии;
- эталонную семиуровневую модель связи открытых систем;
- протоколы сетевого взаимодействия, требования к компьютерной сети;
- базовые сети и их роль в обеспечении качества обслуживания;
- режимы и способы обработки данных;
- процесс принятия решения;
- состав и структуру системы поддержки принятия решений;
- основные технологии интеллектуального анализа данных;
- направления реализации процессов хранения и поиска информации;
- состав и структуру базы данных;
- уровни описания предметной области;
- свойства логических моделей данных;
- состав и структуру системы управления базой данных;
- процесс проектирования базы данных;
- принципы построения объектно-ориентированных и объектно-реляционных баз данных;

- принципы построения и способы реализации интерфейса информационных систем;

2) уметь:

- разрабатывать информационно-логическую, функциональную и объектно-ориентированную модели информационной системы, модели данных информационных систем;
- применять информационные технологии при реализации базовых информационных процессов;
- использовать архитектурные и детализированные решения при реализации базовых информационных процессов;
- осуществлять математическую и информационную постановку задач по обработке информации, использовать алгоритмы обработки информации для различных приложений;
- проводить предпроектное обследование (инжиниринг) объекта проектирования, системный анализ предметной области, их взаимосвязей, проводить выбор исходных данных для выбора базовых информационных процессов;

3) владеть:

- методами и средствами представления данных и знаний о предметной области, методами и средствами анализа и синтеза базовых информационных процессов;
- методологией использования информационных технологий при реализации базовых информационных процессов;
- моделями и средствами разработки базовых информационных процессов;
- технологиями построения и сопровождения инфокоммуникационных систем и сетей.

2.1. ИЗВЛЕЧЕНИЕ ИНФОРМАЦИИ

Источниками информации могут являться данные, знания, документы. Источниками данных в любой предметной области являются объекты и их свойства, процес-

сы и функции, выполняемые этими объектами или для них. Любая предметная область рассматривается в виде трех представлений (рис. 2.1).

Непосредственно в процессе извлечения информации можно выделить следующие фазы [8]:

- накопление — системное или бессистемное (стихийное) накопление информации в рамках предметной области;
- структурирование — выделение основных понятий, выработка структуры представления информации, обладающей максимальной наглядностью, простотой изменения и дополнения;
- формализация — представление структурированной информации в форматах машинной обработки, т. е. на языках описания данных и знаний;
- обслуживание — корректировка формализованных данных и знаний (добавление, обновление), удаление устаревшей информации, фильтрация данных и знаний для поиска информации, необходимой пользователям.

По аналогии с добычей полезных ископаемых процесс извлечения информации направлен на получение наибольшей ее концентрации. В связи с этим процесс извлечения можно представить как ее прохождение через трехслойный фильтр, в котором осуществляется оценка синтаксической ценности (правильность представления),

Рис. 2.1
Предметная область в виде трех представлений

семантической (смысловой) ценности, прагматической (потребительской) ценности.

При извлечении информации важное место занимают различные формы и методы исследования данных:

- нахождение ассоциаций, связанных с привязкой к какому-либо событию;
- нахождение последовательностей событий во времени;
- нахождение скрытых закономерностей по наборам данных путем определения причинно-следственных связей между значениями определенных косвенных параметров исследуемого объекта (ситуации, процесса);
- оценка важности (влияния) параметров на события и ситуации;
- классифицирование (распознавание), осуществляемое путем поиска критериев, по которым можно было бы относить объект (события, ситуации, процессы) к той или иной классификационной категории;
- кластеризация, основанная на группировании объектов по каким-либо признакам;
- прогнозирование событий и ситуаций.

Следует упомянуть неоднородность (разнородность) информационных ресурсов, характерную для многих предметных областей. Одним из путей решения данной проблемы является объектно-ориентированный подход, наиболее распространенный в настоящее время. Кратко рассмотрим его основные положения.

Декомпозиция на основе объектно-ориентированного подхода основана на выделении следующих основных понятий: «объект», «класс», «экземпляр».

Объект — это абстракция множества предметов реального мира, обладающих одинаковыми характеристиками и законами поведения. Объект характеризует собой типичный неопределенный элемент такого множества. Основной характеристикой объекта является состав его атрибутов (свойств).

Атрибуты — это специальные объекты, посредством которых можно задать правила описания свойств других объектов.

Экземпляр объекта — это конкретный определенный элемент множества. Например, объектом может являться государственный номер автомобиля, а экземпляром этого объекта — конкретный номер К173ПА.

Класс — это множество предметов реального мира, связанных общностью структуры и поведением. Элемент класса — это конкретный элемент данного множества. Например, класс регистрационных номеров автомобиля.

Обобщая эти определения, можно сказать, что объект — это типичный представитель класса, а термины «экземпляр объекта» и «элемент класса» равнозначны. На рисунке 2.2 показаны отношения между классами, объектами и предметами реального мира.

Важная особенность объектно-ориентированного подхода связана с понятием инкапсуляции, обозначающим сокрытие данных и методов (действий с объектом) в качестве собственных ресурсов объекта.

Понятия полиморфизма и наследования определяют эволюцию объектно-ориентированной системы, что подразумевает определение новых классов объектов на основе базовых классов.

Полиморфизм интерпретируется как способность объекта принадлежать более чем одному типу.

Наследование выражает возможность определения новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

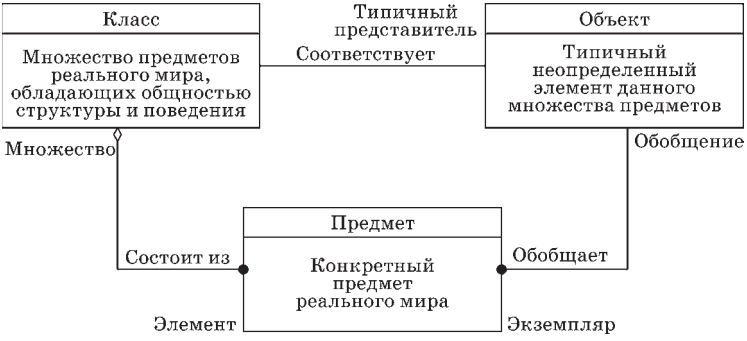


Рис. 2.2
Отношения между классами, объектами и предметами реального мира

Для уменьшения избыточности используется процесс обогащения информации, например при хранении в компьютере списка сотрудников организации иногда достаточно использовать первые 3–4 буквы их фамилий.

Среди методов обогащения информации различают структурное, статистическое, семантическое и прагматическое обогащения.

Структурное обогащение предполагает изменение параметров сообщения, отображающего информацию в зависимости от частотного спектра исследуемого процесса, скорости обслуживания источников информации и требуемой точности.

При статистическом обогащении осуществляют накопление статистических данных, обработку выборок из генеральных совокупностей накопленных данных.

Семантическое обогащение означает минимизацию логической формы, исчислений и высказываний, выделение и классификацию понятий, содержания информации, переход от частных понятий к более общим. В итоге семантического обогащения удается обобщенно представить обрабатываемую либо передаваемую информацию и устранить логическую противоречивость в ней.

Прагматическое обогащение является важной ступенью при использовании информации для принятия решения, при котором из полученной информации отбирается наиболее ценная, отвечающая целям и задачам пользователя.

Развитие методов и средств извлечения информации направлено в сторону стандартизации и унификации. Характерным примером является создание и внедрение технологий Data Mining и Text Mining.

Data Mining (в буквальном переводе с английского — добыча данных) — это направление в информационных технологиях, которое связано с автоматизированным извлечением знаний (неявным образом присутствующих в обрабатываемой информации) и базируется на интеллектуальном анализе данных.

В основе современной технологии Data Mining лежит концепция шаблонов, отражающих различные фрагмен-

ты взаимоотношений в данных. Важное свойство методов Data Mining — нетривиальность обнаруживаемых шаблонов, которые должны отражать неочевидные, ранее неизвестные регулярности в данных, составляющие так называемые скрытые знания (hidden knowledge), например, в банковском деле — изменение клиентуры, выявление мошенничества с кредитными карточками.

Методы Data Mining позволяют выделить следующие типы закономерностей:

- последовательность (например, после события А в течение определенного интервала времени с большой вероятностью следует событие Б);
- связь между событиями (например, события А и Б с большой вероятностью осуществляются одновременно);
- классификация (объекты относятся к одной из групп с относительно постоянными характеристиками);
- кластеризация отличается от классификации тем, что сами группы заранее не задаются и выделяются непосредственно в процессе анализа;
- прогноз — построение временных рядов, отражающих динамику поведения целевых показателей.

Text Mining. Является разновидностью Data Mining, ориентированой на обработку текстовой информации и широко применяемой для мониторинга ресурсов Интернета. Задача Text Mining — проанализировать не синтаксис, а семантику текстов, выбрать из них информацию, наиболее значимую для пользователя (есть тесная связь с контент-анализом). Обычно выделяют такие приложения Text Mining:

- реферирование текстов на естественном языке;
- классификация (тематическое индексирование) текстовых документов;
- кластеризация текстовых документов и их фрагментов;
- построение онтологии текстового документа (основных терминов и связей между ними), например семантической сети;
- визуализация полученных знаний.

В настоящее время сеть Интернет становится одним из основных поставщиков информации. Объем Сети неуклонно растет, пополняясь не только персональными страничками, но и переведенными в электронный вид различными базами знаний, как то, например: фонды библиотек, музеев искусств, электронные версии бумажной прессы. Помимо оцифрованных изданий, свою лепту в рост Сети вкладывают и интернет-порталы различной тематики, объем информации которых уже превышает сотни гигабайт.

Поиск информации в сети Интернет сопряжен с целым рядом технических проблем. Среди них: различные форматы представления документов, работа со слабоструктурированной информацией, необходимость обработки документов на разных языках и учет языковых особенностей, большие и быстрорастущие массивы информации, необходимость высокой скорости поиска документов и навигационные методы.

Существует широкий спектр методов поиска информации в сети Интернет на основе информационно-поисковых систем (ИПС), которые, однако, можно подразделить на два основных класса (рис. 2.3): поисковые машины и поисковые каталоги. Рассмотрим их основные достоинства и недостатки [8].

Серверная поисковая машина — это программно-аппаратный комплекс высокой производительности, нацеленный на обслуживание множества клиентов одновременно. Следствиями его высокой производительности являются: малое время отклика и обработки запроса.

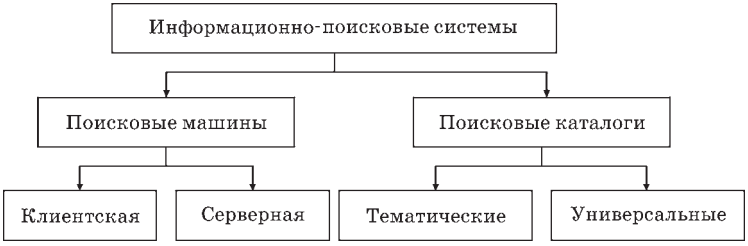


Рис. 2.3
Классификация информационно-поисковых систем

Программное обеспечение (ПО), установленное на сервере, обычно использует все современные возможности поиска информации, в силу высоких аппаратных характеристик сервера.

Клиентская поисковая машина, или *интеллектуальный агент*, — это ПО, предназначенное для поиска информации и установленное на компьютере клиента либо загруженное из Сети и работающее на стороне клиента. По сравнению с серверным вариантом является более узкоспециализированным (ищет ссылки только в определенном секторе Сети), менее быстрым, но при этом более гибким в настройке. В ближайшем будущем появятся агенты, адаптирующиеся к кругу интересов пользователя (например, при помощи нейросетевых или других алгоритмов) и добивающиеся поэтому лучших результатов в поиске. Также такие интеллектуальные агенты могли бы отправлять наиболее популярные ссылки своего владельца на основной поисковой сервер, что повысило бы качество поиска и серверной машины.

Поисковые машины обеспечивают автоматическую индексацию большого количества документов, но не обладают развитыми средствами искусственного интеллекта для экспертной оценки смыслового содержания информации. Этим обусловлена низкая релевантность ответа поисковых систем (релевантность — степень адекватности результатов поиска запросу пользователя). Решение данной проблемы заключается в применении прогрессивных методов искусственного интеллекта для обработки и анализа текстовой информации.

Поисковые каталоги ресурсов представляют собой иерархически организованные наборы резюме содержания информационных ресурсов. Каталоги позволяют пользователю, спускаясь от общих понятий к более узким, найти ссылку на сайт с интересующей их информацией. Преимущество таких систем перед поисковыми машинами заключается в том, что база данных каталогов наполняется людьми, что приводит к высокой релевантности расположенных в них ссылок. Существуют каталоги двух типов: универсальные и тематические. Как показыва-

ет практика, хорошие тематические каталоги содержат больше информации по своей тематике, чем универсальные. Однако информационная полезность таких каталогов, как правило, ограничена небольшим количеством проиндексированных документов, большими затратами средств на поддержание актуальности базы проиндексированных документов и, следовательно, низкой оперативностью ее обновления.

Объем базы данных каталогов сравнительно невелик, скорость пополнения базы на порядок ниже, чем у автоматизированных поисковых машин. Также ниже и скорость поиска по каталогу.

Одним из способов устранения вышеизложенных недостатков каталогов, а также поисковых машин является их объединение в общую структуру. Это позволяет придать такой гибридной структуре скорость поиска поисковых машин в сочетании с точностью (релевантностью) каталогов.

Рассмотрим процесс формирования информационных ресурсов и их представление в информационно-поисковой системе (ИПС). Общеизвестно, что документальным массивом ИПС Интернета является все множество документов шести основных типов: www-страницы, Gopher-файлы, документы Wais, записи архивов FTP, новости Usenet, статьи почтовых списков рассылки. Все это довольно разнородная информация, которая представлена в виде различных, никак несогласованных друг с другом форматов данных. Здесь есть и текстовая информация, и графическая информация, и аудиоинформация, и вообще все, что есть в указанных выше хранилищах. Естественно встает вопрос: как информационно-поисковая система должна со всем этим работать?

Первая задача, которою должна решить информационно-поисковая система — это приписывание списка ключевых слов документу или информационному ресурсу. Именно эта процедура и называется индексированием.

Часто, однако, индексированием называют составление файла инвертированного списка, в котором каждому термину индексирования ставится в соответствие список

документов, в которых он встречается. Такая процедура является только частным случаем, а точнее, техническим аспектом создания поискового аппарата информационно-поисковой системы.

Одним из наиболее важных факторов, влияющих на качество поиска, является метод внутреннего представления документов в поисковой машине. В традиционных системах есть понятие поискового образа документа (ПОД), который его заменяет и используется при поиске вместо реального. Поисковый образ является результатом применения некоторой модели информационного массива документов к реальному массиву.

Обычно поиск информации в документах происходит путем сравнения терминов этих документов с терминами из запроса пользователя. В этих методах есть два существенных недостатка. Во-первых, обычно имеется много способов выражения данного понятия (с помощью синонимов), поэтому относящиеся к делу документы могут быть отвергнуты. Во-вторых, многие слова имеют множественное значение (полисемия), поэтому в результате работы программы могут быть получены ненужные документы. Эти два недостатка приводят к тому, что методы, основанные на сравнении терминов, оказываются неприемлемыми для поиска ответа на запрос пользователя. Более эффективный подход должен позволять пользователю получать информацию, учитывая смысл конкретного документа.

Существует ряд подобного рода методов, рассмотрим наиболее распространенные из них.

Лексическое индексирование. В основе лексического индексирования лежит булева модель. Запросы пользователя представляют собой некоторое логическое выражение, в котором ключевые слова соединены операторами AND, NOT или ANDNOT (редко).

При использовании этой модели индекс организуется в виде инвертированного файла, в котором для каждого термина из словаря коллекции хранится список документов, в которых этот термин встречается.

Данный тип индексирования достаточно хорошо распространен, но при этом имеет существенные недостатки.

Так как поиск ведется при помощи логических объединений/пересечений документов, в которых имеются ключевые слова, то результат поиска является полностью бесконтекстным, что сильно понижает его релевантность.

Векторное индексирование. Наиболее популярной моделью является векторная модель, в которой каждому документу приписывается список терминов, наиболее адекватно отражающих его смысл. В данной модели запрос пользователя, так же как и документы, представляется в виде вектора в базисе слов словаря. Наиболее релевантными считаются те документы, углы векторов которых с вектором запроса минимальны. Если быть более точным, то документу приписывается вектор, размерность которого равна числу терминов, которыми можно воспользоваться при поиске. При булевой векторной модели элемент вектора равен единице или нулю, в зависимости от наличия термина в ПОДе документа или его отсутствия. В более сложных моделях термины взвешиваются, т. е. элемент вектора равен не единице или нулю, а некоторому числу, которое отражает соответствие данного термина документу. Именно эта модель наиболее популярна в информационно-поисковых системах Интернета.

Вероятностное индексирование. Данный вид индексирования сопоставляет каждому слову его вес в документе. Это приводит к значительному повышению качества поиска по сравнению с лексическим и векторным индексированием.

Скрытое семантическое индексирование. Математический аппарат данного метода базируется на экономном сингулярном разложении матриц, которое позволяет выявить скрытые семантические связи при обработке большой коллекции документов.

Теоретическая эффективность метода намного выше лексического или векторного индексирования, но из-за высоких требований к вычислительным возможностям сервера применение его затруднено.

Использование моделей семантического анализа (MSA) является попыткой преодоления проблемы сравнения терминов с использованием статистически получен-

ных смысловых параметров вместо отдельных слов. В методе MSA предполагается, что в каждом образце текста имеется некоторая внутренняя скрытая структура, которая не совсем ясна ввиду возможного использования синонимов. Эта структура фиксируется матрицей терминов и документов, которая представляет собой разреженную (т. е. имеется сравнительно немного ненулевых элементов) матрицу строения $m \times n$, получаемую грамматическим анализом текста. Для анализа структуры использования слов в документах используется сингулярное разложение (SVD). Поиск документов может быть осуществлен путем использования k наибольших сингулярных значений и соответствующих сингулярных векторов, где $k \ll \min(m, n)$. Проведенный анализ показывает, что сингулярные векторы в действительности являются более надежными показателями смысла, чем отдельные слова. SVD является наиболее распространенным примером двустороннего (или полного) ортогонального разложения, в котором матрица представляется в виде произведения трех других матриц: ортогональной, средней и еще одной ортогональной. Средняя матрица — это нижне- (верхне-) трапециевидная или диагональная. Однако среди двусторонних ортогональных разложений, которые могут быть использованы для поиска информации, более эффективным является применение метода MSA. К важному преимуществу данного метода относится тот факт, что количество вычислений по сравнению с SVD уменьшается. Основные вычислительные преимущества MSA над другими методами заключаются в другом подходе к добавлению информации в базу данных. Основная идея в MSA заключается в том, чтобы явно смоделировать взаимосвязи между терминами (через двустороннее ортогональное разложение) и использовать его, чтобы улучшить возможности поисковой системы.

Новой парадигмой построения поисковых механизмов является применение систем нейронных сетей (Neural Network System) и онтологий для поиска документов по запросам пользователей в коллекциях и при объединении результатов поиска серверами запросов. При кластери-

зации локальной коллекции в фоновом режиме профайлы, представляющие локальную коллекцию, становятся входными данными для нейронной сети. Нейронная сеть затем строит кластерное дерево: несколько кластеров верхнего уровня, группу субкластеров для каждого из кластеров верхнего уровня и так далее до отдельных документов. Для осуществления поиска в коллекции посредством запросов каждый кластер представлен документом, наиболее близко находящимся к центроиду кластера (cluster centroid) в векторном пространстве профайлов. Таким образом, профайл заданного запроса необходимо сравнить только с профайлами кластерного центроида. Это в значительной степени увеличивает скорость обработки запросов. Кроме этого, нейронные сети могут использоваться для кластеризации результатов поиска, поступивших в ответ на запрос от нескольких серверов. Главная задача кластеризации — выдать пользователю репрезентативный набор результатов, если общее результирующее число документов превышает «порог», заданный пользователем.

Известны два типа архитектуры нейронных сетей: RCL (Radius-based Competitive Learning — основанную на радиусе, обучающуюся, соревновательного типа) и ее иерархическое расширение, называемое HRCL (Hierarchical Radius-based Competitive Learning — иерархическую, основанную на радиусе, обучающуюся, соревновательного типа). В их основе лежит идея о том, что после каждого введения в систему входного вектора все нейроны упорядочиваются по их расстоянию к точке текущего ввода. Самый близкий к текущему вводу RCL нейрон становится победителем, кроме этого, RCL адаптирует все остальные нейроны из нейронного набора.

Высокая производительность и универсальность подсистемы нейронных сетей дает все основания для предположения о том, что она будет играть значительно большую роль в поисковых системах.

В настоящий момент наметилась тенденция к стандартизации описания структурированных, неструктурированных и полуструктурированных текстов при помощи

XML-технологии, что позволяет наметить пути к созданию единой технологии их обработки.

Представление данных как XML-документов является естественным, поскольку они получаются из реальных документов. Представлять данные как документы привычнее и понятнее, чем представлять их как реляционные таблицы. Реляционная таблица — в лучшем случае отдельный фрагмент документа. Неестественность табличного представления легко прочувствовать вначале при проектировании реляционных баз данных, когда из набора имеющихся документов происходит вычленение сущностей, и затем при подготовке отчета, когда из этих же сущностей вновь создаются документы. Манипулировать данными с использованием такой естественной для человека (но логически избыточной!) сущности, как «связь», также привычнее и понятнее, чем со ссылочными ключами, которые в реальных документах встречаются редко.

IBM разрабатывает базирующуюся на XML систему поиска данных — архитектуру управления неструктурированной информацией (UIMA), которая, как предполагается, значительно расширит возможности средств поиска, применяемых в базах данных. По замыслу UIMA — это нечто, что становится частью базы данных или, скорее, тем, к чему базы данных обращаются, при этом появляется возможность «обдумывать» что-нибудь почти непрерывно. Это значительно изменит автоматизированные или человеко-машинные системы. Например, предполагается, что станет реальностью автоматический перевод с языка на язык и работа с естественными языками.

В основе UIMA лежит теория сочетания гипотез (Combination Hypothesis), которая утверждает, что в ближайшем будущем появится возможность объединить статистическое обучение машины — вроде того, которое использует поисковый сайт Google для интеллектуального ранжирования данных, — синтетический искусственный интеллект и другие методы. Между тем XML обеспечивает простой способ обмена данными и их классификации, что облегчает использование искусственного интеллекта в вычислительной среде. По мнению представителей

IBM, благодаря появлению XML за ближайшие три года индустрия баз данных изменится сильнее, чем за предыдущие двадцать лет. По сути, искусственный интеллект будет функционировать как фильтр. Датчики собирают информацию о внешнем мире и передают ее в компьютер, который выполняет надлежащие действия, беспокоя владельца лишь в случае крайней необходимости. Если нужно найти что-то в веб, человек делает запрос, а компьютер помогает ему уточнить его таким образом, чтобы вышло не 14 страниц списка потенциальных веб-сайтов, а только требуемая информация. В такой ситуации ключевой проблемой является задача быстрого и максимально эффективного поиска, т. е. такого поиска информации, который позволит за минимальное время найти по запросу пользователя наиболее релевантные (подходящие) ресурсы. В настоящее время для решения этой проблемы пытаются применить механизм онтологий.

Онтологии используются для систематизации данных на корпоративном портале для индексации и удобного поиска — несмотря на то что многие крупные организации имеют собственную таксономию для организации внутренней информации, этого обычно недостаточно. Простая классификация сильно ограничивает возможности поиска и индексации, поскольку многие документы могут подпадать под разные категории, поэтому поиск по различным критериям будет намного эффективнее, чем обычный поиск по ключевым словам.

Семантическая сеть — развитие концепции существующей глобальной Сети. Всей информации в ней придается четко определенное значение, что позволяет компьютерам и людям осуществлять совместную работу с гораздо большей эффективностью. Чтобы придать информации четко определенное значение, нужно, в частности, создать язык онтологии, т. е. общий набор терминов, которые используются для описания и представления объектов в Интернете. Именно для этого и создается язык OWL (Ontology Web Language), разработку которого одобрил консорциум W3C. Новый язык поможет запустить автоматизированные инструменты для глобальной Сети нового поколения,

предлагая такие усовершенствованные услуги, как более точный веб-поиск, интеллектуальные программные агенты и управление знаниями.

Сегодня на переднем крае разработок в сфере интернет-стандартов находится Семантическая сеть (Semantic Web, согласно терминологии консорциума W3C), архитектура которой предполагает наличие у любой информации, находящейся в Сети, связанный с этой информацией точный смысл, который нельзя было бы перепутать даже в случае совпадения фраз или слов, встречающихся в разных контекстах. Фактически это означает, что любая информация связана с некоторым неотделимым от нее контекстом. Семантическая сеть активно использует язык XML для определения собственной структуры документов и язык RDF (Resource Definition Framework), предоставляющий удобную среду формализации метаданных и сведений о контексте. RDF создан консорциумом W3C и предназначен для описания метаданных, является подмножеством языка XML и имеет собственный язык RDF Schema для описания структуры документов. Однако RDF — это самый низкоуровневый из существующих языков описания метаданных, поскольку оперирует лишь понятиями связей примитивных сущностей, например «объект *A* владеет субъектом *B*». Со временем разработчикам Семантической сети стало очевидно, что средств XML и RDF для представления информации и метаданных для построения полноценной семантически связанной сети недостаточно. RDF подобен ассемблеру, если сравнивать семантическую нагрузку отдельных конструкций языка, и слишком сложен для решения задачи подобного масштаба. Чистый XML, в свою очередь, являясь метаязыком, включает в себя RDF как подмножество и не создан для какого-либо конкретного применения, а потому для построения Семантической сети его также недостаточно. Поэтому консорциумом W3C и был создан язык онтологий OWL (Web Ontology Language).

Онтология определяет термины, с помощью которых можно описать предметную область. Использование он-

тологий особенно необходимо в приложениях-агентах, осуществляющих поиск и объединение информации из различных источников и из разных сред, в которых один и тот же термин может означать разные вещи. Несмотря на то что DTD (Document Type Definition, формальное описание структуры XML-документов) в стандарте XML и схем XML (XML Schema) вполне достаточно для обмена данными между сторонами, которые заранее договорились о значении определений и терминов, отсутствие семантики в указанных средствах описания структуры серьезно ограничивает надежность выполнения задачи поиска и объединения данных при использовании новых XML-словарей. Например, элемент <Rate>, встретившийся индексатору в разных документах, может означать либо курс рубля, либо оценку за экзамен, либо цену товара, или что-нибудь еще. И без точной информации о том, что именно в конкретном документе имеется в виду под этим элементом, поисковый агент не сможет со стопроцентной вероятностью вернуть именно то, что и требовалось пользователю.

Практически любой пользователь Сети хотя бы раз сталкивался с ситуацией, когда при поиске интересующей его информации он помимо прочего получал от поисковой машины множество бесполезных ссылок. Поскольку поиск информации осуществляется вне контекста, никакие уточнения запросов не смогут надежно найти именно то, что нужно. Для качественного осуществления поиска пользователю необходимо понимать все тонкости предметной области, включая ее лексику, термины, определения, иерархии сущностей, другими словами, досконально знать онтологию. Хорошо, если пользователь является экспертом предметной области, хотя даже в этом случае будет найдена лишняя информация, а что делать обычным пользователям? Язык OWL призван упростить процесс поиска, возложив необходимость знания предметной области и описание контекста поиска полностью на авторов документа и систему поиска, причем передача этих функций авторам документа должна быть незаметна для пользователя.

2.2. ТРАНСПОРТИРОВАНИЕ ИНФОРМАЦИИ

Основным физическим способом реализации операции транспортировки является использование локальных сетей и сетей передачи данных. При разработке и использовании сетей для обеспечения совместимости используется ряд стандартов, объединенных в семиуровневую модель открытых систем, принятую во всем мире и определяющую правила взаимодействия компонентов сети на данном уровне (протокол уровня) и правила взаимодействия компонентов различных уровней (межуровневый интерфейс) [8], [9]. Международные стандарты в области сетевого информационного обмена нашли отражение в эталонной семиуровневой модели, известной как модель OSI (Open System Interconnection — связь открытых систем) (рис. 2.4). Данная модель разработана международной организацией по стандартизации (International Standards Organization — ISO). Большинство производителей сетевых программно-аппаратных средств стремятся придерживаться модели OSI. Но в целом добиться полной совместимости пока не удается.

Физический уровень. Реализует физическое управление и относится к физической цепи, по которой передаются биты информации, например телефонной. На этом уровне модель OSI определяет физические, электрические, функциональные и процедурные характеристики цепей связи, а также требования к сетевым адаптерам и модемам.

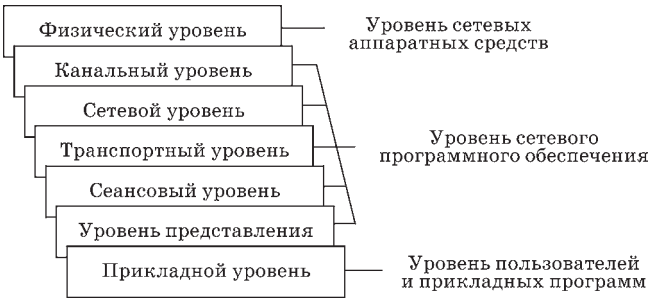


Рис. 2.4
Связь открытых систем

Канальный уровень. Этот уровень осуществляет управление звеном сети (каналом) и относится к пересылке блоков (совокупности битов) по физическому звену. Осуществляет такие процедуры управления, как определение начала и конца блока, обнаружение ошибок передачи, адресация сообщений и др. Канальный уровень определяет правила совместного использования сетевых аппаратных средств компьютерами сети.

Сетевой уровень. Относится к виртуальной (воображаемой) цепи, которая не обязана существовать физически. С помощью интерфейса, обеспечиваемого этим уровнем, удается «спрятать» сложности управления передачей на физическом уровне. Программные средства данного уровня обеспечивают определение маршрута передачи пакетов в сети. Маршрутизаторы, обеспечивающие поиск оптимального маршрута на основе анализа адресной информации, функционируют на сетевом уровне модели OSI. В качестве простейшего маршрутизирующего устройства между сегментами сети или различными локальными сетями может выступать и устройство, функционирующее на более низком канальном уровне модели OSI, называемое мостом.

Транспортный уровень. Первые три уровня образуют общую сеть, которую коллективно могут использовать многие пользователи. На транспортном уровне контролируется очередность пакетов сообщений и их принадлежность. Таким образом, в процессе обмена между компьютерами поддерживается виртуальная связь, аналогичная телефонной коммутации.

Сеансовый уровень. Из-за обилия способов взаимодействия между пользователями в некоторых случаях трудно организовать процесс взаимодействия между ними. Для устранения этих трудностей на данном уровне координируются и стандартизируются процессы установления сеанса, управления передачей и приемом пакетов сообщений, завершения сеанса. На сеансовом уровне между компьютерами устанавливается и завершается виртуальная связь по такому же принципу, как при голосовой телефонной связи.

Управление представлением. Программные средства этого уровня выполняют преобразования данных из внутреннего формата передающего компьютера во внутренний формат компьютера получателя, если эти форматы отличаются друг от друга (например, IBM PC и DEC). Данный уровень включает функции, относящиеся к используемому набору символов, кодированию данных и способам представления данных на экранах дисплеев или печати. Помимо конвертирования форматов, на данном уровне осуществляется сжатие передаваемых данных и их распаковка.

Прикладной уровень. Относится к функциям, которые обеспечивают поддержку пользователю на более высоком прикладном и системном уровне, например:

- организация доступа к общим сетевым ресурсам — информации, дисковой памяти, программным приложениям, внешним устройствам (принтерам, стримерам и др.);
- общее управление сетью (управление конфигурацией, разграничение доступа к общим ресурсам сети, восстановление работоспособности после сбоев и отказов, управление производительностью);
- передача электронных сообщений, включая электронную почту;
- организация электронных конференций;
- диалоговые функции высокого уровня.

Модель OSI представляет собой стандартизированный каркас и общие рекомендации, требования же к конкретным компонентам сетевого программного обеспечения задаются протоколами.

Протокол является стандартом в области сетевого программного обеспечения и определяет совокупность функциональных и эксплуатационных требований к какому-либо его компоненту, которых придерживаются производители этого компонента. Требования протокола могут отличаться от требований эталонной модели OSI.

Международный институт инженеров по электротехнике и радиоэлектронике (IEEE) разработал стандарты для протоколов передачи данных в локальных сетях. Эти

стандарты, которые описывают методы доступа к сетевым каналам данных, получили название IEEE 802.

Протоколы сетевого взаимодействия можно классифицировать по степени близости к физической среде передачи данных:

- протоколы нижнего уровня, распространяемые на канальный и физический уровни модели OSI;
- протоколы среднего уровня, распространяемые на сетевой, транспортный и сеансовый уровни OSI;
- протоколы верхнего уровня, распространяемые на уровень представления и прикладной уровень модели OSI.

Каждая из реализаций протоколов вышестоящих уровней использует в процессе своей работы реализации протоколов нижестоящих уровней.

Протоколы нижнего уровня OSI соответствуют уровню сетевых аппаратных средств и нижнему уровню сетевого программного обеспечения. Среди наиболее распространенных стандартов данного уровня выделим следующие [9]:

- стандарт NDIS (Network Driver Interface Specification — спецификация интерфейса сетевых драйверов), разработанный совместно фирмами Microsoft и 3Com;
- стандарт ODI (Open Datalink Interface — открытый интерфейс связи), разработанный совместно фирмами Novell и Apple Computer.

Данные стандарты позволяют реализовывать протоколы среднего уровня независимо от сетевых аппаратных средств и обеспечивают совместное функционирование разнотипных протоколов среднего уровня. Универсальный интерфейс канального уровня представлен на рисунке 2.5. Производители сетевых аппаратных средств, как правило, разрабатывают драйверы, удовлетворяющие обоим стандартам.

Драйвер сетевого адаптера является последним программным компонентом перед физическим уровнем модели OSI и называется подуровнем управления доступом к среде MAC (Media Access Control). Подуровень MAC ориентирован на выполнение таких функций, как непосредственное управление доступом к передающей среде, проверку пакетов сообщений на наличие ошибок.

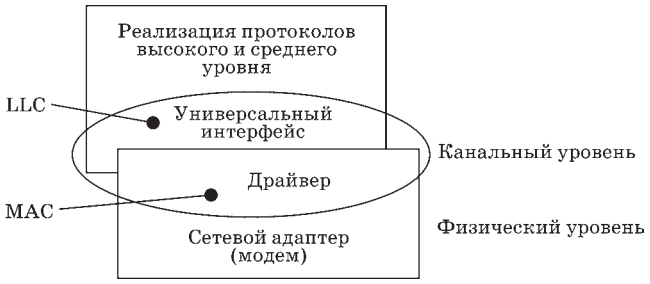


Рис. 2.5
Универсальный интерфейс канального уровня

Подуровень LLC (Logical Linc Control) считается независимым от особенностей физической передающей среды и используемых методов доступа к каналам передачи данных. Стандарты по разработке интерфейсов для связи реализаций протоколов среднего уровня модели OSI с драйверами сетевых аппаратных средств как раз относятся прежде всего к подуровню LLC.

Протоколы среднего уровня распространяются на сетевой, транспортный и сеансовый уровни эталонной модели. По типу межкомпьютерного обмена эти протоколы можно классифицировать следующим образом:

- сеансовые протоколы (протоколы виртуального соединения);
- дейтаграммные протоколы.

Сеансовые протоколы определяют организацию передачи информации между компьютерами по так называемому виртуальному каналу в три этапа:

- установление виртуального канала (установка сеанса);
- реализация непосредственного обмена информацией;
- уничтожение виртуального канала (разъединение).

В сеансовых протоколах порядок следования пакетов при передаче соответствует их исходному порядку в сообщении, а передача осуществляется с подтверждением доставки, а в случае потери отправленных пакетов они передаются повторно.

При использовании дейтаграммных протоколов пакеты сообщений передаются так называемыми дейтаграммами независимо друг от друга, поэтому порядок достав-

ки пакетов каждого сообщения может не соответствовать их исходному порядку в сообщении. При этом пакеты сообщений передаются без подтверждения.

Таким образом, с точки зрения достоверности сеансовые протоколы являются более предпочтительными, зато скорость передачи при использовании дейтаграммных протоколов гораздо выше.

Любой протокол среднего уровня предусматривает следующие этапы реализации межкомпьютерного обмена:

- инициализация связи;
- непосредственный информационный обмен;
- завершение обмена.

Наиболее часто используемыми наборами протоколов среднего уровня являются следующие:

- набор протоколов SPX/IPX, используемый в локальных сетях, функционирующих под управлением сетевой операционной системы NetWare;
- протоколы NetBIOS и NetBEUI, поддерживаемые большинством сетевых операционных систем и используемых только в локальных сетях;
- протоколы TCP/IP, являющиеся стандартом для глобальной сети Интернет, используемые в локальных сетях и поддерживаемые большинством сетевых операционных систем.

Набор протоколов SPX/IPX используется в сетевой операционной системе NetWare фирмы Novell.

Протокол IPX (Internetwork Packet Exchange — межсетевой обмен пакетами) является дейтаграммным протоколом и соответствует сетевому уровню эталонной модели. Применяется для выполнения функций адресации при обмене пакетами сообщений.

Протокол SPX (Sequenced Packet Exchange — последовательный обмен пакетами) является сеансовым протоколом и соответствует транспортному и сеансовому уровням эталонной модели. По степени близости к самому низкому уровню эталонной модели протокол SPX находится над протоколом IPX и использует этот протокол.

Драйвер, реализующий протокол SPX, использует в процессе своей работы драйвер, реализующий протокол

IPX. Протокол IPX является более быстродействующим, чем протокол SPX.

Важным недостатком протоколов SPX и IPX является несовместимость с протоколами TCP/IP, используемыми в глобальной сети Интернет. Для подключения локальной сети NetWare к Интернету используется один из следующих способов:

- непосредственная инсталляция на каждом сетевом компьютере драйверов, реализующих набор протоколов TCP/IP;
- подключение локальной сети к Интернету через шлюз IPX-IP.

Протоколы NetBIOS и NetBEUI разработаны фирмой IBM и предназначены только для локальных компьютерных сетей.

Протокол NetBIOS (Network Basic Input/Output System — базовая система ввода-вывода) соответствует сетевому, транспортному и сеансовому уровням эталонной модели. Реализация данного протокола обеспечивает прикладной интерфейс, используемый для создания сетевых программных приложений.

Протокол NetBEUI (Extended User Interface NetBIOS — расширенный пользовательский интерфейс NetBIOS) является модификацией предыдущего протокола и распространяется только на сетевой и транспортный уровни.

Реализации протоколов NetBIOS и NetBEUI обеспечивают решение следующих задач: поддержка имен, сеансового и дейтаграммного взаимодействия, получение информации о состоянии сети.

Достоинства протоколов NetBIOS и NetBEUI: удобная адресация, высокая производительность, самонастройка и хорошая защита от ошибок, экономное использование оперативной памяти.

Недостатки NetBIOS и NetBEUI связаны с отношением к глобальным сетям: отсутствие поддержки функций маршрутизации и низкая производительность.

Семейство протоколов TCP/IP было разработано для объединения различных компьютерных сетей в одну глобальную сеть, получившую название Интернет.

Семейство протоколов TCP/IP включает протоколы, относящиеся как к средним, так и другим уровням модели OSI:

- прикладной уровень и уровень представления — протокол передачи файлов (FTP), протоколы электронной почты (SMTP, POP3, IMAP4), протоколы удаленного доступа (SLIP, PPP, Telnet), протокол сетевой файловой системы (NPS), протокол управления сетями (SNMP), протокол передачи гипертекста (HTPP) и др.;
- сеансовый и транспортные уровни — протоколы TCP и UDP;
- сетевой уровень — протоколы IP, ICMP, IGMP;
- канальный уровень — протоколы ARP, RARP.

Дейтаграммный протокол IP (Internet Protocol) является основным для сетевого уровня и обеспечивает маршрутизацию передаваемых пакетов сообщений.

Протокол ICMP (Internet Control Message Protocol) отвечает за обмен сообщениями об ошибках и другой важной информацией с программными средствами сетевого уровня на другом компьютере, маршрутизаторе или шлюзе.

Протокол IGMP (Internet Management Protocol) используется для отправки IP-пакетов множеству компьютеров в сети.

Протокол TCP (Transmission Control Protocol) является протоколом сетевого уровня и обеспечивает надежную передачу данных между двумя компьютерами путем организации виртуального канала обмена и использования его для передачи больших массивов данных.

Протокол UDP (User Datagram Protocol) реализует гораздо более простой сервис передачи, обеспечивая надежную доставку данных без установления логического соединения.

Протоколы верхнего уровня соответствуют уровню пользователей и прикладных программ и распространяются на уровень представления и прикладной уровень эталонной модели сетевого взаимодействия. Наиболее распространенными являются следующие высокоуровневые протоколы:

- протоколы перенаправления запросов и обмена сообщениями (SMB, NCP);
- протокол управления сетями (SNMP);
- протокол сетевой файловой системы (NFS);
- протокол вызова удаленных процедур (RPC);
- протоколы, повышающие эффективность использования протоколов TCP/IP среднего уровня (DNS, DHSP);
- протоколы удаленного доступа к компьютерным ресурсам (SLIP, PPP, Telnet);
- протокол передачи файлов (FTP);
- протокол передачи гипертекста (HTTP);
- протоколы электронной почты (SMTP, POP3, IMAP4);
- протокол организации электронных конференций и системы новостей (NNTP).

Протокол SMB (Server Message Blocks — блоки серверных сообщений), разработанный совместно корпорациями Microsoft, Intel и IBM, используется в сетевых операционных системах Windows NT, Lan Manager, LAN Server. Данный протокол определяет серии команд, используемых для передачи информации между сетевыми компьютерами.

Протокол NCP (NetWare Core Protocol — протокол ядра NetWare) разработан фирмой Novell и используется в сетевых ОС NetWare.

Протокол SNMP (Simple Network Management Protocol — простой протокол управления сетью) осуществляет гибкое и полное управление сетью, предполагая выполнение администратором следующих функций: управление конфигурацией, управление доступом к общим сетевым ресурсам, управление производительностью, управление подготовкой к восстановлению, управление восстановлением. При этом любая из функций управления должна обеспечивать решение трех базовых задач:

- получение информации о состоянии управляемого объекта;
- анализ полученной информации и выработка управляющих воздействий;

- передача управляющих воздействий на исполнение.

Протокол NFS (Network File System — сетевая файловая система) предназначен для предоставления универсального интерфейса работы с файлами для различных операционных систем, сетевых архитектур и протоколов среднего уровня.

Протокол RPC (Remote Procedure Call — сервис вызова удаленных процедур) предназначен для организации межпрограммных взаимодействий для сети «клиент — сервер» и обеспечивает связь между процессами-клиентами и процессами-серверами, функционирующими на разных компьютерах сети.

Протокол DNS (Domain Name System — система доменных имен) предназначен для установления соответствия между смысловыми символьными именами и IP-адресами компьютеров.

Протокол DHCP (Dynamic Host Configuration Protocol — протокол динамической конфигурации компьютеров) позволяет автоматически назначать IP-адреса подключаемых к сети компьютеров и изменять их при перемещении из одной подсети в другую.

Протокол SLIP (Serial Line Internet Protocol — протокол Интернета последовательного соединения) обеспечивает работу протоколов TCP/IP при коммутируемом телефонном соединении.

Протокол PPP (Point-to-Point Protocol — протокол «точка — точка») обеспечивает установление соединения и реализацию непосредственного обмена информацией, а также по сравнению со SLIP позволяет решать следующие задачи:

- конфигурация и проверка качества связи;
- подтверждение подлинности (аутентификация) удаленного пользователя;
- динамическое присвоение адресов IP и управление этими адресами;
- обнаружение и коррекция ошибок и др.

Протокол PPTP (Point-to-Point Tunntling Protocol — туннельный протокол «точка — точка») ориентирован на поддержку мультипротокольных виртуальных частных

сетей (Virtual Private Networks — VPN), предоставляя возможность удаленным пользователям иметь безопасный доступ к корпоративным сетям по Интернету.

Протокол Telnet является общепризнанным стандартом удаленного дистанционного управления в Интернете, позволяющим в режиме командной строки запускать и выполнять программы на компьютере, с которым установлено удаленное соединение.

Основное требование к компьютерной сети — это выполнение сетью того набора услуг, для которого она предназначена. К таким услугам могут относиться:

- доступ к файловым архивам;
- доступ к страницам веб-сайтов;
- обмен с использованием электронной почты;
- интерактивный обмен с помощью IP-телефонии;
- потоковое видео и т. д.

Все остальные требования (как то: производительность, надежность, совместимость, управляемость, защищенность, расширяемость и масштабируемость) связаны с качеством выполнения основной задачи.

Хотя важны все перечисленные требования, в понятие «качество обслуживания» (Quality of Service — QoS) для компьютерной сети часто включают только две важнейшие характеристики — производительность и надежность.

Рассмотрим эти параметры.

Производительность

К основным характеристикам производительности относятся:

- время реакции;
- скорость передачи данных;
- задержка передачи и ее вариация.

Время реакции. Определяют как интервал времени между возникновением запроса пользователя к какой-либо сетевой службе и получением ответа на этот запрос.

Обычно это время складывается из нескольких составляющих (рис. 2.6):

t_1 — время подготовки запроса на пользовательском компьютере;

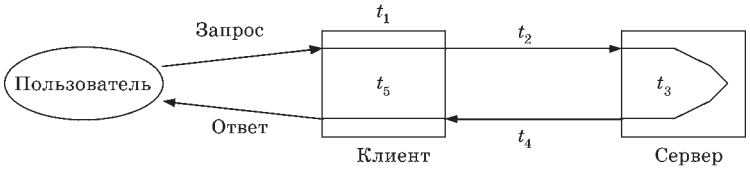


Рис. 2.6
Обмен «пользователь — клиент — сервер»

t_2 — время передачи между клиентом и сервером;

t_3 — время обработки запроса на сервере;

t_4 — время доставки ответа с сервера на клиентский компьютер;

t_5 — время обработки ответа на этом компьютере.

С точки зрения работы сети здесь важнейшими составляющими являются t_2 и t_4, т. е. задержки на передачу по сети.

Скорость передачи данных. Это объем данных, передаваемый в единицу времени. Используется также понятие пропускной способности — это скорость передачи пакетов между узлами сети через различные коммуникационные устройства. Скорость передачи измеряется:

- в битах в секунду;
- в пакетах в секунду.

Различают три скорости.

Средняя скорость. Берется достаточно длительный промежуток времени (час, сутки) и общий объем переданных данных делится на время.

Мгновенная скорость. Для усреднения выбирается очень маленький промежуток времени, например 10 мс или 1 с.

Максимальная скорость. Это максимальная мгновенная скорость, зафиксированная за время наблюдения.

При проектировании сети чаще всего пользуются параметрами средней и максимальной скорости.

Иногда оперируют и общей пропускной способностью, рассматриваемой как максимальное количество информации, передаваемой между всеми узлами сети в единицу времени.

Задержка передачи. Это время между моментом поступления данных на вход какого-то сетевого устройства или части сети и моментом появления этих данных на выходе, т. е.

$$t_{\text{задержки}} = t_{\text{вых}} - t_{\text{вх}}.$$

Отличием этого параметра от времени реакции является то, что в $t_{\text{задержки}}$ никогда не включается время на обработку в конечных узлах сети.

Обычно рассматривают:

- максимальную задержку;
- вариацию задержки.

Надежность и безопасность

Для обычных технических средств используют такие показатели надежности, как:

- среднее время наработки на отказ;
- вероятность отказа;
- интенсивность отказов.

Однако эти характеристики пригодны для оценки только простых устройств, которые могут находиться в двух состояниях — работоспособном и отказа.

Сложные системы из многих элементов могут иметь и промежуточные состояния. Поэтому для оценки надежности таких сложных систем, как сети, применяют другой набор характеристик.

Готовность (или *коэффициент готовности*, *Availability*). Это доля времени, в течение которого система может быть использована. Для увеличения коэффициента готовности в состав системы включаются резервные элементы.

Сохранность данных (и *их защита от искажений*).

Согласованность данных (*их непротиворечивость*). Требуется, например, когда несколько копий данных хранятся на разных файловых серверах.

Вероятность доставки пакета узлу назначения без искажений. Может также применяться параметр *вероятность потери пакета* (например, из-за переполнения буфера маршрутизатора, отсутствия работоспособного пути

в сети, поражения пакета ошибками). Этот параметр может представляться в виде:

- вероятности искажения отдельного бита передаваемых данных;
- отношения числа потерянных пакетов к общему числу передаваемых пакетов.

Безопасность (*Security*). Это способность системы защитить данные от несанкционированного доступа. Сюда относятся защита каналов, защита компьютеров, защита от взлома паролей и т. д.

Отказоустойчивость (*Fault tolerance*). В сетях под этим понимают способность системы скрывать от пользователя факт отказа отдельных элементов. Например, если копии данных хранятся на нескольких файловых серверах, пользователь может и не заметить факт отказа одного из них. В этом случае говорят о деградации системы, так как при отказе одного сервера увеличивается время доступа к базе данных из-за уменьшения степени распараллеливания запросов.

Специфичными для сетей являются параметры расширяемости и масштабируемости.

Расширяемость (*Extensibility*). Это возможность легко добавлять в сеть новые элементы (пользователей, компьютеры, приложения, службы), наращивать длину сегментов и заменять аппаратуру более мощной.

Масштабируемость (*Scalability*). Этот параметр означает, что в сети возможно наращивание количества узлов и протяженности связей в очень широких пределах. При этом производительность сети не ухудшается.

Часто термины «расширяемость» и «масштабируемость» используются как синонимы. Однако если взять, к примеру, сеть Ethernet, то можно говорить о хорошей расширяемости (количество компьютеров на сегменте можно увеличить до 100), но при этом резко снижается производительность сети, т. е. это указывает на плохую масштабируемость.

Прозрачность (*Transparency*). Прозрачность сети достигается в том случае, когда для пользователя сеть представляется не как множество компьютеров, связанных

сложной системой каналов, а как единая вычислительная машина с системой разделения времени.

Символом прозрачности считают принцип: «Сеть — это компьютер».

Прозрачность может достигаться на двух уровнях — *пользователя* и *программиста*.

На уровне пользователя — для работы в сети используются те же команды и привычные процедуры, что и для работы с локальными ресурсами.

На уровне программиста — приложению для доступа к удаленным ресурсам требуются те же вызовы, что и для локальных ресурсов.

Сеть должна скрывать различия операционных систем и компьютеров. Можно одинаково обращаться к ресурсам на компьютере с ОС Macintosh, Windows или Unix. От пользователя не требуется знание места расположения ресурса. Ресурсы должны свободно перемещаться с одного компьютера на другой без изменения их имен.

Поддержка разных видов трафика. Наряду с традиционным трафиком передачи данных все увеличивается доля мультимедийного трафика — передаваемых в цифровой форме речи и изображения.

Особенность мультимедийного трафика — это жесткие требования к синхронизации передаваемых данных. При запаздывании сообщений будут наблюдаться искажения.

Необходимость передачи мультимедийного трафика требует внесения изменений, как в протоколы, так и в оборудование.

В сети в общем случае должны сосуществовать два вида трафика:

- традиционный компьютерный (пульсирующий);
- мультимедийный (синхронный).

Это является сложной задачей и ближе всего к ее решению подошли сети ATM.

Управляемость. Это возможность централизованно контролировать состояние основных элементов сети, выявлять и устранять неисправности, выполнять анализ производительности и планировать развитие сети.

В этой области еще очень много нерешенных проблем. В основном существующие системы не управляют сетью, а лишь осуществляют наблюдение за ее работой.

Совместимость. Совместимость или *интегрируемость* означает, что сеть способна включать в себя разнообразное ПО и аппаратное обеспечение. То есть сеть может быть неоднородной или гетерогенной. Еще такие сети называют интегрированными.

В них могут сосуществовать различные ОС, сетки протоколов, аппаратные средства и приложения от разных производителей.

Основной путь обеспечения совместимости — это использование открытых стандартов и спецификаций.

Традиционные сети обеспечивают сервис, который получил название Best Effort — с максимальными усилиями. Это означает, что сеть не дает никаких гарантий на обслуживание.

Примеры: сети Ethernet, Token Ring, IP, X25.

При обработке очередей используется обычно алгоритм FIFO, а при переполнении буфера — пакеты отбрасываются.

В настоящее время самый распространенный путь обеспечения QoS — это постоянное наращивание пропускной способности сети, т. е. стремление постоянно иметь избыточную пропускную способность.

Встроенные механизмы QoS пока применяются только в таких сетях, как ATM и Frame Relay.

Типы QoS

Сервис Best Effort — с максимальными усилиями. Это фактически отсутствие QoS. Обслуживание производится без каких-либо гарантий. Пример: IP-сети и Ethernet с принципом FIFO.

Сервис с предпочтением (называют также «мягким» сервисом QoS). Здесь некоторые виды трафика обслуживаются лучше остальных. Но характеристики обслуживания точно неизвестны — они зависят от характеристик трафика. Например, при высокой интенсивности высокоприоритетного трафика может совсем прекратиться обслуживание трафика с низким приоритетом.

Гарантированный сервис (его называют также «жестким» или «истинным» сервисом QoS). Различным типам трафика даются статистические гарантии. Обычно этот тип QoS основан на предварительном резервировании сетевых ресурсов для каждого из потоков, получивших гарантии обслуживания. Однако и эти гарантии носят статистический характер. Например, с вероятностью 0,999 задержка пакета не должна превышать 100 мс.

При этом производится контроль интенсивности входных потоков — чтобы это значение не превышало заранее оговоренную величину. Такой тип QoS применяется обычно для обслуживания тех приложений, для которых нужны гарантии пропускной способности и/или задержек. Например, это может быть трафик видеоконференции или трафик, поступающий от измерительных систем реального времени.

Рассмотрим новые базовые сети и их роль в обеспечении качества обслуживания.

Эти сети являются основой вторичных сетей — компьютерных и телефонных. Как было показано выше, обеспечение QoS зависит от имеющегося резерва пропускных способностей, т. е. от производительности первичной сети. Без развития первичных сетей невозможен прогресс сетевых технологий.

Такие сети называют также *опорными* и *первичными*.

Современные сети основаны на коммутации каналов. Для создания абонентского канала коммутаторы первичных сетей поддерживают один из методов мультиплексирования и коммутации.

В настоящее время для мультиплексирования абонентских каналов используются:

- техника частотного мультиплексирования — Frequency Division Multiplexing, FDM;
- мультиплексирование с разделением времени — Time Division Multiplexing, TDM;
- мультиплексирование по длине волны — Wave Division Multiplexing, WDM.

Частотное мультиплексирование. Этот метод применяется в основном в телефонных сетях, где речевой

телефонный канал имеет спектр 300–3400 Гц (т. е. на его передачу необходима пропускная способность 3100 Гц). Кабельные же системы между телефонными коммутаторами имеют пропускную способность в сотни мегагерц. Для передачи производится модуляция высокочастотного сигнала низкочастотным. Таким образом, спектр модулированного сигнала переносится в другой частотный диапазон. Высокочастотный сигнал делится на полосы по 4000 Гц (3100 + 900 — страховой промежуток). В сетях на основе FDM-коммутации принято несколько уровней уплотненных каналов. Это базовая группа (12 абонентских каналов), супергруппа (60 абонентских каналов) и главная группа (600 абонентских каналов), которая имеет полосу пропускания 2520 кГц (564–3084 кГц).

Мультиплексирование по длине волны. Первичные сети с мультиплексированием по длине волны (WDM и DWDM) используют тот же принцип частотного разделения, но информационным сигналом в них является не электрический ток, а свет. Используется инфракрасный диапазон с длинами волн от 850 до 1565 нм, что соответствует частотам 196–350 ТГц. В магистральном канале обычно мультиплексируется достаточно много спектральных каналов: 16, 32, 40, 80 или 160. Если используются 16 и более каналов, такая техника часто называется плотной, т. е. Dense WDM или DWDM. Внутри спектрального канала данные могут кодироваться как дискретным, так и аналоговым способом.

Коммутация каналов на основе разделения времени. Как уже упоминалось, эта техника носит название мультиплексирование с разделением времени (Time Division Multiplexing, TDM). Основой этой технологии являются каналы T1/E1, которые были предложены для передачи вызовов между телефонными станциями (АТС). Это дуплексные цифровые каналы.

Для передачи используются две пары витых проводников (по паре в каждую сторону). В 1990-е гг. эти каналы стали очень популярны в качестве средства подключения абонентов (небольших фирм, корпоративных сетей) к сети Интернет.

На базе внедренных в телефонии каналов T1 к настоящему времени сформировались два поколения таких цифровых базовых сетей:

1) технология *плезиохронной цифровой передачи* (*Plesiochronous Digital Hierarchy*, *PDH*). «Плезо» означает «почти», т. е. это почти синхронная передача;

2) *синхронная цифровая иерархия* (*Sinchronous Digital Hierarchy*, *SDH*). В США технология SDH называется SONET.

Недостатки технологии PDH:

1) сложность мультиплексирования и демультиплексирования данных. Например, для извлечения одного канала из потока T3 надо демультиплексировать канал до T2, затем — до T1, а уже затем демультиплексировать канал T1;

2) отсутствие встроенных средств контроля и управления сетью. Нет и процедур поддержки отказоустойчивости;

3) слишком низкие (по понятиям современных сетей) скорости передачи. Иерархия скоростей E1 заканчивается на 139 Мбит/с, а современные оптоволоконные каналы позволяют передавать со скоростью в десятки гигабит в секунду.

Эти недостатки были устранены в сетях SDH, которые и стали в настоящее время одним из тех типов сетей, который позволяет удовлетворить требования к качеству обслуживания.

Сети SDH. Технология синхронной цифровой иерархии (*Sinchronous Digital Hierarchy*, *SDH*) разработана для создания надежных транспортных каналов, позволяющих гибко формировать цифровые каналы в широком диапазоне скоростей — от единиц мегабит в секунду до десятков гигабит в секунду.

Основная область применения — первичные сети операторов связи.

Эти сети относятся к классу полупостоянных сетей с коммутацией каналов — формирование канала происходит по инициативе оператора связи. Поэтому здесь чаще всего вместо термина «коммутация» используют термин

«кросс-коннект» (cross-connect). Используется мультиплексирование с разделением времени TDM. Информация адресуется путем относительного временного положения внутри составного кадра. Обычно эти сети используются для объединения большого числа более низкоскоростных каналов PDH. На рисунке 2.7 приведен пример структуры сети SDH.

Достоинства сетей SDH:

1) гибкая иерархическая система мультиплексирования цифровых потоков с различными скоростями. Возможность ввода и вывода пользовательской информации без демультиплексирования потока в целом;

2) отказоустойчивость сети. Использование резервных маршрутов и резервного оборудования. Переход на резервный путь обычно требует не более 50 мс;

3) мониторинг и управление сетью на основе той информации, которая встроена в заголовки кадров;

4) высокое качество транспортного обслуживания для трафика любого вида — голосового, видео и компьютерного. Техника мультиплексирования TDM, лежащая в основе SDH, обеспечивает трафику каждого абонента гарантированную пропускную способность, а также низкий и фиксированный уровень задержек.

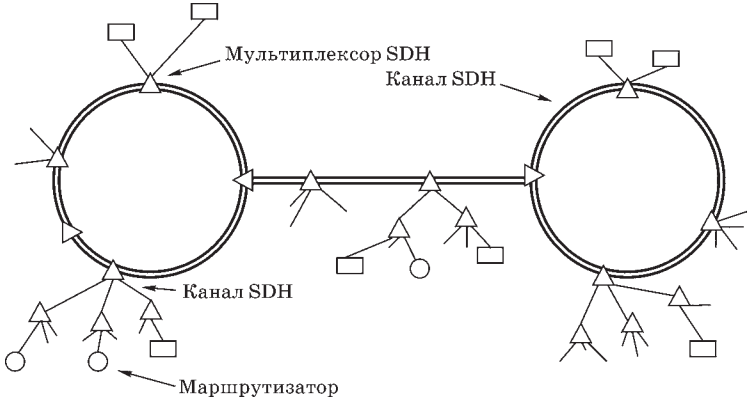


Рис. 2.7
Пример структуры сети SDH

5) сети SDH составляют сегодня фундамент практически всех крупных телекоммуникационных сетей — региональных, национальных и международных;

6) эти сети легко интегрируются с сетями DWDM, обеспечивая передачу информации по оптическим магистралям со скоростями сотни гигабит в секунду за счет мультиплексирования с разделением по длине волны (рис. 2.8).

В DWDM сетях сети SDH играют роль сетей доступа, т. е. ту же роль, что играют по отношению к ним сети PDH.

Типовые топологии. Чаще всего используются кольца, линейные цепи и ячеистая топология, близкая к полносвязной.

Сети DWDM. Вторым типом современных базовых сетей, которые позволяют обеспечить поддержку служб, имеющих требуемое качество обслуживания, являются сети DWDM.

Технология плотного волнового (спектрального) мультиплексирования (Dense Wave Division Multiplexing, DWDM) предназначена для создания оптических магистралей нового поколения, работающих на мультимегабитных и терабитных скоростях.

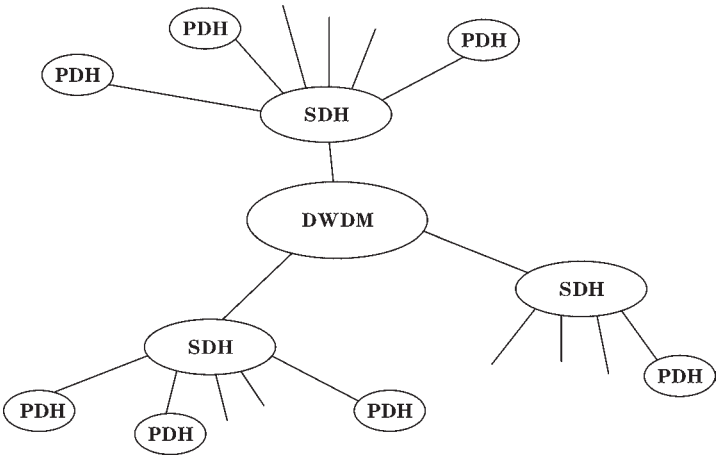


Рис. 2.8
Иерархия базовых сетей и сетей доступа

Информация в оптическом волокне передается одновременно большим количеством световых волн.

Сети DWDM работают по принципу коммутации каналов — при этом каждая световая волна представляет собой отдельный спектральный канал.

Каждая волна несет собственную информацию, при этом оборудование DWDM не занимается непосредственно проблемами передачи данных на каждой волне, т. е. способом кодирования информации и протоколом ее передачи.

Устройства DWDM занимаются только объединением различных волн в одном световом пучке, а также выделением из общего сигнала информации каждого спектрального канала.

Оборудование DWDM позволяет передавать по одному оптическому каналу 32 и более волн различной длины в окне прозрачности 1550 нм. При этом каждая волна может переносить информацию со скоростью до 10 Гбит/с (при применении протоколов STM-64 или 10GE). Ведутся работы по повышению этой скорости до 40–80 Гбит/с.

Мультиплексирование DWDM называется «плотным», так как в нем используется существенно меньшее (чем у предшествующей технологии WDM) расстояние между длинами волн.

В рекомендации G.692 определен частотный план с разнесением частот на 100 ГГц ($\Delta\lambda = 0{,}8$ нм). Для передачи используется 41 волна (от 1528 нм (191 ТГц) до 1560 нм (192 ТГц)).

Определен также частотный план с разнесением на 50 ГГц ($\Delta\lambda = 0{,}4$ нм), что позволяет передавать в этом диапазоне 81 волну.

Имеются экспериментальные образцы с разнесением на 25 ГГц. Такая технология называется *High Dence WDM* (HDWDM).

Как видно на рисунке 2.9, необходимо обеспечить высокую точность частоты, чтобы не допустить перекрытия спектра каналов.

Преимущества технологии DWDM:

1) повышение коэффициента использования частотного диапазона оптоволокна;

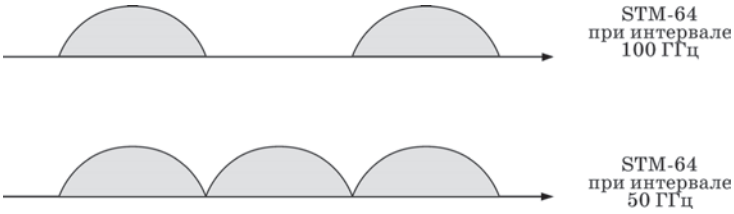


Рис. 2.9
Интервалы между каналами

2) отличная масштабируемость — достаточно просто добавить новые спектральные каналы;

3) экономическая эффективность — не требуется электрическая регенерация на длинных маршрутах;

4) независимость от протокола передачи — через магистраль DWDM можно передавать трафик сетей любого типа;

5) независимость спектральных каналов друг от друга;

6) совместимость с технологией SDH. Мультиплексоры DWDM оснащаются интерфейсами STM-N, способными принимать и передавать данные мультиплексоров SDH;

7) совместимость с технологиями Ethernet — Gigabit Ethernet и 10GE.

8) стандартизация на уровне Международного союза по телекоммуникациям ITU-T.

2.3. ОБРАБОТКА ИНФОРМАЦИИ

Обработка информации состоит в получении одних информационных объектов из других информационных объектов путем выполнения некоторых алгоритмов и является одной из основных операций, выполняемых над информацией и главным средством увеличения ее объема и разнообразия.

На самом верхнем уровне можно выделить числовую и нечисловую обработку. В указанные виды обработки вкладывается различная трактовка содержания понятия «данные». При числовой обработке используются такие объекты, как переменные, векторы, матрицы, многомер-

ные массивы, константы и т. д. При нечисловой обработке объектами могут быть файлы, записи, поля, иерархии, сети, отношения и т. д. Другое отличие заключается в том, что при числовой обработке содержание данных не имеет большого значения, в то время как при нечисловой обработке нас интересуют непосредственные сведения об объектах, а не их совокупность в целом.

Режимы обработки данных

При проектировании технологических процессов ориентируются на режимы их реализации. Режим реализации технологии зависит от объемно-временных особенностей решаемых задач: периодичности и срочности, требований к быстроте обработки сообщений, а также от режимных возможностей технических средств, и в первую очередь ЭВМ. Существуют: пакетный режим; режим реального масштаба времени; режим разделения времени; регламентный режим; запросный; диалоговый; телеобработки; интерактивный; однопрограммный; многопрограммный (мультиобработка).

Пакетный режим. При использовании этого режима пользователь не имеет непосредственного общения с ЭВМ. Сбор и регистрация информации, ввод и обработка не совпадают по времени. Вначале пользователь собирает информацию, формируя ее в пакеты в соответствии с видом задач или каким-то другим признаком. (Как правило, это задачи неоперативного характера, с долговременным сроком действия результатов решения.) После завершения приема информации производится ее ввод и обработка, т. е. происходит задержка обработки. Этот режим используется, как правило, при централизованном способе обработки информации.

Диалоговый режим (запросный) — режим, при котором существует возможность пользователя непосредственно взаимодействовать с вычислительной системой в процессе работы пользователя. Программы обработки данных находятся в памяти ЭВМ постоянно, если ЭВМ доступна в любое время или в течение определенного промежутка времени, когда ЭВМ доступна пользователю.

Взаимодействие пользователя с вычислительной системой в виде диалога может быть многоаспектным и определяться различными факторами: языком общения, активной или пассивной ролью пользователя; кто является инициатором диалога — пользователь или ЭВМ; временем ответа; структурой диалога и т. д. Если инициатором диалога является пользователь, то он должен обладать знаниями по работе с процедурами, форматами данных и т. п. Если инициатор ЭВМ, то машина сама сообщает на каждом шаге, что нужно делать с разнообразными возможностями выбора. Этот метод работы называется выбором меню. Он обеспечивает поддержку действий пользователя и предписывает их последовательность. При этом от пользователя требуется меньшая подготовленность.

Диалоговый режим требует определенного уровня технической оснащенности пользователя, т. е. наличия терминала или ПЭВМ, связанных с центральной вычислительной системой каналами связи. Этот режим используется для доступа к информации, вычислительным или программным ресурсам. Возможность работы в диалоговом режиме может быть ограничена во времени начала и конца работы, а может быть и неограниченной.

Иногда различают диалоговый и запросный режимы, тогда под запросным понимается одноразовое обращение к системе, после которого она выдает ответ и отключается, а под диалоговым — режим, при которым система после запроса выдает ответ и ждет дальнейших действий пользователя.

Режим реального масштаба времени. Означает способность вычислительной системы взаимодействовать с контролируемыми или управляемыми процессами в темпе протекания этих процессов. Время реакции ЭВМ должно удовлетворять темпу контролируемого процесса или требованиям пользователей и иметь минимальную задержку. Как правило, этот режим используется при децентрализованной и распределенной обработке данных.

Режим телеобработки дает возможность удаленному пользователю взаимодействовать с вычислительной системой.

Интерактивный режим предполагает возможность двустороннего взаимодействия пользователя с системой, т. е. у пользователя есть возможность воздействия на процесс обработки данных.

Режим разделения времени предполагает способность системы выделять свои ресурсы группе пользователей поочередно. Вычислительная система настолько быстро обслуживает каждого пользователя, что создается впечатление одновременной работы нескольких пользователей. Такая возможность достигается за счет соответствующего программного обеспечения.

Однопрограммный и многопрограммный режимы характеризуют возможность системы работать одновременно по одной или нескольким программам.

Регламентный режим характеризуется определенностью во времени отдельных задач пользователя. Например, получение результатных сводок по окончании месяца, расчет ведомостей начисления зарплаты к определенным датам и т. д. Сроки решения устанавливаются заранее по регламенту в противоположность к произвольным запросам.

Способы обработки данных

Различаются следующие способы обработки данных: централизованный, децентрализованный, распределенный и интегрированный.

Централизованная обработка. При этом способе пользователь доставляет на ВЦ исходную информацию и результаты обработки получает в виде результативных документов. Особенностями такого способа обработки являются сложность и трудоемкость налаживания быстрой, бесперебойной связи, большая загруженность ВЦ информацией (так как велик ее объем), регламентацией сроков выполнения операций, организация безопасности системы от возможного несанкционированного доступа.

Децентрализованная обработка. Этот способ связан с появлением ПЭВМ, дающих возможность автоматизировать конкретное рабочее место.

Распределенный способ обработки данных основан на распределении функций обработки между различными

ЭВМ, включенными в сеть. Этот способ может быть реализован двумя путями: первый предполагает установку ЭВМ в каждом узле сети (или на каждом уровне системы), при этом обработка данных осуществляется одной или несколькими ЭВМ в зависимости от реальных возможностей системы и ее потребностей на текущий момент времени. Второй путь — размещение большого числа различных процессоров внутри одной системы. Такой путь применяется в системах обработки банковской и финансовой информации, там, где необходима сеть обработки данных (филиалы, отделения и т. д.). Преимущества распределенного способа: возможность обрабатывать в заданные сроки любой объем данных; высокая степень надежности, так как при отказе одного технического средства есть возможность моментальной замены его на другой; сокращение времени и затрат на передачу данных; повышение гибкости систем, упрощение разработки и эксплуатации программного обеспечения и т. д. Распределенный способ основывается на комплексе специализированных процессоров, т. е. каждая ЭВМ предназначена для решения определенных задач или задач своего уровня.

Интегрированный способ обработки информации. Он предусматривает создание информационной модели управляемого объекта, т. е. создание распределенной базы данных. Такой способ обеспечивает максимальное удобство для пользователя. С одной стороны, базы данных предусматривают коллективное пользование и централизованное управление. С другой стороны, объем информации, разнообразие решаемых задач требуют распределения базы данных. Технология интегрированной обработки информации позволяет улучшить качество, достоверность и скорость обработки, так как обработка производится на основе единого информационного массива, однократно введенного в ЭВМ. Особенностью этого способа является отделение технологически и по времени процедуры обработки от процедур сбора, подготовки и ввода данных.

С точки зрения реализации на основе современных достижений вычислительной техники выделяют следующие виды обработки информации:

- последовательная обработка, применяемая в традиционной фоннеймановской архитектуре ЭВМ, располагающей одним процессором;
- параллельная обработка, характеризующаяся наличием нескольких процессоров в ЭВМ;
- конвейерная обработка, связанная с использованием в архитектуре ЭВМ одних и тех же ресурсов для решения разных задач, причем если эти задачи тождественны, то это последовательный конвейер, если задачи одинаковые — векторный конвейер.

Принято относить существующие архитектуры ЭВМ с точки зрения обработки информации к одному из следующих классов [28].

Архитектуры с одиночным потоком команд и одиночным потоком данных (SISD). К этому классу относятся традиционные фоннеймановские однопроцессорные системы, где имеется центральный процессор, работающий с парами «атрибут — значение».

Архитектуры с одиночным потоком команд и множественным потоком данных (SIMD). Особенностью данного класса является наличие одного (центрального) контроллера, управляющего рядом одинаковых процессоров. В зависимости от возможностей контроллера и процессорных элементов, числа процессоров, организации режима поиска и характеристик маршрутных и выравнивающих сетей выделяют:

- матричные процессоры, используемые для решения векторных и матричных задач;
- ассоциативные процессоры, применяемые для решения нечисловых задач и использующие память, в которой можно обращаться непосредственно к информации, хранящейся в ней;
- процессорные ансамбли, применяемые для числовой и нечисловой обработки;
- конвейерные и векторные процессоры.

Архитектуры с множественным потоком команд и одиночным потоком данных (MISD). К этому классу могут быть отнесены конвейерные процессоры.

Архитектуры с множественным потоком команд и множественным потоком данных (MIMD). К этому клас-

су могут быть отнесены следующие конфигурации: мультипроцессорные системы, системы с мультобработкой, вычислительные системы из многих машин, вычислительные сети.

Основные процедуры обработки данных представлены на рисунке 2.10 [8].

Создание данных как процесс обработки предусматривает их образование в результате выполнения некоторого алгоритма и дальнейшее использование для преобразований на более высоком уровне.

Модификация данных связана с отображением изменений в реальной предметной области, осуществляемых путем включения новых данных и удалением ненужных.

Контроль, безопасность и целостность направлены на адекватное отображение реального состояния предметной области в информационной модели и обеспечивают защиту информации от несанкционированного доступа (безопасность) и от сбоев и повреждений технических и программных средств.

Поиск информации, хранимой в памяти компьютера, осуществляется и как самостоятельное действие при вы-

Рис. 2.10
Основные процедуры обработки данных

полнении ответов на различные запросы, и как вспомогательная операция при обработке информации.

Поддержка принятия решения является наиболее важным действием, выполняемым при обработке информации. Широкая альтернатива принимаемых решений приводит к необходимости использования разнообразных математических моделей [10], [11].

Создание документов, сводок, отчетов заключается в преобразовании информации в формы, пригодные для чтения и человеком, и компьютером. С этим действием связаны и такие операции, как обработка, считывание, сканирование и сортировка документов.

Преобразование информации осуществляет ее перевод из одной формы представления или существования в другую и определяется потребностями, возникающими в процессе реализации информационных технологий.

Реализация всех действий, выполняемых в процессе обработки информации, осуществляется с помощью разнообразных программных средств.

Наиболее распространенной областью применения технологической операции обработки информации является принятие решений.

В зависимости от степени информированности о состоянии управляемого процесса, полноты и точности моделей объекта и системы управления, взаимодействия с окружающей средой *процесс принятия решения протекает в различных условиях*.

Принятие решений в условиях определенности. В этой задаче модели объекта и системы управления считаются заданными, а влияние внешней среды — несущественным. Поэтому между выбранной стратегией использования ресурсов и конечным результатом существует однозначная связь, откуда следует, что в условиях определенности достаточно использовать решающее правило для оценки полезности вариантов решений, принимая в качестве оптимального то, которое приводит к наибольшему эффекту. Если таких стратегий несколько, то все они считаются эквивалентными. Для поиска решений в условиях определенности используют методы математического программирования.

1. Принятие решений в условиях риска. В отличие от предыдущего случая, для принятия решений в условиях риска необходимо учитывать влияние внешней среды, которое не поддается точному прогнозу, а известно только вероятностное распределение ее состояний. В этих условиях использование одной и той же стратегии может привести к различным исходам, вероятности появления которых считаются заданными или могут быть определены. Оценку и выбор стратегий проводят с помощью решающего правила, учитывающего вероятность достижения конечного результата.

Принятие решений в условиях неопределенности. Как и в предыдущей задаче, между выбором стратегии и конечным результатом отсутствует однозначная связь. Кроме того, неизвестны также значения вероятностей появления конечных результатов, которые либо не могут быть определены, либо не имеют в контексте содержательного смысла. Каждой паре «стратегия — конечный результат» ставится в соответствие некоторая внешняя оценка в виде выигрыша. Наиболее распространенным является использование критерия получения максимального гарантированного выигрыша.

2. Принятие решений в условиях многокритериальности. В любой из перечисленных выше задач многокритериальность возникает в случае наличия нескольких самостоятельных, несводимых одна к другой целей. Наличие большого количества решений усложняет оценку и выбор оптимальной стратегии. Одним из возможных путей решения является использование методов моделирования.

Решение задач с помощью искусственного интеллекта (см. п. 3.6) заключается в сокращении перебора вариантов при поиске решения, при этом программы реализуют те же принципы, которыми пользуется в процессе мышления человек.

Процесс выработки решения на основе первичных данных, схема которого представлена на рисунке 2.11, можно разбить на два этапа: выработка допустимых вариантов решений путем математической формализации

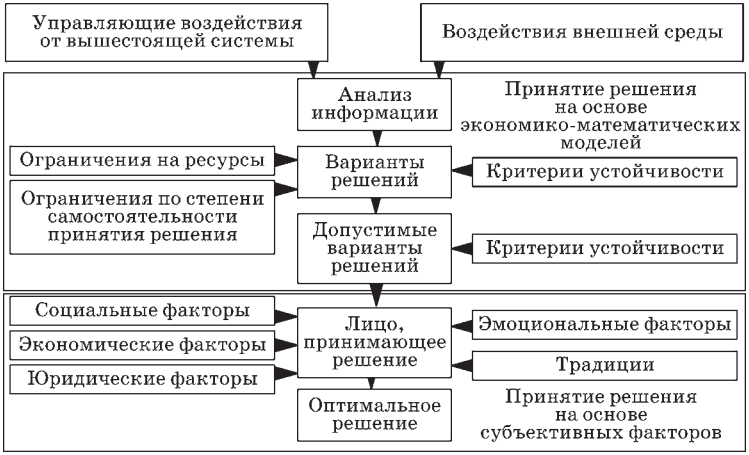


Рис. 2.11
Процесс выработки решения на основе первичных данных

с использованием разнообразных моделей и выбор оптимального решения на основе субъективных факторов.

Информационные потребности лиц, принимающих решение, во многих случаях ориентированы на интегральные технико-экономические показатели, которые могут быть получены в результате обработки первичных данных, отражающих текущую деятельность предприятия. Анализируя функциональные взаимосвязи между итоговыми и первичными данными, можно построить так называемую информационную схему, которая отражает процессы агрегирования информации. Первичные данные, как правило, чрезвычайно разнообразны, интенсивность их поступления высока, а общий объем на интересующем интервале велик. С другой стороны, состав интегральных показателей относительно мал, а требуемый период их актуализации может быть значительно ниже периода изменения первичных данных — аргументов.

Для поддержки принятия решений обязательным является наличие следующих компонент:

- обобщающий анализ;
- прогнозирование;
- ситуационное моделирование.

Система поддержки принятия решений (СППР) может быть представлена как автоматизированная интерактивная человеко-машинная система поддержки принятия решений на основе использования данных и моделей. СППР предназначены для поддержки управленческих решений и интуитивного подхода к решению управленческих задач. Для этого они обеспечивают пользователей необходимой информацией, генерируют, оценивают и предлагают несколько вариантов. Системы поддержки принятия решений (СППР) входят в состав практически любой современной информационной системы. Понятия информационной системы (ИС) и понятия системы поддержки принятия решений являются взаимодополняющими. В соответствии с характером обработки информации в ИС на различных уровнях управления заданной системой (оперативном, тактическом и стратегическом) выделяются следующие типы информационных подсистем:

- системы обработки данных (EDP — Electronic Data Processing);
- информационная система управления (MIS — Management Information System);
- система поддержки принятия решений (DSS — Decision Support System).

Для задач СППР свойственны недостаточность имеющейся информации, ее противоречивость и нечеткость, преобладание качественных оценок целей и ограничений, слабая формализованность алгоритмов решения. В качестве инструментов обобщения чаще всего используются средства составления аналитических отчетов произвольной формы, методы статистического анализа, экспертных оценок и систем, математического и имитационного моделирования.

Таким образом, СППР — это автоматизированная система, используемая для различных видов деятельности при принятии решений в ситуациях, где невозможно или нежелательно иметь автоматическую систему, полностью выполняющую весь процесс решения, вследствие слабой структурированности или неструктурированности решаемых проблем.

Такая «система поддержки принятия решения» может состоять из нескольких подсистем, реализующих следующие семь основных функций СППР:

1) оценка обстановки (ситуации), выбор критериев и оценка их относительной важности;

2) генерация возможных решений (сценариев действий);

3) оценка решений (сценариев действий) и выбор лучшего;

4) обмен информацией об обстановке принимаемых решений и согласование групповых решений (в тех случаях, когда это возможно);

5) моделирование принимаемых решений (в тех случаях, когда это возможно);

6) динамический компьютерный анализ возможных последствий принимаемых решений;

7) сбор данных о результатах реализации принятых решений и оценка результатов.

СППР зачастую используются как интегрированные подсистемы в составе сложных комплексов управления и контроля. Для подобных интегрированных СППР можно выделить следующие основные функции:

- обеспечение ЛПР информацией для процесса принятия решения, включая ее предварительную обработку;
- организационно-методическое обеспечение процесса принятия решений;
- моделирование последствий принятия решений;
- экспертные функции: выдача рекомендаций и обоснований;
- обеспечение согласованности решений, принимаемых в группах.

Модель процесса принятия решений человеком включает три основные ступени: интеллектуальную, конструирование и выбор. Термин «поддержка» подразумевает различные шаги и задачи на каждом этапе процесса принятия решений.

СППР можно, в зависимости от данных, с которыми они работают, разделить на:

- оперативные, предназначенные для немедленного реагирования на текущую ситуацию;
- стратегические, основанные на анализе большого количества информации из разных источников с привлечением сведений, содержащихся в системах, аккумулирующих опыт решения проблем.

Оперативные СППР получили название информационных систем руководства (Executive Information Systems, ИСР). По сути, они представляют собой конечные наборы отчетов, построенные на основании данных из информационной системы предприятия или OLTP-системы. Для ИСР характерны следующие основные черты:

- отчеты, как правило, базируются на стандартных для организации запросах; число последних относительно невелико;
- ИСР представляет отчеты в максимально удобном виде, включающем наряду с таблицами деловую графику, мультимедийные возможности и т. п.;
- как правило, ИСР ориентированы на конкретный вертикальный рынок, например финансы, маркетинг, управление ресурсами.

Стратегические СППР предполагают достаточно глубокую проработку данных, специально преобразованных так, чтобы их было удобно использовать в ходе процесса принятия решений. Неотъемлемым компонентом СППР этого уровня являются правила принятия решений, которые на основе агрегированных данных подсказывают менеджерскому составу выводы и придают системе черты искусственного интеллекта. Такого рода системы создаются только в том случае, если структура бизнеса уже достаточно определена и имеются основания для обобщения и анализа не только данных, но и процессов их обработки.

Бóльшая часть используемых сегодня СППР разработана для генерации и оценки альтернатив посредством анализа «что — если» и «поиска цели» на этапах конструирования и выбора. *Финансовые* модели служат для поддержки и планирования, рассчитывая последствия запланированных действий на основе оценки прибылей.

Репрезентативные модели оценивают последствия действий (принятия решений) на основе комплекса моделей, включая все имитационные модели. *Оптимизирующие* модели находят оптимальные решения. *Предлагающие* модели дают специальное решение для четко структурированных задач. Такие системы выполняют механические вычисления и не оставляют места управленческому суждению.

Управление данными и моделями в СППР практически неразделимо, поэтому многие исследователи фокусируют внимание на обеих сторонах вопроса, на управлении и данными, и моделями. Данные представляют собой фактические результаты наблюдения за физическими явлениями, например это могут быть размеры суточного выпуска продукции, объем дневных продаж и уровень запасов продукта А. База данных содержит совокупность взаимосвязанных файлов.

Управление данными в СППР — это необходимая функция, используемая главным образом на интеллектуальной стадии процесса принятия решений, но недостаточная для поддержки этапов конструирования и выбора альтернатив. Для поддержки этих этапов СППР должны обеспечить выполнение следующих операций: проекцию, дедукцию, анализ, генерацию альтернатив, сравнение альтернатив, оптимизацию и имитацию. При выполнении этих задач СППР используют различные типы моделей из областей знаний об управлении и исследовании операций. Эти модели включают линейное программирование, целочисленное программирование, сетевые модели, программирование целей, имитационные и статистические модели и электронные таблицы.

Другой важной, развивающейся в последнее время разновидностью СППР являются системы поддержки принятия решений, основанные на базе знаний (СППРБЗ), которые представляют собой гибрид СППР и ЭС и предназначены для решения широкого круга организационных задач. В интегрировании СППР и ЭС выделяются два основных подхода: экспертные системы поддержки (ЭСП) и интеллектуальные системы поддержки (ИСП). Основное

различие между этими системами заключается в следующем — ЭСП предназначены для замены живого эксперта машинным экспертом, а задачи ИСП заключаются в поддержке знаний отдельных пользователей и групп. Широкий ряд управленческих задач реального мира легче поддается решению, если используются как количественные, так и качественные данные. Новая интегрированная система (ЭСП или ИСП) может помогать ЛПР, используя при этом знания и опыт ключевых фигур в организации. Узким местом при разработке систем, основанных на знаниях, таких как ЭСП, например, является задача приобретения знаний; этот процесс состоит из представления знаний, проверки, механизма построения логических выводов, механизмов объяснения и управления.

Множество новых средств и технологий способно расширить возможности СППР, среди них важное место занимает интеллектуальный анализ данных (ИАД), обычно определяемый как метод поддержки принятия решений, основанный на анализе зависимостей между данными [12, 23]. Существуют два подхода автоматизации поиска зависимостей между данными. В первом случае, используемом в традиционной технологии анализа, пользователь сам выдвигает гипотезы относительно зависимостей между данными. Гипотеза приводит к построению отчета, анализ отчета — к выдвижению новой гипотезы и т. д. Такая методика действует и в том случае, когда пользователь применяет такие развитые средства, как OLAP, поскольку процесс поиска по-прежнему полностью контролируется человеком. Во многих системах ИАД в этом процессе автоматизирована проверка достоверности гипотез, что позволяет оценить вероятность тех или иных зависимостей в базе данных. Второй подход основан на автоматическом поиске зависимости между данными. Растущее количество продуктов, реализующих автоматический поиск зависимостей, говорит об интересе производителей и потребителей к системам именно такого типа.

Процессы ИАД подразделяются на три большие группы: поиск зависимостей (discovery), прогнозирование (predictive modelling) и анализ аномалий (forensic

analysis). В первом случае при просмотре базы данных осуществляется автоматическое выявление зависимостей. Проблемой здесь является отбор действительно важных зависимостей из огромного числа существующих в БД. Прогнозирование основано на том, что пользователь предъявляет системе записи с незаполненными полями и запрашивает недостающие значения. Система сама анализирует содержимое базы и делает правдоподобное предсказание относительно этих значений. Анализ аномалий заключается в процессе поиска подозрительных данных, сильно отклоняющихся от устойчивых зависимостей.

Необходимость оперативной обработки стремительно возрастающих объемов информации вызвала к жизни появление и активное развитие технологии интеллектуального анализа данных (ИАД, или Data Mining). Теоретической базой ИАД являются методы математической статистики и искусственного интеллекта. Сложность решаемых задач потребовала создания специализированных инструментальных средств, ориентированных на конечного пользователя и предназначенных для решения как типовых, так и специфических задач в различных предметных областях.

Области применения ИАД: научные исследования, образование, статистика, здравоохранение, производство, финансы, правоохранительная и военная деятельность.

Разнообразие решаемых задач вынуждает разработчиков создавать исследовательские ИАД, предназначенные для работы с новыми типами проблем, и прикладные ИАД, ориентированные на решение типовых задач. Каждый из классов систем может быть ориентирован как на специалистов, так и непрограммирующих пользователей.

К сожалению, универсальные средства ИАД довольно сложны и до́роги, поэтому они не могут широко применяться в рамках интегрированных систем, ориентированных на конечного пользователя. В основу технологии ИАД положен не один, а несколько принципиально различных подходов (табл. 2.1), причем использование некоторых из них невозможно без специальной подготовки [8]. Выбор подхода нередко требует привлечения специалиста по ИАД.

Таблица 2.1

Основные технологии интеллектуального анализа данных

Технология	Область применения	Недостатки
Правила вывода	Данные связаны отношениями, представимыми в виде правил «если — то»	Потеря наглядности при большом количестве правил, не всегда удается выделить отношения «если — то»
Нейронные сети	Работа с нелинейными зависимостями, зашумленными и неполными данными	Модель типа «черный ящик» не позволяет объяснить выявленные знания, при этом данные обязательно должны быть преобразованы к числовому виду
Нечеткая логика	Ранжировка данных по степени близости к желаемым результатам; нечеткий поиск в базах данных	Ввиду новизны технологии в настоящее время известно ограниченное число специализированных приложений
Визуализация	Многомерное графическое представление данных, пользователю самому представляется возможность выявления закономерностей отношений между данными	Зависимость интерпретации модели от аналитика
Статистика	Научные и инженерные приложения, наличие большого числа алгоритмов и опыта их применения	Крен в сторону проверки гипотез, а не приоритет выявления новых закономерностей в данных
К-ближайший сосед	Выявление кластеров, обработка целостных источников данных	Требует больших объемов памяти, проблемы с чувствительностью
Интегрированные технологии	Выбор подходов, адекватных задачам, или их сравнение	Сложность средств поддержки; высокая стоимость. Для каждой отдельно взятой технологии не всегда реализуется наилучшее решение

Это наблюдение легло в основу подхода к поиску и выборке данных, называемого оперативной аналитической обработкой (On-line Analytical Processing, OLAP). Общеизвестно, что OLAP-системы построены на двух базовых принципах:

- все данные, необходимые для принятия решений, предварительно агрегированы на всех соответствующих уровнях и организованы так, чтобы обеспечить максимально быстрый доступ к ним;

- язык манипулирования данными основан на использовании бизнес-понятий.

Так как в основе OLAP лежит понятие гиперкуба, или многомерного куба данных, в ячейках которого хранятся анализируемые (числовые) данные, то дальнейшее усложнение модели данных может идти по нескольким направлениям:

- увеличение числа измерений, например данные о статьях не только по месяцам и годам, но и по темам. В этом случае куб становится трехмерным;
- усложнение содержимого ячейки — в этом случае в ячейке будет несколько значений признаков;
- введение иерархии в пределах одного измерения — общее понятие ВРЕМЯ естественным образом связано с иерархией значений: год состоит из кварталов, квартал из месяцев и т. д.

Речь пока идет не о физической структуре хранения, а лишь о логической модели данных. Другими словами, определяется лишь пользовательский интерфейс модели данных. В рамках этого интерфейса должны быть введены следующие базовые операции:

- поворот;
- проекция (при проекции значения в ячейках, лежащих на оси проекции, суммируются по некоторому предопределенному закону);
- раскрытие (drill-down) — одно из значений измерения заменяется совокупностью значений из следующего уровня иерархии измерения; соответственно заменяются значения в ячейках гиперкуба;
- свертка (roll-up/drill-up) — операция, обратная раскрытию;
- сечение (slice-and-dice).

В зависимости от ответа на вопрос, существует ли гиперкуб как отдельная физическая структура или лишь как виртуальная модель данных, различают системы MOLAP (Multidimensional OLAP) и ROLAP (Relational OLAP). В первых гиперкуб реализуется как отдельная база данных специальной нереляционной структуры, обеспечивающая максимально эффективный по скорости до-

ступ к данным, но требующая дополнительного ресурса памяти. MOLAP-системы весьма чувствительны к объемам хранимых данных. Поэтому данные из хранилища сначала помещаются в специальную многомерную базу (Multidimensional Data Base, MDB), а затем эффективно обрабатываются OLAP-сервером.

Реализация процесса принятия решений заключается в построении информационных приложений. Выделим в информационном приложении типовые функциональные компоненты, достаточные для формирования любого приложения на основе БД.

PS (Presentation Services) — средства представления. Обеспечиваются устройствами, принимающими ввод от пользователя и отображающими то, что сообщает ему компонент логики представления PL, плюс соответствующая программная поддержка. Может быть текстовым терминалом или *X*-терминалом, а также персональным компьютером или рабочей станцией в режиме программной эмуляции терминала или *X*-терминала.

PL (Presentation Logic) — логика представления. Управляет взаимодействием между пользователем и ЭВМ. Обрабатывает действия пользователя по выбору альтернативы меню, по нажатию кнопки или выборе элемента из списка.

BL (Business or Application Logic) — прикладная логика. Набор правил для принятия решений, вычислений и операций, которые должно выполнить приложение.

DL (Data Logic) — логика управления данными. Операции с базой данных (SQL-операторы SELECT, UPDATE и INSERT), которые нужно выполнить для реализации прикладной логики управления данными.

DS (Data Services) — операции с базой данных. Действия СУБД, вызываемые для выполнения логики управления данными, такие как манипулирование данными, определение данных, фиксация или откат транзакций и т. п. СУБД обычно компилирует SQL-приложения.

FS (File Services) — файловые операции. Дисковые операции чтения и записи данных для СУБД и других компонент. Обычно являются функциями ОС.

Среди средств разработки информационных приложений можно выделить следующие основные группы:

- традиционные системы программирования;
- инструменты для создания файл-серверных приложений;
- средства разработки приложений клиент-сервер;
- средства автоматизации делопроизводства и документооборота;
- средства разработки интернет/интранет-приложений;
- средства автоматизации проектирования приложений.

2.4. ХРАНЕНИЕ ИНФОРМАЦИИ

Хранение и накопление являются одними из основных действий, осуществляемых над информацией и главным средством обеспечения ее доступности в течение некоторого промежутка времени. В настоящее время определяющим направлением реализации этой операции является концепция базы данных, склада (хранилища) данных [9], [13].

База данных может быть определена как совокупность взаимосвязанных данных, используемых несколькими пользователями и хранящихся с регулируемой избыточностью. Хранимые данные не зависят от программ пользователей, для модификации и внесения изменений применяется общий управляющий метод.

Система баз данных — совокупность управляющей системы, прикладного программного обеспечения, базы данных, операционной системы и технических средств, обеспечивающих информационное обслуживание пользователей.

Хранилище данных (склад данных, информационное хранилище) (ХД) (Data Warehouse) — это база данных, хранящая данные, агрегированные по многим измерениям. Основные отличия ХД от БД: агрегирование данных; данные из ХД никогда не удаляются; пополнение ХД происходит на периодической основе; автоматическое формирование новых агрегатов данных, зависящих от старых;

доступ к ХД осуществляется на основе многомерного куба или гиперкуба.

Альтернативой хранилищу данных является концепция витрин данных (Data Mart). Витрины данных — множество тематических БД, содержащих информацию, относящуюся к отдельным информационным аспектам предметной области.

Еще одним важным направлением развития баз данных являются репозитарии. Репозитарий в упрощенном виде можно рассматривать просто как базу данных, предназначенную для хранения не пользовательских, а системных данных. Технология репозитариев проистекает от словарей данных, которые по мере обогащения новыми функциями и возможностями приобретали черты инструмента для управления метаданными.

Каждый из участников действия имеет свое представление об информации:

- пользователь;
- группа пользователей;
- «физическая память».

По отношению к пользователям принято использовать трехуровневое представление для описания предметной области: концептуальное, логическое и внутреннее (физическое) (рис. 2.12).

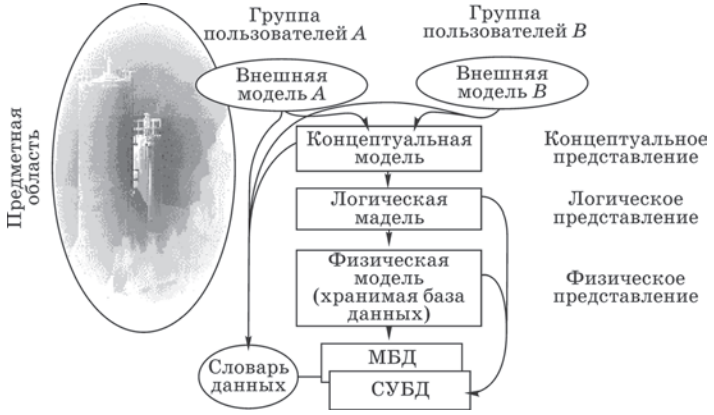


Рис. 2.12
Описания предметной области

Концептуальный уровень связан с частным представлением данных группы пользователей в виде внешней схемы, объединяемых общностью используемой информации. Каждый конкретный пользователь работает с частью БД и представляет ее в виде внешней модели. Данный уровень характеризуется разнообразием используемых моделей (модель «сущность — связь» (ER-модель, модель Чена), бинарные и инфологические модели, семантические сети)). На рисунке 2.13 представлен фрагмент предметной базы данных «Сбыт» и одно из возможных его концептуальных представлений, которое отражает не только объекты, и их свойства, но и взаимосвязи между ними.

Логический уровень является обобщенным представлением данных всех пользователей в абстрактной форме. Используются три классических вида моделей: иерархические, сетевые и реляционные.

Сетевая модель является моделью объектов-связей, допускающей только бинарные связи «многие к одному»,

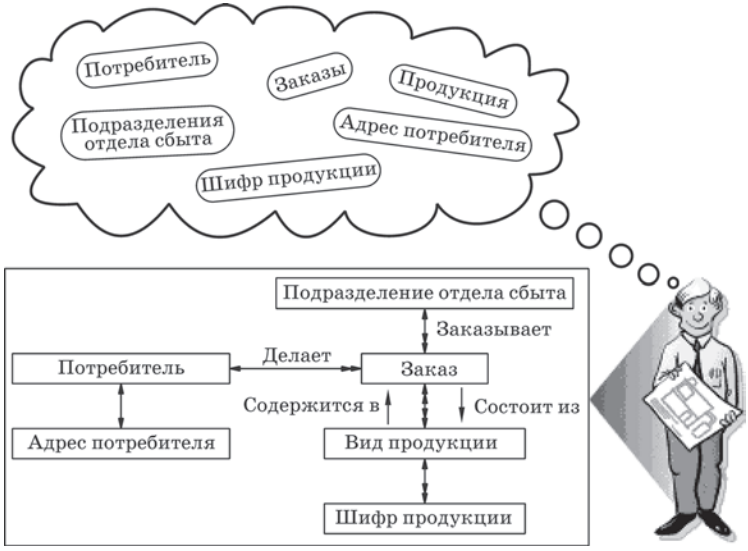


Рис. 2.13
Фрагмент предметной базы данных «Сбыт» и одно из возможных его концептуальных представлений

и использует для описания модель ориентированных графов.

Иерархическая модель является разновидностью сетевой, являющейся совокупностью деревьев (лесом).

Реляционная модель использует представление данных в виде таблиц (реляций), в основе лежит математическое понятие теоретико-множественного отношения, базируется на реляционной алгебре и теории отношений.

Представление предметной базы данных «Сбыт» на логическом уровне для различных моделей показано на рисунке 2.14.

Дальнейшим развитием логического представления данных является переход к объектно-ориентированным моделям данных, что связано с процессом «перекачки» в них огромных объемов информации, которая в настоящее время хранится преимущественно в реляционных базах данных.

Суть объектно-ориентированной базы данных (ООБД) определяется объектно-ориентированным подходом.

Объектно-ориентированный подход подразумевает объектно-ориентированное проектирование и объектно-ориентированное программирование.

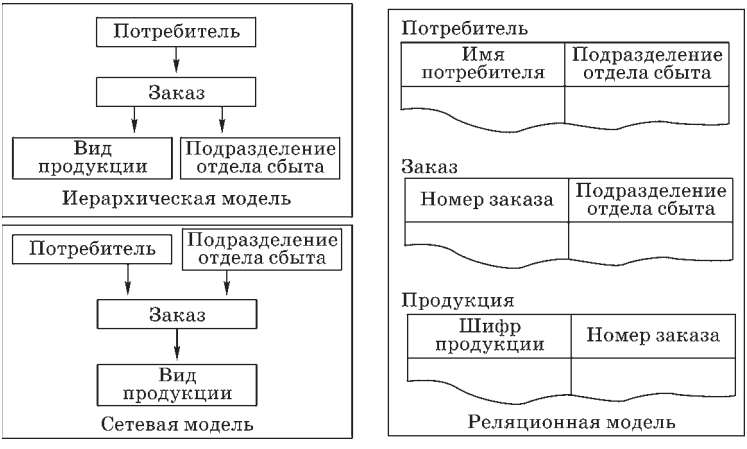


Рис. 2.14
Представление предметной базы данных «Сбыт» на логическом уровне для различных моделей

Объектно-ориентированное программирование — методология программирования, основанная на представлении программ в виде связанной совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию по наследованию.

Объектно-ориентированное проектирование предполагает не только деление (декомпозицию) базы знаний или базы данных на составные части, но и рассмотрение общей этапности реализации БД, выбор инструмента реализации с учетом оговоренных в техническом задании вариантов реализации.

Чтобы упростить процесс перехода к ООБД, сформировали объектно-реляционную модель данных, в которой выделяются две разновидности — гибридные и расширенные.

Объектно-реляционная модель (ОРБД) является промежуточной моделью данных между реляционными и объектно-ориентированными базами данных. Ее появление вызвано двумя причинами:

1) сложностью построения новой модели данных «с листа». Удобнее это делать на основе имеющихся проверенных разработок;

2) учет преемственности с широко используемыми реляционными моделями, которые нельзя мгновенно заменить на объектно-ориентированные БД.

Различают, как отмечалось ранее, две разновидности ОРБД — гибридные и расширенные.

1. Гибридные. Интерфейс пользователя и алгоритм приложения выполнены с учетом объектно-ориентированного подхода, тогда как собственно БД является реляционной. Примерами могут служить СУБД Paradox и InterBase в рамках программного продукта Delphi. В каком-то смысле гибридной можно считать СУБД Access при использовании языка программирования Visual Basic for Application (VBA).

2. Расширенные (постреляционные). В них предполагается объектно-ориентированное построение собственно базы данных путем использования известных и введения новых типов данных, связанных между со-

бой. Эта связь чаще всего осуществляется созданием методов с помощью триггеров и хранимых процедур. В расширенной объектно-реляционной модели [3], [4], [36] допускается, в отличие от реляционной модели данных, неатомарность данных в поле. В таких полях может располагаться другая таблица или массив. К подобным СУБД относятся Informix Universal Server, Oracle 8, UniSQL.

Свойства реляционных, сетевых, иерархических, объектно-иерархических и объектно-реляционных моделей данных (многомерные модели считаем разновидностью иерархических моделей данных) можно систематизировать и сравнить, как это сделано в таблице 2.2.

Сравнительная характеристика

Вид модели	Достоинства
Иерархическая	Простота понимания. Высокое быстродействие при совпадении структур базы данных и запроса
Сетевая	Сохранение информации при уничтожении записи-владельца. Более богатая структура запросов. Меньшая зависимость логической и физической моделей. Возможность реализации таблиц с нелинейной структурой
Реляционная	Произвольная структура запроса. Простота работы и отражения представлений пользователя. Отделение физической модели от логической и логической от концептуальной. Хорошая теоретическая проработка
Объектно-ориентированная	Неограниченный набор типов данных. Возможность реализации таблиц с нелинейной структурой. Послойное представление данных. Высокая скорость работы из-за отсутствия ключа. Ненужность нормализации. Легкая расширяемость структуры и ее гибкость. Повторное использование типов данных и компонент. Реализация отношений M:M

Физический (внутренний) уровень связан со способом фактического хранения данных в физической памяти ЭВМ. Во многом определяется конкретным методом управления. Основными компонентами физического уровня являются хранимые записи, объединяемые в блоки; указатели, необходимые для поиска данных; данные переполнения; промежутки между блоками; служебная информация.

По наиболее характерным признакам БД можно классифицировать следующим образом:

1) по способу хранения информации:

- интегрированные;
- распределенные;

Таблица 2.2

моделей БД

	Недостатки
	Отношения М:М могут быть реализованы только искусственно. Могут быть избыточные данные. Усложняются операции включения и удаления. Удаление исходных сегментов приводит к удалению порожденных сегментов. Процедурный характер построения структуры БД и манипулирования данными. Доступ к любому порожденному сегменту возможен только через корневой сегмент. Сильная зависимость логической и физической моделей. Ограниченный набор структур запроса. Невозможность реализации таблиц с нелинейной структурой
	Отношения М:М могут быть реализованы только искусственно. Необходимость программисту знать логическую структуру БД. Процедурный характер построения структуры БД и манипулирования данными. Возможная потеря независимости данных при реорганизации БД
	Отношения М:М могут быть реализованы только искусственно. Необходимость нормализации данных. Возможность логических ошибок при нормализации и реализации. Невозможность реализации таблиц с нелинейной структурой
	Сложность освоения модели из-за сложности структуры БД. Нечеткий язык программирования. Недостаточная защита данных. Нечетко проработанный одновременный доступ. Плохая обозримость структуры

2) по типу пользователя:

- монопользовательские;
- многопользовательские;

3) по характеру использования данных:

- прикладные;
- предметные.

В настоящее время при проектировании БД используют два подхода. Первый из них основан на стабильности данных, что обеспечивает наибольшую гибкость и адаптируемость к используемым приложениям. Применение такого подхода целесообразно в тех случаях, когда не предъявляются жесткие требования к эффективности функционирования (объем памяти и время поиска), существует большое количество разнообразных задач с изменяемыми и непредсказуемыми запросами.

Другой подход базируется на стабильности процедур запросов к БД и является предпочтительным при жестких требованиях к эффективности функционирования, особенно это касается быстродействия.

Другим важным аспектом проектирования БД является проблема интеграции и распределения данных. Господствовавшая до недавнего времени концепция интеграции данных при резком увеличении их объема оказалась несостоятельной. Этот факт, а также увеличение объемов памяти внешних запоминающих устройств при ее удешевлении, широкое внедрение сетей передачи данных способствовали внедрению распределенных БД. Распределение данных по месту их использования может осуществляться различными способами:

1) копируемые данные. Одинаковые копии данных хранятся в различных местах использования, так как это дешевле передачи данных. Модификация данных контролируется централизованно;

2) подмножество данных. Группы данных, совместимые с исходной базой данных, хранятся отдельно для местной обработки;

3) реорганизованные данные. Данные в системе интегрируются при передаче на более высокий уровень;

4) секционированные данные. На различных объектах используются одинаковые структуры, но хранятся разные данные;

5) данные с отдельной подсхемой. На различных объектах используются различные структуры данных, объединяемые в интегрированную систему;

6) несовместимые данные. Независимые базы данных, спроектированные без координации, требующие объединения.

Важное влияние на процесс создания БД оказывает внутреннее содержание информации. Существуют два направления:

- прикладные БД, ориентированные на конкретные приложения, например может быть создана БД для учета и контроля поступления материалов;
- предметные БД, ориентированные на конкретный класс данных, например предметная БД «Материалы», которая может быть использована для различных приложений.

Конкретная реализация системы баз данных, с одной стороны, определяется спецификой данных предметной области, отраженной в концептуальной модели, а с другой стороны, типом конкретной СУБД (МБД), устанавливающей логическую и физическую организацию.

Для работы с БД используется специальный обобщенный инструментарий в виде СУБД (МБД), предназначенный для управления БД и обеспечения интерфейса пользователя.

Основные стандарты СУБД:

- независимость данных на концептуальном, логическом, физическом уровнях;
- универсальность (по отношению к концептуальному и логическому уровню, типу ЭВМ);
- совместимость, неизбыточность;
- безопасность и целостность данных;
- актуальность и управляемость.

Существуют два основных направления реализации СУБД: программное и аппаратное.

Программная реализация (в дальнейшем — СУБД) представляет собой набор программных модулей, работа-

ет под управлением конкретной ОС и выполняет следующие функции:

- описание данных на концептуальном и логическом уровнях;
- загрузку данных;
- хранение данных;
- поиск и ответ на запрос (транзакцию);
- внесение изменений;
- обеспечение безопасности и целостности.

Обеспечивает пользователя следующими языковыми средствами:

- язык описания данных (ЯОД);
- язык манипулирования данными (ЯМД);
- прикладной (встроенный) язык данных (ПЯД, ВЯД).

Аппаратная реализация предусматривает использование так называемых машин баз данных (МБД). Их появление вызвано возросшими объемами информации и требованиями к скорости доступа. Слово «машина» в термине МБД означает вспомогательный периферийный процессор. Термин «компьютер БД» — автономный процессор баз данных или процессор, поддерживающий СУБД. Основные направления МБД:

- параллельная обработка;
- распределенная логика;
- ассоциативные ЗУ;
- конвейерные ЗУ;
- фильтры данных и др.

На рисунке 2.15 представлена совокупность процедур проектирования БД, которые можно объединить в четыре этапа.

На этапе формулирования и анализа требований устанавливаются цели организации, определяются требования к БД. Эти требования документируются в форме доступной конечному пользователю и проектировщику БД. Обычно при этом используется методика интервьюирования персонала различных уровней управления.

Этап концептуального проектирования заключается в описании и синтезе информационных требований пользователей в первоначальный проект БД. Результатом это-

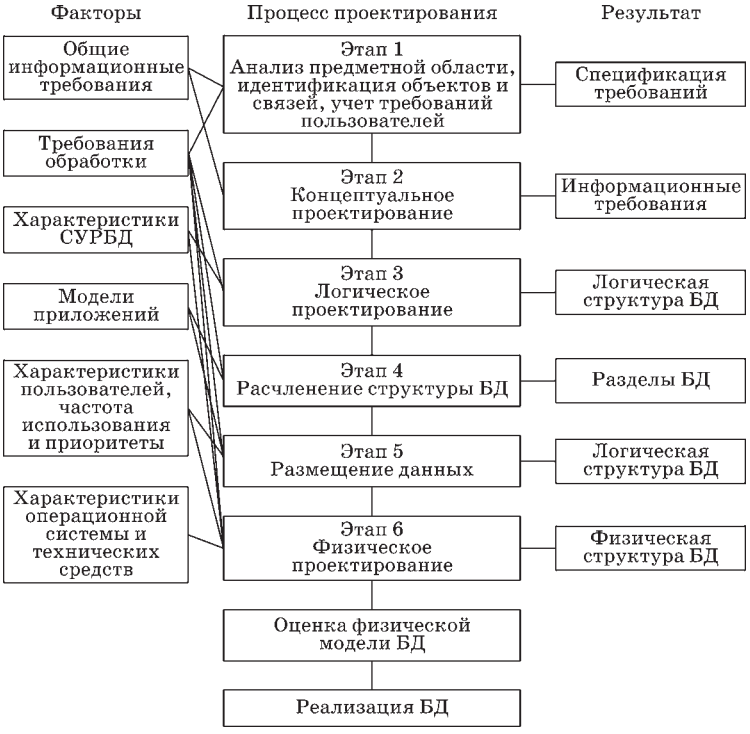


Рис. 2.15
Совокупность процедур проектирования БД

го этапа является высокоуровневое представление информационных требований пользователей на основе различных подходов.

В процессе логического проектирования высокоуровневое представление данных преобразуется в структуре используемой СУБД. Полученная логическая структура БД может быть оценена количественно с помощью различных характеристик (число обращений к логическим записям, объем данных в каждом приложении, общий объем данных и т. д.). На основе этих оценок логическая структура может быть усовершенствована с целью достижения большей эффективности.

На этапе физического проектирования решаются вопросы, связанные с производительностью системы,

определяются структуры хранения данных и методы доступа.

Весь процесс проектирования БД является итеративным, при этом каждый этап рассматривается как совокупность итеративных процедур, в результате выполнения которых получают соответствующую модель.

Взаимодействие между этапами проектирования и словарной системой необходимо рассматривать отдельно. Процедуры проектирования могут использоваться независимо в случае отсутствия словарной системы. Сама словарная система может рассматриваться как элемент автоматизации проектирования.

Этап расчленения БД связан с разбиением ее на разделы и синтезом различных приложений на основе модели. Основными факторами, определяющими методику расчленения, помимо указанных на рисунке, являются: размер каждого раздела (допустимые размеры); модели и частоты использования приложений; структурная совместимость; факторы производительности БД. Связь между разделом БД и приложениями характеризуется идентификатором типа приложения, идентификатором узла сети, частотой использования приложения и его моделью.

Модели приложений могут быть классифицированы следующим образом:

1) приложения, использующие единственный файл;

2) приложения, использующие несколько файлов, в том числе:

- допускающие независимую параллельную обработку;
- допускающие синхронизированную обработку.

Сложность реализации этапа размещения БД определяется многовариантностью. Поэтому на практике рекомендуется в первую очередь рассмотреть возможность использования определенных допущений, упрощающих функции СУБД, например допустимость временного рассогласования БД, осуществление процедуры обновления БД из одного узла и др. Такие допущения оказывают большое влияние на выбор СУБД и рассматриваемую фазу проектирования.

Средства проектирования и оценочные критерии используются на всех стадиях разработки. Любой метод проектирования (аналитический, эвристический, процедурный), реализованный в виде программы, становится инструментальным средством проектирования практически не подверженным влиянию стиля проектирования.

В настоящее время неопределенность при выборе критериев является наиболее слабым местом в проектировании БД. Это связано с трудностью описания и идентификации бесконечного числа альтернативных решений. При этом следует иметь в виду, что существует много признаков оптимальности, являющихся неизмеримыми свойствами, которые трудно выразить в количественном представлении или в виде целевой функции. Поэтому оценочные критерии принято делить на количественные и качественные. Наиболее часто используемые критерии оценки БД, сгруппированные в такие категории, представлены ниже.

Количественные критерии: время ответа на запрос, стоимость модификации, стоимость памяти, время на создание, стоимость на реорганизацию.

Качественные критерии: гибкость, адаптивность, доступность для новых пользователей, совместимость с другими системами, возможность конвертирования в другую вычислительную среду, возможность восстановления, возможность распределения и расширения.

Трудность в оценке проектных решений связана также с различной чувствительностью и временем действия критериев. Например, критерий эффективности обычно является краткосрочным и чрезвычайно чувствительным к проводимыми изменениям, а такие понятия, как «адаптируемость» и «конвертируемость», проявляются на длительных временных интервалах и менее чувствительны к воздействию внешней среды.

Предназначение склада данных — информационная поддержка принятия решений, а не оперативная обработка данных. Потому база данных и склад данных не являются одинаковыми понятиями. Архитектура ХД представлена на рисунке 2.16.

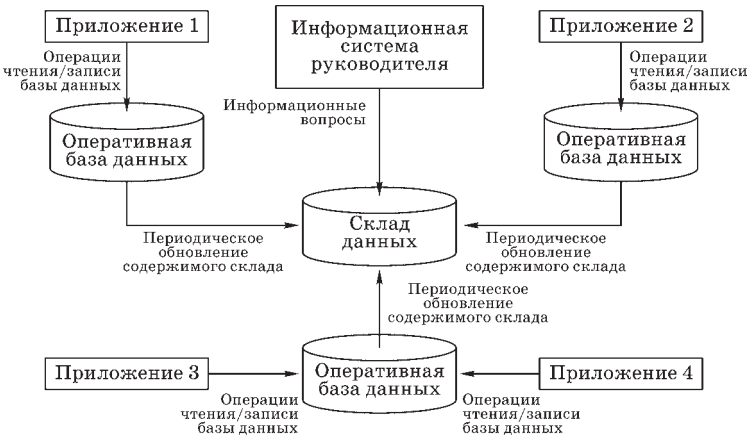


Рис. 2.16
Архитектура ХД

Основные принципы организации хранилищ данных [39].

1. Предметная ориентация. В оперативной базе данных обычно поддерживается несколько предметных областей, каждая из которых может послужить источником данных для ХД. Например, для магазина, торгующего видео- и музыкальной продукцией, интерес представляют следующие предметные области: клиенты, видеокассеты, CD-диски и аудиокассеты, сотрудники, поставщики. Явно прослеживается аналогия между предметными областями ХД и классами объектов в объектно-ориентированных базах данных. Это говорит о возможности применения методов проектирования, применяемых в объектно-ориентированных СУБД.

2. Средства интеграции. Приведение разных представлений одних и тех же сущностей к некоторому общему типу.

3. Постоянство данных. В ХД не поддерживаются операции модификации в смысле традиционных баз данных. В ХД поддерживается модель «массовых загрузок» данных, производимых в заданные моменты времени по установленным правилам, в отличие от традиционной модели индивидуальных модификаций.

4. Хронология данных. Благодаря средствам интеграции реализуется определенный хронологический временной аспект, присущий содержимому ХД.

Основные функции репозитариев:

- парадигма включения/выключения и некоторые формальные процедуры для объектов;
- поддержка множественных версий объектов и процедуры управления конфигурациями для объектов;
- способность оповещения инструментальных и рабочих систем об интересующих их событиях;
- управление контекстом и разные способы обзора объектов репозитария;
- возможность определения потоков работ.

Рассмотрим кратко основные направления научных исследований в области баз данных:

- развитие теории реляционных баз данных;
- моделирование данных и разработка конкретных моделей разнообразного назначения;
- отображение моделей данных, направленных на создание методов преобразования моделей данных и конструирования коммутативных отображений, разработку архитектурных аспектов отображения моделей данных и спецификаций определения отображений для конкретных моделей данных;
- СУБД с мультимодельным внешним уровнем, обеспечивающие возможности отображения широко распространенных моделей;
- разработка, выбор и оценка методов доступа;
- самоописываемые базы данных, позволяющие применять единые методы доступа для данных и метаданных;
- управление конкурентным доступом;
- системы программирования баз данных и знаний, которые обеспечивали бы единую эффективную среду как для разработки приложений, так и для управления данными;
- машины баз данных;
- дедуктивные базы данных, основанные на применении аппарата математической логики и средств логического программирования в области баз данных;

- пространственно-временные базы данных;
- интеграция неоднородных информационных ресурсов.

Серьезные недостатки реляционной модели данных привели к необходимости поиска других моделей. Такой прогрессивной и перспективной моделью данных является объектно-ориентированная модель данных.

В ней собственно база данных, интерфейс пользователя и алгоритм приложения построены с использованием объектно-ориентированного подхода.

Автор реляционной модели данных Э. Ф. Кодд первоначально сформулировал 12 требований к БД, чтобы она могла называться реляционной. В дальнейшем этот перечень увеличился до 333 требований. Им, несмотря на широкое распространение реляционных баз данных, не удовлетворяет ни одна из известных СУБД.

Более того, при значительных объемах данных начинают проявляться недостатки реляционных баз данных. К этим недостаткам относятся: сложность структуры, вызванная необходимостью проведения нормализации; низкая производительность из-за поиска по ключу, что в 3–5 раз увеличивает количество операций доступа; ограниченный набор типов данных; представление данных только в виде двумерных таблиц и невозможность реализации таблиц с нелинейной структурой; невозможность послойного рассмотрения данных (например, РАБОТАЮЩИЕ — в одном слое, научные сотрудники и преподаватели — в другом, подчиненном слое); нестыковка с принципами все более широко применяемого объектно-ориентированного подхода; невозможность задать для определенного типа данных набор операторов-методов, которые приходится вводить в конкретном приложении; возникновение конфузии — утраты при многочисленных обновлениях третьей (а порой — и второй) нормальной формы; сложность совмещения с другой парадигмой хранилищ данных.

Суть объектно-ориентированной БД определяется особенностями объектной модели. Выделяется несколько специфических понятий. Данные называют *свойствами*, а алгоритмы — *методами*. Доступ к классу осуществляется через свойства — в статическом режиме написания

и отладки программы или через методы — в процессе выполнения (работы) программы.

Начало работы элемента класса задается с помощью специальных внутренних (например, нажатие кнопки) или внешних (вызов из другой подпрограммы) сигналов, называемых *событием*.

Программную реализацию класса называют компонентой. Реализация компоненты в некоторой программе получила название объекта.

В объектно-ориентированном программировании используются три основных принципа: инкапсуляция, наследование, плиморфизм.

Инкапсуляция — выделение класса с доступом к нему через свойства или методы.

Наследование — трансформация класса путем изменения свойств и методов с помощью методов, называемых конструктором и деструктором.

Все классы (компоненты) строятся по иерархическому принципу с происхождением от некоторого исходного класса.

Полиморфизм позволяет использовать метод с одним именем как в базовом, так и в производных классах. Дело в том, что количество классов значительно: уже сейчас оно приближается к ста и продолжает расти. Если в производных классах применять для однотипных, «похожих» методов (например, начертание квадрата и окружности) разные имена, пользователю, особенно начинающему, будет сложно освоиться с таким разнообразием имен.

Приведем основные положения ООБД, базирующиеся на объектно-ориентированном подходе:

1) в качестве значения столбца может быть использован произвольный кортеж. Иными словами, столбец может являться как бы некоторой другой таблицей. Таким образом создается возможность реализации таблиц с нелинейной структурой;

2) процедура манипуляции данными позволяет присоединять процедуры, определенные значениями столбцов;

3) данные столбцов могут наследоваться;

4) элементами отношений могут служить не только отдельные элементы, как в реляционных БД, но и множества;

5) формируются классы данных, которые организуют столбцы в иерархию.

Базовым языком ООБД чаще всего является C^{++}. Для работы с такими ООБД разрабатывается новый вариант языка программирования SQL, получивший название SQL3 и содержащий в качестве частного случая реляционную модель (SQL2). Если SQL2 определяет семь способов связывания со стандартными языками программирования, в SQL3 это количество предполагается увеличить.

В технологии разработки ООБД конкурируют два направления:

1) Distributed Object Linking and Embedding (OLE) фирмы Microsoft;

2) Common Object Request Broker Architecture (CORBA) группы OBDMG, поддерживаемая фирмами IBM, Novell, DEC, с ориентацией на все платформы. В рамках этого направления выделены и сформированы указанные ранее язык определения объектов Object Definition Language (ODL); объектный язык запроса Object Query Language (OQL); язык определения интерфейсов Interface Definition Language (IDL).

Во втором направлении обычно выделяют [44] два стандарта управления БД:

- ODL/OQL, в котором объекты и методы формируются обычно с помощью языка программирования C;
- язык программирования SQL3.

Необходимо отметить, что к объектно-ориентированным возможно отнести только те базы данных, у которых все структурные элементы реализации построены с использованием объектно-ориентированного подхода. Если хотя бы один структурный элемент реализации не использует объектно-ориентированный подход, говорят об *объектно-реляционной базе данных* (ОРБД).

Таким образом, чтобы воспользоваться объектно-ориентированным подходом в построении собственно БД, необходимо:

- провести инкапсуляцию данных, т. е. выделить классы и объекты;
- определить возможные виды структуры реализуемых таблиц;
- создать наследование классов данных;
- обеспечить полиморфизм.

Реализация даже первой позиции неоднозначна и различна, например в ООСУБД и ОРСУБД. Имеется некоторое отличие даже в рамках различных ООСУБД.

По сравнению с реляционными БД ООБД обладают следующими преимуществами.

1) лучшие возможности моделирования систем из блоков, обладающих произвольными связями;

2) легкая расширяемость структуры за счет создания новых типов данных (свойств), наследования, установления новых связей и корректировки методов;

3) возможность использования рекурсивных методов при навигационном методе доступа к большим объемам данных;

4) повышение производительности в 10–30 раз;

5) более широкая сфера применения (например, использование в мультимедиасистемах).

Преимущества ООБД [13] приведут, видимо, к очень широкому их распространению. Однако прежде следует решить ряд задач по устранению недостатков ООБД: создать гибкую структуру БД; построить четкий язык программирования; отработать синтаксис разбора запросов, в том числе вложенных; определить несколько методов доступа к данным; отработать вопросы одновременного доступа (разрешение конфликтов при множественном наследии); определить сложный перебор данных; отработать защиту и восстановление данных; уточнить семантику (действия) операторов при динамических изменениях; встроить изменение атрибутов дочерних объектов.

Переход к объектно-ориентированным моделям данных связан с процессом «перекачки» в них огромных объемов информации, которая в настоящее время хранится преимущественно в реляционных базах данных. Чтобы упростить этот процесс, сформировали объектно-

реляционную модель данных, в которой выделяются две разновидности — гибридные и расширенные.

В гибридных объектно-реляционных базах данных объектно-ориентированный подход используется для создания интерфейса пользователя и алгоритма приложения. В то же время система таблиц формируется в рамках реляционной модели данных.

В расширенных объектно-реляционных базах данных объектно-ориентированный подход используется прежде всего при построении системы таблиц. Для этого разработана модификация языка SQL2, получившая название языка программирования SQL3.

«Промежуточной» моделью данных между реляционными и объектно-ориентированными базами данных является объектно-реляционная модель (ОРБД).

Ее появление вызвано двумя причинами:

1) сложностью построения новой модели данных «с листа». Удобнее это делать на основе имеющихся проверенных разработок;

2) учетом преемственности с широко используемыми реляционными моделями, которые нельзя мгновенно заменить на объектно-ориентированные БД.

Различают, как отмечалось ранее, две разновидности ОРБД — гибридные и расширенные.

Гибридные [13]. Интерфейс пользователя и алгоритм приложения выполнены с учетом объектно-ориентированного подхода, тогда как собственно БД является реляционной. Примерами могут служить СУБД Paradox и InterBase в рамках программного продукта Delphi. В каком-то смысле гибридной можно считать СУБД Access при использовании языка программирования Visual Basic for Application (VBA).

Расширенные (*постреляционные*). В них предполагается объектно-ориентированное построение собственно базы данных путем использования известных и введения новых типов данных, связанных между собой. Эта связь чаще всего осуществляется созданием методов с помощью триггеров и хранимых процедур. В расширенной объектно-реляционной модели допускается, в отличие

от реляционной модели данных, неатомарность данных в поле. В таких полях может располагаться другая таблица или массив. К подобным СУБД относятся Informix Universal Server, Oracle 8, UniSQL. В таких СУБД широко используется язык SQL3.

Заметим, что постреляционными называют БД, в которых частично проведена денормализация данных, при этом допускается неатомарность записей. Рассмотрим более подробно обе разновидности.

Чтобы понять перспективы развития ОРБД, следует оценить их достоинства и недостатки.

К достоинствам следует отнести:

- устранение ряда недостатков реляционных БД;
- повторное использование компонентов;
- использование накопленных знаний по реляционным БД.

К недостаткам ОРБД можно отнести:

- усложнение структуры БД и частичную утрату простой обозримости результатов, как в реляционных БД;
- сложность построения абстрактных типов данных и методов, связывающих типы в иерархию;
- менее широкий набор типов связей, определяемых языком программирования SQL, чем в объектно-ориентированных БД;
- менее продуманный, отлаженный и стандартизованный набор типов данных, чем в ООБД.

ОРБД, по-видимому, будут существовать еще достаточно долго, чему есть по меньшей мере два объяснения:

1) быстрое накопление с помощью ОРБД опыта, который можно использовать при создании ООСУБД;

2) необходимость иметь средства для постепенного эволюционного «перевода» многочисленных реляционных БД в разряд ООБД, за которыми, видимо, будущее.

В последнее время распределенные базы данных (РБД) находят все более широкое применение в связи с массовым распространением «сетевых» технологий.

Теория создания, использования и функционирование РБД имеет свои особенности по сравнению с централизованными БД.

Быстрое распространение сетей передачи данных, резкое увеличение объема внешней памяти ПК при ее удешевлении в 1980-е гг., развитие возможностей персональных компьютеров, которые по своим характеристикам в 1990-е гг. уже превосходили суперЭВМ 1980-х гг., создали необходимую базу для реализации и широкого внедрения РБД.

К достоинствам РБД относятся:

- соответствие структуры РБД структуре организаций;
- гибкое взаимодействие локальных БД;
- широкие возможности централизации узлов;
- непосредственный доступ к информации, снижение стоимости передач (за счет уплотнения и концентрации данных);
- высокие системные характеристики (малое время отклика за счет распараллеливания процессов, высокая надежность);
- модульная реализация взаимодействия или расширение аппаратных средств, возможность использования объектно-ориентированного подхода в программировании;
- возможность распределения файлов в соответствии с их активностью;
- независимые разработки локальных БД через стандартный интерфейс.

Вместе с тем РБД обладают более сложной структурой, что вызывает появление дополнительных проблем (избыточность, несогласованность данных по времени, согласование процессов обновления и запросов, использования телекоммуникационных ресурсов, учет работы дополнительно подсоединенных локальных БД, стандартизация общего интерфейса, усложнение защиты данных) согласования работы элементов.

Серьезные проблемы возникают при интеграции в рамках РБД однородных (гомогенных) локальных БД с одинаковыми, чаще всего реляционными, моделями данных.

Проблемы значительно усложняются, если локальные БД построены с использованием различных моделей данных (неоднородные, гетерогенные РБД).

Возникла необходимость в теоретической проработке процессов в РБД.

Распределенная база данных (РБД) — система логически интегрированных и территориально распределенных БД, языковых, программных, технических и организационных средств, предназначенных для создания, введения и обработки информации.

Это означает, что информация физически хранится на разных ЭВМ, связанных сетью передачи данных. Любой узел (участок) может выполнять приложение и участвовать в работе по крайней мере одного приложения.

Большинство требований, предъявляемых к РБД, аналогично требованиям к централизованным БД, но их реализация имеет свою, рассматриваемую ниже специфику. В частности, в РБД иногда полезна избыточность.

Дополнительными специфическими требованиями являются:

1) ЯОД в рамках схемы должен быть один для всех локальных БД;

2) доступ должен быть коллективным к любой области РБД с соответствующей защитой информации;

3) подсхемы должны быть определены в месте сосредоточения алгоритмов (приложений, процессов) пользователя;

4) степень централизации должна быть разумной;

5) необходим сбор и обработка информации об эффективности функционирования РБД.

В дальнейшем К. Дейт сформулировал 12 правил для РБД:

1) локальная автономность;

2) отсутствие опоры на центральный узел;

3) непрерывное функционирование (развитие) РБД;

4) независимость РБД от расположения локальных БД;

5) независимость от фрагментации данных;

6) независимость от репликации (дублирования) данных;

7) обработка распределенных запросов;

8) обработка распределенных транзакций;

9) независимость от типа оборудования;

10) независимость от операционной системы;
11) независимость от сетевой архитектуры;
12) независимость от типа СУБД.

В дальнейшем рассмотрении РБД выделим:

- общие вопросы (состав, работа РБД);
- теоретические вопросы РБД, в которых по-прежнему выделим три составляющих (создание, использование и функционирование РБД).

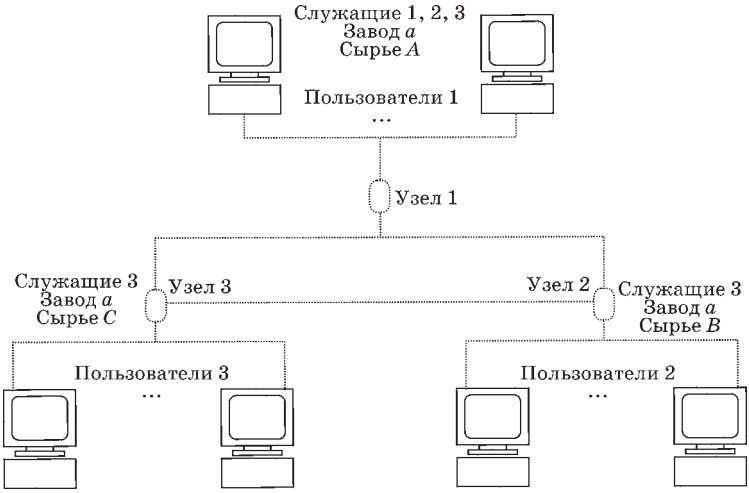


Рис. 2.17
Схема распределенной базы данных

Стратегии хранения

Название	Суть стратегии	Достоинство
Централизация (в том числе технология клиент-сервер)	Единственная копия в одном узле	Простота структуры
Локализация (расчленение)	Единственная копия, расчленение по узлам (полная копия БД не допускается)	Объем БД определяется памятью сети. Снижение стоимости РБД. Время отклика при параллельной обработке уменьшается. Малая чувствительность к узким местам. Повышенная надежность при высокой локализации данных

Схема РБД может быть представлена в виде, показанном на рисунке 2.17.

В ней выделяют пользовательский, глобальный (концептуальный), фрагментарный (логический) и распределенный (локальный) уровни представления данных, определяющие сетевую СУБД.

Сравнительные характеристики стратегий хранения приведены в таблице 2.3. На ее основе может быть построен простейший алгоритм выбора стратегии, показанный на рисунке 2.18.

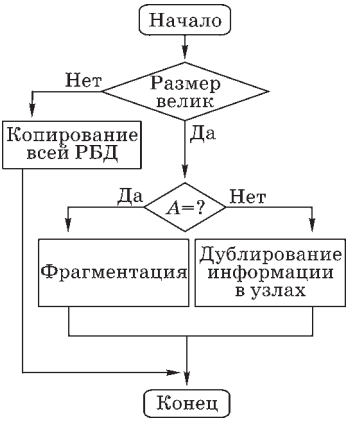


Рис. 2.18
Алгоритм выбора стратегии хранения данных:
A — запрос локален.

Отметим, что в обычной сети имеет место равноправие компьютеров, что может вызвать дополнительные осложнения в части доступа к данным в процедурах обновления и запросов.

В связи с этим часто используют архитектуру «клиент-сервер» (см. п. 3.5) — структуру локальной сети, в которой применено распределенное управление сервером и рабочими станциями (клиентами) для максимально эффективного использования вычислительной мощности.

Таблица 2.3

данных

	Недостатки
	Скорость обработки ограничена одним узлом. Ограниченный доступ. Малая надежность. Долговременная память определяет объем БД
	Запрос может быть по всем узлам. Доступ хуже, чем при централизации. Рекомендации применения. Долговременная память ограничена по сравнению с объемом БД. Должна быть повышена эффективность функционирования при высокой степени локализации

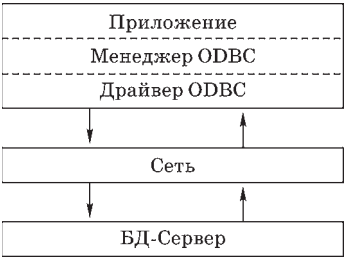


Рис. 2.19
ODBC в архитектуре «клиент-сервер»

Работа в архитектуре «клиент-сервер» может поддерживаться и с помощью схемы Open DataBase Connectivity (ODBC), как показано на рисунке 2.19.

Сеть образуется в этом случае путем соединения серверов. Такое соединение обеспечивается или средствами СУБД (SQL Server) или мониторами транзакций (TUXEDO).

2.5. ПРЕДСТАВЛЕНИЕ И ИСПОЛЬЗОВАНИЕ ИНФОРМАЦИИ

В условиях использования информационных технологий функции распределены между человеком и техническими устройствами. При анализе деятельности человека наибольшее значение имеют эргономические (инженерно-психологические) и психологические (социально-психологические) факторы [14].

Эргономические факторы позволяют, во-первых, определить рациональный набор функций человека, во-вторых, обеспечить рациональное сопряжение человека с техническими средствами и информационной средой.

Психологические факторы имеют большое значение, так как внедрение информационных технологий в корне изменяет деятельность человека. Наряду с положительными моментами, связанными с рационализацией деятельности, предоставлением новых возможностей, возникают и негативные явления. Это может быть вызвано различными факторами: психологическим барьером, усложнением функций, другими субъективными факторами (условиями и организацией труда, уровнем заработной платы, результативностью труда, изменением квалификации).

При работе в среде информационных технологий человек воспринимает не сам объект, а некоторую его

обобщенную информационную модель, что накладывает особые требования на совместимость пользователя с различными компонентами информационных технологий.

Важным признаком, который необходимо учитывать при разработке и внедрении информационных технологий, является отношение человека к информации. Оно может быть пассивным, когда пользователю предоставляется информация по жесткому алгоритму, и активным, когда пользователь создает необходимые ему данные.

Основной задачей операции представления информации пользователю является создание эффективного интерфейса в системе «человек — компьютер». При этом осуществляется преобразование информации в форму, удобную для восприятия пользователя.

Интерфейс должен обеспечивать:

- *наглядность* отображения информации; приближенность к естественному языку, естественным знаковым системам;
- возможность отображения *различной* — как фактографической, так и документальной информации (т. е. текста и мультимедиа содержимого);
- возможность работы с *максимально* доступным множеством источников данных *без потери гибкости*, причем с условием регулярности элементов управления; независимость от архитектуры системы и организации сетевых ресурсов.

Пользователи или *потребители информации* — это животный и растительный мир, люди и технические устройства.

При этом *конечный пользователь* (*англ.* End user) — это пользователь, не работающий непосредственно с системой, но применяющий результат ее функционирования.

Интерфейс пользователя — это совокупность правил, методов и программно-аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Пользовательский интерфейс часто понимают только как

внешний вид программы. В действительности интерфейс пользователя включает в себя все аспекты, оказывающие влияние на взаимодействие пользователя и системы.

Экран. В настоящее время сформировались следующие основные *режимы представления* и *управления информацией* на экране, которым соответствуют определенные сценарии диалога «человек — ЭВМ»: режим командной строки; режим форматированного экрана; режим меню.

Интерфейсы информационных систем условно можно разделить на три группы:

- текстовые (текст-ориентированные);
- смешанные (псевдографические);
- графические;
- веб-интерфейсы;
- объектно-ориентированные интерфейсы.

Рассмотрим ретроспективу подходов к построению пользовательского интерфейса.

В качестве примера текстовых (текст-ориентированных) интерфейсов, приведем интерфейс командной строки DOS или shell-интерпретатор UNIX. Пользователь взаимодействует с ВС с помощью клавиатуры, набирая специальные команды. Для задания различных опций служат параметры. Система как ответ на действия пользователя выдает сообщения или результат выполнения введенной команды в текстовом виде. Курсор может иметь вид мигающего прямоугольника или черточки, обозначающей место ввода. В таком режиме можно одновременно взаимодействовать лишь с одной программой. Управлять взаимодействием программ можно только с командной строки, причем проверить результат — по окончании работы.

В *смешанных* (*псевдографических*) *интерфейсах* различают понятия «оконный» и «графический интерфейсы». Первый базируется на принципе разделения реального окна монитора (или виртуального desktop'а намного большего размера, чем физический дисплей) на прямоугольные области, внутри каждой из которых определенная программа направляет свой вывод и откуда получает команды.

Термин «графический» означает использование оконного графического интерфейса (каждое окно отображает графический интерфейс); полноэкранного режима (выполняется только одна программа, осуществляющая вывод в графическом режиме). То есть: оконный не обязательно графический, а графический не всегда оконный.

Псевдографическими обозначают *интерфейсы* с графическими интерфейсными элементами, например кнопками, индикаторами процесса выполнения, реализуемыми с помощью псевдографики набора ANSI, скажем, оболочка FAR. Псевдографический интерфейс можно отнести к промежуточному между чисто командным интерфейсом и графическим.

К *графическим интерфейсам* относят все оконные графические системы Windows, оболочки для UNIX — KDE, GNOM, CDE, X-Window, Photon из ОС QNX, Aqua из MacOS X. Графическими они называются потому, что все элементы пользовательского интерфейса, как и сами данные в окнах, отображаются в графическом режиме, с помощью 256 вариантов цветов, 16-битной или 32-битной глубины цветового буфера. Это позволяет сформировать привлекательные с точки зрения пользователя окна, кнопки, пиктограммы, ползунки, индикаторы.

Понятие окна общее для всех этих систем. *Окно* — прямоугольная область экрана, куда программа выводит свои данные и откуда получает команды.

Среди существующих вариантов интерфейса в системе «человек — компьютер» можно выделить два основных типа: на основе меню («смотри и выбирай») и на основе языка команд («вспоминай и набирай»).

Интерфейсы типа меню облегчают взаимодействие пользователя с компьютером, так как не требуют предварительного изучения языка общения с системой. На каждом шаге диалога пользователю предъявляются все возможные в данный момент команды в виде наборов пунктов меню, из которого пользователь должен выбрать нужный. Такой способ общения удобен для начинающих и непрофессиональных пользователей.

Интерфейс на основе языка команд требует знания пользователем синтаксиса языка общения с компьютером. Достоинством командного языка является его гибкость и мощность.

Указанные два способа реализации интерфейса представляют собой крайние случаи, между которыми возможно существование различных промежуточных вариантов. Составные части интерфейса представлены на рисунке 2.20.

Технология представления информации должна давать дополнительные возможности для понимания данных пользователями, поэтому целесообразно использование графики, диаграмм, карт.

Пользовательский интерфейс целесообразно строить на основе концептуальной модели предметной области, которая представляется совокупностью взаимосвязанных объектов со своей структурой. Однако доступ к объектам и их экземплярам возможен только через систему окон различных типов. Ряд окон связан с конкретным объектом. В соответствии с этим предложением в сценарии работы пользователя при информационном наполнении понятий предметной области выделяем две фазы:

- выбор окон;
- работа с окнами.

Для упрощения работы окна можно группировать в соответствии с функциональными потребностями. С этой це-

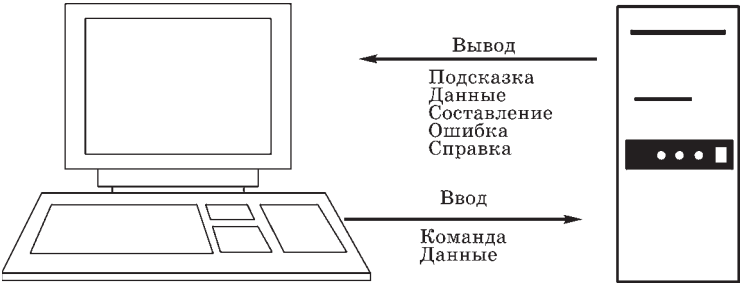


Рис. 2.20
Составные части интерфейса

лью вводится механизм разделов, который предоставляет возможность создания иерархии ориентированных функционально разделов, в каждый из которых включается необходимый набор других разделов и окон. Посредством спецификации окон для каждого из объектов возможно указать допустимые режимы работы с экземплярами и состав видимых атрибутов с режимами работы с ними. Возможно отобрать несколько разделов и несколько окон в них одновременно.

Таким образом, фаза выбора объектов должна поддерживаться следующими функциями:

- работа с общим каталогом окон в главном разделе;
- создание нового раздела;
- удаление раздела;
- редактирование описания раздела;
- передача определений и окон между разделами;
- движение по иерархии разделов;
- отбор разделов для работы;
- отбор окон для работы.

Позиции окон могут быть связаны с другими окнами через соответствующие команды из типового набора. По существу спецификация окон задает сценарий работы с экземплярами.

Окно — средство взаимосвязи пользователя с системой. Окно представляется как специальный объект. Проектирование пользовательского интерфейса представляет собой процесс спецификации окон.

Примером оконного интерфейса является интерфейс MS Windows, использующий метафору рабочего стола и включающий ряд понятий, близких к естественным (окна, кнопки, меню и т. д.).

Пользователь информационной системы большей частью вынужден использовать данные из самых разных источников: файлов, баз данных, электронных таблиц, электронной почты и т. д. При этом данные имеют самую различную форму: текст, таблицы, графику, аудио- и видеоданные и др. В связи с этим возникает проблема интеграции источников информации, заключающаяся в том, что, во-первых, пользователю должны представляться

не данные, а информация в форме, максимально удобной для восприятия, во-вторых, он должен использовать единственный универсальный интерфейс, позволяющий единообразно работать с подготовленной информацией. Пассивные пользователи, называемые иногда потребителями, обладают рядом специфических качеств, связанных с отсутствием времени, желания и квалификации для более глубокого изучения используемых инструментальных средств. В этом случае алгоритм общения с системой должен быть предельно простым. Другая часть пользователей требует предоставления достаточно широкого круга средств активного влияния на выполняемые информационные процессы.

Этим требованиям удовлетворяет веб-технология. Развитие средств вычислительной техники привело к ситуации, когда вместо традиционных параметров — производительность, пропускная способность, объем памяти — узким местом стал интерфейс с пользователем. Первым шагом на пути преодоления кризисной ситуации стала концепция гипертекста, впервые предложенная Теодором Хольмом Нельсоном. По сути гипертекст — это обычный текст, содержащий ссылки на собственные фрагменты и другие тексты (рис. 2.21).

Аналогом гипертекста можно считать книгу, оглавление которой представляет ссылки на главы, разделы, страницы. Внутри книги содержатся ссылки на другие источники. Дальнейшее развитие гипертекст получил с возникновением сети Интернет, когда появилась возможность размещать тексты на различных, территориально удаленных компьютерах. При этом требовалось дальнейшее совершенствование интерфейса, так как имеющийся не позволял представить разнообразную инфор-

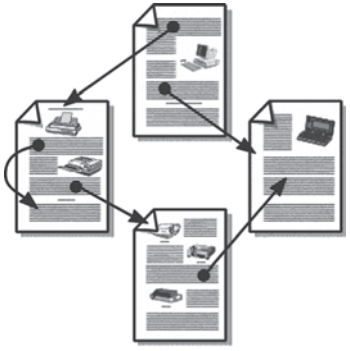


Рис. 2.21
Гипертекст

мацию разной природы, был ограничен и затруднен для восприятия, отсутствовал доступ множества потребителей к единому массиву структурированной информации. В результате была предложена и реализована концепция навигатора веб. Веб-сервер выступает в качестве информационного концентратора, получающего информацию из разных источников и в однородном виде представляющем пользователю. Средства веб обеспечивают также представление информации с нужной степенью детализации с помощью веб-навигатора. Таким образом, веб — это инфраструктурный интерфейс для пользователей различных уровней.

Несомненным преимуществом веб-технологии является удобная форма предоставления информационных услуг потребителям, определяемая как концепция публикаций информации и имеющая следующие особенности:

- информация предоставляется потребителю в виде публикаций;
- публикация может объединять информационные источники различной природы и географического расположения;
- изменения в информационных источниках мгновенно отражаются в публикациях;
- в публикациях могут содержаться ссылки на другие публикации без ограничения на местоположение и источники последних (гипертекстовые ссылки);
- потребительские качества публикаций соответствуют современным стандартам мультимедиа (доступны текст, графика, звук, видео, анимация);
- публикатор не заботится о процессе доставки информации к потребителю;
- количество потенциальных потребителей информации практически не ограничено;
- публикации отражают текущую информацию, время запаздывания определяется исключительно скоростью подготовки электронного документа;
- информация, предоставленная в публикации, легкодоступна благодаря гипертекстовым ссылкам и средствам контекстного поиска;

- информация легко усваивается потребителем благодаря широкому спектру изобразительных возможностей, предоставляемых веб-технологией;
- технология не предъявляет особых требований к типам и источникам информации;
- технология допускает масштабируемые решения: увеличение числа одновременно обслуживаемых потребителей не требует радикальной перестройки системы.

В настоящее время сформировались и на уровне стандартов утвердились следующие стили пользовательских интерфейсов.

Графический пользовательский интерфейс (Graphical User Interface — GUI) определяется как стиль взаимодействия «пользователь — компьютер», в котором применяются четыре фундаментальных элемента: окна, пиктограммы, меню и указатели. Иногда GUI-интерфейс называют WIMP-интерфейсом (Windows — окна, Icons — пиктограммы, Menus — меню и Pointers — указатели).

Пользовательский веб-интерфейс (WUI-интерфейс). Базовый WUI-стиль (Web User Interface) весьма схож с меню иерархической структурой, которое пользователи знают по опыту работы в средах с графическим интерфейсом за исключением более наглядного представления и использования гиперссылок. Необходимая навигация выполняется в рамках одного или нескольких приложений с использованием текстовых или визуальных гиперссылок. В зависимости от структуры гиперссылок приложения навигация в пределах WUI-интерфейса приводит к отображению веб-страниц в иерархии приложения — по одной за раз — в линейном или нелинейном стиле внутри одного GUI-окна. Во многих отношениях WUI-ориентированные приложения — это «шаг назад в будущее» или, может быть, нечто худшее, учитывая объемы электронных документов и других материалов в веб.

Объектно-ориентированный пользовательский интерфейс. Проектирование программных объектов дает

возможность предоставить в распоряжение пользователя приложение, обладающее объектно-ориентированным стилем и/или объектно-ориентированной внутренней структурой (реализацией). Многие объектно-ориентированные свойства реального мира находят отражение во внешнем виде, поведении, требованиях к взаимодействию и функциональных возможностях. Компьютеризованное усовершенствование или дополнение объектов реального мира, если только оно плохо спроектировано или реализовано, не очевидно для конечного пользователя и не в состоянии преодолеть его устоявшиеся знания и восприятие. Представленные в явном виде при проектировании обозначения классов объектов, иерархии классов и наследование посредством иерархии классов остаются прозрачными для пользователя.

Объектно-ориентированный прикладной интерфейс должен обладать следующими свойствами:

- обеспечивать непосредственное манипулирование (перетаскивать любые объекты куда угодно);
- обеспечивать непосредственный ввод данных (записывать любую информацию);
- обеспечивать контекстную зависимость от объектов (всплывающие (контекстные) меню, справки, согласованность и т. д.).

Хороший прикладной объектно-ориентированный интерфейс прост в использовании; это значит, что механизмы его пользовательского интерфейса прозрачны.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Укажите основные фазы извлечения информации.
2. Объясните суть декомпозиции на основе объектно-ориентированного подхода.
3. Что такое инкапсуляции, полиформизм и наследование?
4. Какие существуют методы обогащения информации?
5. Раскройте содержание технологии Data Mining.
6. В чем особенности технологии Text Mining?

7. Охарактеризуйте методы поиска информации в сети Интернет на основе информационно-поисковых систем.
8. Поясните процесс формирования информационных ресурсов.
9. Что такое поисковый образ документа?
10. Какие существуют методы индексирования данных?
11. Укажите особенности применения аппарата нейронных сетей и онтологий в поисковых механизмах.
12. Что представляет собой модель OSI?
13. Какие существуют протоколы сетевого взаимодействия?
14. Что такое драйвер?
15. Что такое дейтаграммный протокол?
16. Укажите функции, выполняемые протоколами канального уровня.
17. Какие функции выполняют протоколы среднего уровня?
18. Какие функции выполняют протоколы верхнего уровня?
19. Охарактеризуйте основное требование к компьютерной сети.
20. Перечислите основные характеристики качества обслуживания компьютерной сети.
21. Как оценивается производительность компьютерной сети?
22. Как оценивается надежность и безопасность компьютерной сети?
23. Как реализуются основные виды качества обслуживания компьютерной сети?
24. Укажите основные особенности сетей SDH DWDM.
25. Поясните содержание числовой и нечисловой обработки информации.
26. Охарактеризуйте виды обработки информации.
27. Какие существуют архитектуры ЭВМ с точки зрения обработки информации?
28. Определите содержание основных процедур обработки данных.
29. Поясните особенности принятия решений в различных условиях.
30. Укажите основные компоненты поддержки принятия решений.

31. Какие существуют системы поддержки принятия решений?
32. Для решения каких задач предназначены СППР?
33. Какие основные функции реализует СППР?
34. Чем отличаются оперативные СППР от стратегических?
35. Укажите особенности систем поддержки принятия решений, основанных на базе знаний.
36. Какую роль играет интеллектуальный анализ данных в СППР?
37. Укажите основные разновидности процессов ИАД.
38. Что является теоретической базой ИАД?
39. Укажите базовые принципы построения OLAP-систем.
40. В чем отличие MOLAP- и ROLAP-систем?
41. Как реализуются информационные приложения в процессе принятия решений?
42. Назовите отличительные признаки концепции баз данных.
43. В чем отличие хранилища данных от базы данных?
44. Какие модели используются для описания предметной области?
45. Какие модели используются на концептуальном, логическом и физическом уровнях?
46. Дайте краткую характеристику основных типов баз данных.
47. Сформулируйте подходы к проектированию баз данных.
48. Что такое СУБД, какие существуют ее стандарты и способы реализации?
49. Опишите содержание процесса проектирования баз данных.
50. Какие существуют критерии оценки баз данных?
51. В чем суть объектно-ориентированной БД?
52. Перечислите основные положения объектно-ориентированных баз данных.
53. Какие существуют разновидности объектно-реляционных баз данных?
54. Каковы преимущества использования распределенных баз данных?

55. Каковы отличительные черты архитектуры «клиент-сервер»?
56. Что такое интерфейс и какова его роль в процессе представления и использования информации?
57. Какие существуют виды интерфейсов?
58. На чем основана концепция гипертекста?
59. В чем заключается концепция публикаций информации?

ГЛАВА 3

БАЗОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

В результате освоения данной темы студент должен:

1) знать:

- основные принципы, свойства, модели и средства реализации мультимедиатехнологий;
- основные принципы и методы организации данных в геоинформационных системах;
- виды информационных угроз, цели, способы и средства защиты информации;
- методологию проектирования информационных систем на основе CASE-технологии, задачи, решаемые средствами CASE-технологии, состав и структуру средств CASE-технологии;
- виды распределенных архитектур информационных систем, модели, методы, структуру и состав телекоммуникационных технологий;
- основные понятия искусственного интеллекта, виды и структуру интеллектуальной системы, модели представления знаний;
- методы и средства разработки программного обеспечения, основные этапы разработки программного продукта, технологии создания программного продукта;
- основные характеристики облачных сервисов, модели развертывания инфраструктуры, примеры реализации облачных технологий;
- характеристики и методы анализа больших данных, средства реализации технологии больших данных;

2) уметь:
- применять базовые информационные технологии при проектировании информационных систем;
- использовать архитектурные и детализированные решения базовых информационных технологий при проектировании систем;

3) владеть:
- методологией использования базовых информационных технологий при создании информационных систем;
- моделями и средствами разработки архитектуры информационных систем на основе базовых информационных технологий;
- навыками владения одной из технологий программирования.

3.1. МУЛЬТИМЕДИАТЕХНОЛОГИИ

Мультимедиа (*англ.* multimedia от *лат.* multum — много и media, medium — средоточие, средства) — это комплекс аппаратных и программных средств, позволяющих пользователю работать в диалоговом режиме с разнородными данными (графикой, текстом, звуком, видео и анимацией), организованными в виде единой информационной среды.

Таким образом, мультимедиа объединяет несколько типов разнородных данных (текст, звук, видео, графическое изображение и анимацию) в единое целое. И это понятие само по себе имеет три лица:
- мультимедиа — как идея, т. е. новый подход к хранению информации различного типа в единой цифровой форме;
- мультимедиа — как оборудование для обработки и хранения информации, без него мультимедиаидею реализовать невозможно;
- программное обеспечение, позволяющее объединить четыре элемента информации в законченное мультимедиаприложение.

Одним из важных факторов мультимедиа является интерактивность — свойство реагировать на действия пользователей, в том числе и управлять пользователем.

В настоящее время мультимедиа является активно развивающейся областью информационных технологий. В этом направлении активно работает значительное число крупных и мелких фирм, технических университетов и студий (в частности, IBM, Apple, Motorola, Philips, Sony, Intel и др.). Области использования чрезвычайно многообразны: интерактивные обучающие и информационные системы, САПР, развлечение и др.

Три основных принципа мультимедиа:

- представление информации с помощью комбинации множества воспринимаемых человеком сред;
- наличие нескольких сюжетных линий в содержании продукта (в том числе и выстраиваемых самим пользователем на основе «свободного поиска» в рамках предложенной в содержании продукта информации);
- художественный дизайн интерфейса и средств навигации.

Основными характерными особенностями этих технологий являются:

- объединение многокомпонентной информационной среды (текста, звука, графики, фото, видео) в однородном цифровом представлении;
- обеспечение надежного (отсутствие искажений при копировании) и долговечного хранения (гарантийный срок хранения — десятки лет) больших объемов информации;
- простота переработки информации от рутинных до творческих операций.

Несомненным достоинством и особенностью технологии являются следующие возможности мультимедиа, которые активно используются в представлении информации:

- хранение большого объема самой разной информации на одном носителе;
- увеличение (детализация) на экране изображения или его наиболее интересных фрагментов, иногда в двадцатикратном увеличении (режим «лупа»), при сохранении качества изображения. Это особенно важно для презентации произведений искусства и уникальных исторических документов;

- сравнение изображения и обработки его разнообразными программными средствами с научно-исследовательскими или познавательными целями;
- выделение в сопровождающем текстовом или другом визуальном материале «горячих слов (областей)», по которым осуществляется немедленное получение справочной или любой другой пояснительной (в том числе визуальной) информации (технологии гипертекста и гипермедиа);
- осуществление непрерывного музыкального или любого другого аудиосопровождения, соответствующего статичному или динамичному визуальному ряду;
- использование видеофрагментов из фильмов, видеозаписей и т. д., функции стоп-кадра, покадрового «пролистывания» видеозаписи;
- включение в содержание диска баз данных, методик обработки образов, анимации (к примеру, сопровождение рассказа о композиции картины графической анимационной демонстрацией геометрических построений ее композиции) и т. д.;
- подключение к глобальной сети Интернет;
- работа с различными приложениями (текстовыми, графическими и звуковыми редакторами, картографической информацией);
- автоматический просмотр всего содержания продукта (слайд-шоу) или создания анимированного и озвученного «путеводителя-гида» по продукту («говорящей и показывающей инструкции пользователя»), включение в состав продукта игровых компонентов с информационными составляющими;
- «свободная» навигация по информации и выход в основное меню (укрупненное содержание), на полное оглавление или вовсе из программы в любой точке продукта.

Таким образом, мультимедийный продукт — наиболее эффективная форма подачи информации в среде компьютерных информационных технологий. Он позволяет собрать воедино огромные и разрозненные объемы информации, дает возможность с помощью интерактив-

ного взаимодействия выбирать интересующие в данный момент информационные блоки, значительно повышая эффективность восприятия информации.

Достигнутый технологический базис основан на использовании нового стандарта оптического носителя DVD (Digital Versalite/Video Disk), имеющего емкость порядка единиц и десятков гигабайт и заменяющего все предыдущие: CD-ROM, Video CD, CDaudio. Использование DVD позволило реализовать концепцию однородности цифровой информации. Одно устройство заменяет аудиоплейер, видеомагнитофон, CD-ROM дисковод, слайдер и др. В плане представления информации оптический носитель DVD приближает ее к уровню виртуальной реальности.

Многокомпонентную мультимедиасреду целесообразно разделить на три группы: аудиоряд, видеоряд, текстовая информация.

Аудиоряд может включать речь, музыку, эффекты (звуки типа шум, гром, скрип и т. д., объединяемые обозначением WAVE (волна) [42]. Главной проблемой при использовании этой группы мультисреды является информационная емкость. Для записи одной минуты WAVE звука высшего качества необходима память порядка 10 Мбайт, поэтому стандартный объем CD (до 640 Мбайт) позволяет записать не более часа WAVE. Для решения этой проблемы используются методы компрессии звуковой информации.

Другим направлением является использование в мультисреде звуков (одноголосая и многоголосая музыка, вплоть до оркестра, звуковые эффекты) MIDI (Musical Instrument Digitale Interface). В данном случае звуки музыкальных инструментов, звуковые эффекты синтезируются программно-управляемыми электронными синтезаторами. Коррекция и цифровая запись MIDI-звуков осуществляется с помощью музыкальных редакторов (программ-секвенсоров). Главным преимуществом MIDI является малый объем требуемой памяти — 1 минута MIDI-звука занимает в среднем 10 Кбайт.

Видеоряд характеризуется по сравнению с аудиорядом большим количеством элементов. Выделяют статический и динамический видеоряды.

Статический видеоряд включает графику (рисунки, интерьеры, поверхности, символы в графическом режиме) и фото (фотографии и сканированные изображения).

Динамический видеоряд представляет последовательность статических элементов (кадров). Можно выделить три типовых группы:

- обычное видео (life video) — последовательность фотографий (около 24 кадров в секунду);
- квазивидео — разреженная последовательность фотографий (6–12 кадров в секунду);
- анимация — последовательность рисованных изображений.

Первая проблема при реализации видеорядов — разрешающая способность экрана и количество цветов. Выделяют три направления:

- стандарт VGA дает разрешение 640×480 пикселей (точек) на экране при 16 цветах или 320×200 пикселей при 256 цветах;
- стандарт SVGA (видеопамять 512 К, 8 бит/пиксель) дает разрешение 640×480 при 256 цветах;
- 24-битные видеоадаптеры (видеопамять 2 Мбайт, 24 бит/пиксель) позволяют использовать 16 млн цветов.

Вторая проблема — объем памяти. Для статических изображений один полный экран требует следующих объемов памяти:

- в режиме 640×480, 16 цветов — 150 Кбайт;
- в режиме 320×200, 256 цветов — 62,5 Кбайт;
- в режиме 640×480, 256 цветов — 300 Кбайт.

Такие значительные объемы при реализации аудио- и видеорядов определяют высокие требования к носителю информации, видеопамяти и скорости передачи информации.

При размещении текстовой информации на CD-ROM нет никаких сложностей и ограничений ввиду большого информационного объема оптического диска.

Области применения мультимедиаприложений:

1) обучение с использованием компьютерных технологий (научно-просветительская или образовательная сфера);

2) видеоэнциклопедии, интерактивные путеводители, тренажеры, ситуационно-ролевые игры и др.;

3) информационная и рекламная служба;

4) популяризаторская и развлекательная сферы;

5) интернет-вещание. Спектр возможных применений — от просмотра заказной телепередачи, выбора нужной книги до участия в мультимедиаконференциях. Такие разработки получили название Information Highway;

6) развлечения, игры, системы виртуальной реальности;

7) презентационная (витринная реклама), СМИ;

8) творчество;

9) военные технологии;

10) промышленность и техника (сенсорные экраны);

11) торговля;

12) мультимедийные информационные системы («мультимедиакиоски»), выдающие по запросу пользователя наглядную информацию.

13) научные исследования — электронные архивы и библиотеки;

14) медицина: базы знаний, методики операций, каталоги лекарств и т. п.;

15) искусственный интеллект — внедрение элементов искусственного интеллекта в системе мультимедиа. Они обладают способностью «чувствовать» среду общения, адаптироваться к ней и оптимизировать процесс общения с пользователем: подстраиваются под читателей, анализируют круг их интересов, помнят вопросы, вызывающие затруднения, и могут сами предложить дополнительную или разъясняющую информацию;

16) системы распознавания речи, понимающие естественный язык, еще больше расширяют диапазон взаимодействия с компьютером.

Технологию мультимедиа составляют специальные аппаратные и программные средства.

Для построения мультимедиасистемы необходима дополнительная аппаратная поддержка: аналого-цифровые и цифроаналоговые преобразователи для перевода ана-

логовых аудио- и видеосигналов в цифровой эквивалент и обратно, видеопроцессоры для преобразования обычных телевизионных сигналов к виду, воспроизводимому электронно-лучевой трубкой дисплея, декодеры для взаимного преобразования телевизионных стандартов, специальные интегральные схемы для сжатия данных в файлы допустимых размеров и т. д.

Аппаратные средства мультимедиа:

- средства звукозаписи (звуковые платы, микрофоны);
- средства звуковоспроизведения (усилитель, колонки, акустические системы, наушники и гарнитуры);
- манипуляторы (компьютерные мыши, джойстики, миди-клавиатуры);
- средства «виртуальной реальности» (перчатки, очки, шлемы виртуальной реальности, используемые в играх);
- носители информации (CD, DVD и HDD);
- средства передачи (мини-видеокамеры, цифровые фотоаппараты);
- средства записи (приводы CD/DVD-ROM, CDRW/DVD+RW, TV- и FM-тюнеры);
- средства обработки изображения (платы видеомонтажа, клавиатуры, графические акселераторы);
- компьютер, телевизор, средства для получения и удобного восприятия информации и др.

С точки зрения технических средств на рынке представлены как полностью укомплектованные мультимедиакомпьютеры, так и отдельные комплектующие и подсистемы (Multimedia Upgrade Kit), включающие в себя звуковые карты, приводы компакт-дисков, джойстики, микрофоны, акустические системы.

Программные средства мультимедиа складываются из трех компонентов:

1) системные программные средства;

2) инструментальные программные средства;

3) прикладные программные средства.

Для персональных компьютеров класса IBM PC утвержден специальный стандарт MPC, определяющий минимальную конфигурацию аппаратных средств для

воспроизведения мультимедиапродуктов. Для оптических дисков CD-ROM разработан международный стандарт (ISO 9660).

3.2. ГЕОИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Геоинформационные технологии можно определить как совокупность программно-технологических, методических средств получения новых видов информации об окружающем мире. Геоинформационные технологии предназначены для повышения эффективности процессов управления, хранения и представления информации, обработки и поддержки принятия решений. Это заключается во внедрении геоинформационных технологий в науку, производство, образование и применение в практической деятельности получаемой информации об окружающей реальности. Геоинформационные технологии являются новыми информационными технологиями, направленными на достижение различных целей, включая информатизацию производственно-управленческих процессов. Особенностью геоинформационных систем (ГИС) является то, что как информационные системы они являются результатом эволюции этих систем и поэтому включают в себя основы построения и функционирования информационных систем. ГИС как система включает множество взаимосвязанных элементов, каждый из которых связан прямо или косвенно с каждым другим элементом, а два любые подмножества этого множества не могут быть независимыми, не нарушая целостность, единство системы. Еще одной особенностью ГИС является то, что она является интегрированной информационной системой. Интегрированные системы построены на принципах интеграции технологий различных систем. Они зачастую применяются настолько в разных областях, что их название часто не определяет все их возможности и функции. По этой причине не следует связывать ГИС с решением задач только геодезии или географии. «Гео» в названии геоинформационных систем и технологий определяет объект исследований, а не предметную область использования этих систем.

Интеграция ГИС с другими информацинными системами порождает их многоаспектность. В ГИС осуществляется комплексная обработка информации от сбора данных до ее хранения, обновления и представления, поэтому следует рассмотреть ГИС с различных позиций.

Как *системы управления* ГИС предназначены для обеспечения процесса принятия решений по оптимальному управлению землями и ресурсами, городским хозяйством, организации транспорта и розничной торговли, использованию океанов или других пространственных объектов. В отличие от информационных систем, в ГИС появляется множество новых технологий пространственного анализа данных, объединенных с технологиями электронного офиса и оптимизации решений на этой основе. В силу этого ГИС является эффективным местом преобразования и синтеза разнообразных данных для задач управления.

Как *геосистемы*, ГИС интегрируют технологии сбора информации таких систем, как географические информационные системы, системы картографической информации, автоматизированные системы картографирования, автоматизированные фотограмметрические системы, земельные информационные системы, автоматизированные кадастровые системы и т. п.

Как *системы баз данных*, ГИС характерны широким набором данных, собираемых с помощью разных методов и технологий. При этом следует подчеркнуть, что они объединяют возможности текстовых и графических баз данных.

Как *системы моделирования*, ГИС использует максимальное количество методов и процессов моделирования, применяемых в других информационных системах, и в первую очередь в САПР.

Как *системы получения проектных решений*, ГИС во многом используют концепции и методы автоматизированного проектирования и решают ряд специальных проектных задач, которые в типовом автоматизированном проектировании не встречаются.

Как *системы представления информации*, ГИС являются развитием автоматизированных систем документа-

ционного обеспечения с использованием современных технологий мультимедиа. Они обладают средствами деловой графики и статистического анализа и дополнительно к этому средствами тематического картографирования. Именно эффективность последнего обеспечивает разнообразное решение задач в разных отраслях при использовании интеграции данных на основе картографической информации.

Как *прикладные системы*, ГИС не имеют себе равных по широте, так как применяются в транспорте, навигации, геологии, географии, военном деле, топографии, экономике, экологии и т. д.

Как *системы массового использования*, ГИС позволяют использовать картографическую информацию на уровне деловой графики, что делает их доступными любому школьнику или бизнесмену, а не только специалисту-географу. Именно поэтому принятие многих решений на основе ГИС-технологий не сводится к созданию карт, а лишь использует картографические данные.

Организация данных в ГИС

Тематические данные хранятся в ГИС в виде таблиц, поэтому проблем с их хранением и организацией в базах данных не возникает. Наибольшие проблемы вызывают хранение и визуализация графических данных.

Основным классом данных геоинформационных систем (ГИС) являются координатные данные, содержащие геометрическую информацию и отражающие пространственный аспект. Основные типы координатных данных: точка (узлы, вершины), линия (незамкнутая), контур (замкнутая линия), полигон (ареал, район). На практике для построения реальных объектов исполь-

Рис. 3.1
Основные элементы координатных данных

зуют большее число данных (например, висячий узел, псевдоузел, нормальный узел, покрытие, слой и др.). На рисунке 3.1 показаны основные из рассмотренных элементов координатных данных [15].

Рассмотренные типы данных имеют большее количество разнообразных связей, которые можно условно разделить на три группы:

- взаимосвязи для построения сложных объектов из простых элементов;
- взаимосвязи, вычисляемые по координатам объектов;
- взаимосвязи, определяемые с помощью специального описания и семантики при вводе данных.

В общем случае модели пространственных (координатных) данных могут иметь векторное или растровое (ячеистое) представление, содержать или не содержать топологические характеристики. Этот подход позволяет классифицировать модели по трем типам: растровая модель; векторная нетопологическая модель; векторная топологическая модель. Все эти модели взаимно преобразуемы. Тем не менее при получении каждой из них необходимо учитывать их особенности. В ГИС форме представления координатных данных соответствуют два основных подкласса моделей — векторные и растровые (ячеистые или мозаичные). Возможен класс моделей, которые содержат характеристики как векторов, так и мозаик. Они называются гибридными моделями.

Графическое представление какой-либо ситуации на экране компьютера подразумевает отображение различных графических образов. Сформированный графический образ на экране ЭВМ состоит из двух различных с точки зрения среды хранения частей — графической «подложки» или графического фона и других графических объектов. По отношению к этим другим графическим образам «образ-подложка» является «площадным», или пространственным двухмерным, изображением. Основной проблемой при реализации геоинформационных приложений является трудность формализованного описания конкретной предметной области и ее отображения на электронной карте.

Таким образом, геоинформационные технологии предназначены для широкого внедрения в практику методов и средств информационного взаимодействия над пространственно-временными данными, представляемыми в виде системы электронных карт, и предметно-ориентированных сред обработки разнородной информации для различных категорий пользователей.

Рассмотрим более подробно основные графические модели.

Векторные модели широко применяются в ГИС. Они строятся на векторах, занимающих часть пространства, в отличие от занимающих все пространство растровых моделей. Это определяет их основное преимущество — требование на порядки меньшей памяти для хранения и меньших затрат времени на обработку и представление, а главное, более высокая точность позиционирования и представления данных. При построении векторных моделей объекты создаются путем соединения точек прямыми линиями, дугами окружностей, полилиниями. Площадные объекты-ареалы задаются наборами линий. Векторные модели используются преимущественно в транспортных, коммунальных, маркетинговых приложениях ГИС. Системы ГИС, работающие в основном с векторными моделями, получили название векторных ГИС. В реальных ГИС имеют дело не с абстрактными линиями и точками, а с объектами, содержащими линии и ареалы, занимающими пространственное положение, а также со сложными взаимосвязями между ними. Поэтому полная векторная модель данных ГИС отображает пространственные данные как совокупность следующих основных частей: геометрические (метрические) объекты (точки, линии и полигоны); атрибуты — признаки, связанные с объектами; связи между объектами. Векторные модели (объектов) используют в качестве элементарной модели последовательность координат, образующих линию. Линией называют границу, сегмент, цепь или дугу. Основные типы координатных данных в классе векторных моделей определяются через базовый элемент «линия» следующим образом. Точка определяется как вы-

родившаяся линия нулевой длины, линия — как линия конечной длины, а площадь представляется последовательностью связанных между собой отрезков. Каждый участок линии может являться границей для двух ареалов либо двух пересечений (узлов). Отрезок общей границы между двумя пересечениями (узлами) имеет разные названия, которые являются синонимами в предметной области ГИС. Специалисты по теории графов предпочитают слову «линия» термин «ребро», а для обозначения пересечения употребляют термин «вершина». Национальным стандартом США официально санкционирован термин «цепь». В некоторых системах (Arclnfo, GeoDraw) используется термин «дуга». В отличие от обычных векторов в геометрии, дуги имеют свои атрибуты. Атрибуты дуг обозначают полигоны по обе стороны от них. По отношению к последовательному кодированию дуги эти полигоны именуются левый и правый. Понятие дуги (цепи, ребра) является фундаментальным для векторных ГИС. Векторные модели получают разными способами. Один из наиболее распространенных — векторизация сканированных (растровых) изображений. Векторизация — процедура выделения векторных объектов с растрового изображения и получение их в векторном формате. Для векторизации необходимо высокое качество (отчетливые линии и контуры) растровых образов. Чтобы обеспечить требуемую четкость линий, иногда приходится заниматься улучшением качества изображения.

При векторизации возможны ошибки, исправление которых осуществляется в два этапа:

1) корректировка растрового изображения до его векторизации;

2) корректировка векторных объектов.

Векторные модели с помощью дискретных наборов данных отображают непрерывные объекты или явления. Следовательно, можно говорить о векторной дискретизации. При этом векторное представление позволяет отразить бóльшую пространственную изменчивость для одних районов, чем для других, по сравнению с растровым представлением, что обусловлено более четким показом

границ и их меньшей зависимостью от исходного образа (изображения), чем при растровом отображении. Это типично для социальных, экономических, демографических явлений, изменчивость которых в ряде районов более интенсивна. Некоторые объекты являются векторными по определению, например границы соответствующего земельного участка, границы районов и т. д. Поэтому векторные модели обычно используют для сбора данных координатной геометрии (топографические записи), данных об административно-правовых границах и т. п.

Особенности векторных моделей. В векторных форматах набор данных определен объектами базы данных. Векторная модель может организовывать пространство в любой последовательности и дает «произвольный доступ» к данным. В векторной форме легче осуществляются операции с линейными и точечными объектами, например анализ сети — разработка маршрутов движения по сети дорог, замена условных обозначений. В растровых форматах точечный объект должен занимать целую ячейку. Это создает ряд трудностей, связанных с соотношением размеров растра и размера объекта. Что касается точности векторных данных, то здесь можно говорить о преимуществе векторных моделей перед растровыми, так, векторные данные могут кодироваться с любой мыслимой степенью точности, которая ограничивается лишь возможностями метода внутреннего представления координат. Обычно для представления векторных данных используется 8 или 16 десятичных знаков (одинарная или двойная точность). Только некоторые классы данных, получаемых в процессе измерений, соответствуют точности векторных данных. Это данные, полученные точной съемкой (координатная геометрия); карты небольших участков, составленные по топографическим координатам и политические границы, определенные точной съемкой. Не все природные явления имеют характерные четкие границы, которые можно представить в виде математически определенных линий. Это обусловлено динамикой явлений или способами сбора пространственной информации. Почвы, типы растительности, склоны, места обитания диких животных — все

эти объекты не имеют четких границ. Обычно линии на карте имеют толщину 0,4 мм и, как часто считается, отражают неопределенность положения объекта. В растровой системе эта неопределенность задается размером ячейки. Поэтому следует помнить, что в ГИС действительное представление о точности дают размер растровой ячейки и неопределенность положения векторного объекта, а не точность координат. Для анализа связей в векторных моделях необходимо рассмотреть их топологические свойства, т. е. рассмотреть топологические модели, которые являются разновидностью векторных моделей данных.

В **растровых моделях** дискретизация осуществляется наиболее простым способом — весь объект (исследуемая территория) отображается в пространственные ячейки, образующие регулярную сеть. Каждой ячейке растровой модели соответствует одинаковый по размерам, но разный по характеристикам (цвет, плотность) участок поверхности. Ячейка модели характеризуется одним значением, являющимся средней характеристикой участка поверхности. Эта процедура называется пикселизацией. Растровые модели делятся на регулярные, нерегулярные и вложенные (рекурсивные или иерархические) мозаики. Плоские регулярные мозаики бывают трех типов: квадрат (рис. 3.2), треугольник и шестиугольник (рис. 3.3).

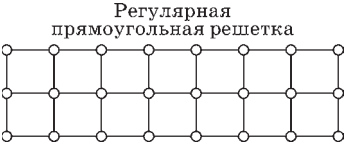


Рис. 3.2
Мозаика-квадрат

Рис. 3.3
Мозаика-треугольник

Квадратная форма удобна при обработке больших объемов информации, треугольная — для создания сферических поверхностей. В качестве нерегулярных мозаик используют треугольные сети неправильной формы (Triangulated Irregular Network — TIN) и полигоны Тиссена (рис. 3.4). Они удобны для создания цифровых моделей отметок местности по заданному набору точек.

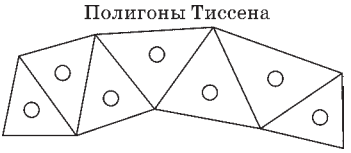


Рис. 3.4
Полигоны Тиссена

Таким образом, векторная модель содержит информацию о местоположении объекта, а растровая о том, что расположено в той или иной точке объекта. Векторные модели относятся к бинарным или квазибинарным. Растровые позволяют отображать полутона. Основной областью использования растровых моделей является обработка аэрокосмических снимков.

Если векторная модель дает информацию о том, где расположен тот или иной объект, то растровая — информацию о том, что расположено в той или иной точке территории. Это определяет основное назначение растровых моделей — непрерывное отображение поверхности. В растровых моделях в качестве атомарной модели используют двухмерный элемент пространства — пиксель (ячейка). Упорядоченная совокупность атомарных моделей образует растр, который, в свою очередь, является моделью карты или геообъекта. Векторные модели относятся к бинарным или квазибинарным. Растровые позволяют отображать полутона и цветовые оттенки. Как правило, каждый элемент растра или каждая ячейка должны иметь лишь одно значение плотности или цвета. Это применимо не для всех случаев. Например, когда граница двух типов покрытий может проходить через центр элемента растра, элементу дается значение, характеризующее бóльшую часть ячейки или ее центральную точку. Ряд систем позволяет иметь несколько значений для одного элемента растра. Для растровых моделей существует ряд характеристик: разрешение, ориентация, зоны, значение, положение. Разрешение — минимальный линейный размер наименьшего участка отображаемого пространства (поверхности), отображаемый одним пикселем. Пиксели обычно представляют собой прямоугольники или квадраты, реже используются треугольники и шестиугольники. Более высоким разрешением обладает растр с меньшим размером ячеек. Высокое разрешение подразумевает

обилие деталей, множество ячеек, минимальный размер ячеек. Значение — элемент информации, хранящийся в элементе растра (пикселе). Поскольку при обработке применяют типизированные данные, т. е. необходимость определить типы значений растровой модели. Тип значений в ячейках растра определяется как реальным явлением, так и особенностями ГИС. В частности, в разных системах можно использовать разные классы значений: целые числа, действительные (десятичные) значения, буквенные значения. Целые числа могут служить характеристиками оптической плотности или кодами, указывающими на позицию в прилагаемой таблице или легенде. Например, возможна следующая легенда, указывающая наименование класса почв: 0 — пустой класс; 1 — суглинистые; 2 — песчаные; 3 — щебнистые и т. п. Ориентация — угол между направлением на север и положением колонок растра. Зона растровой модели включает соседствующие друг с другом ячейки, имеющие одинаковое значение. Зоной могут быть отдельные объекты, природные явления, ареалы типов почв, элементы гидрографии и т. п. Для указания всех зон с одним и тем же значением используют понятие «класс зон». Естественно, что не во всех слоях изображения могут присутствовать зоны. Основные характеристики зоны — ее значение и положение. Буферная зона — зона, границы которой удалены на известное расстояние от любого объекта на карте. Буферные зоны различной ширины могут быть созданы вокруг выбранных объектов на базе таблиц сопряженных характеристик. Положение обычно задается упорядоченной парой координат (номер строки и номер столбца), которые однозначно определяют положение каждого элемента отображаемого пространства в растре. Проводя сравнение векторных и растровых моделей, отметим удобство векторных для организации и работы со взаимосвязями объектов. Тем не менее, используя простые приемы, например включая взаимосвязи в таблицы атрибутов, можно организовать взаимосвязи и в растровых системах. Необходимо остановиться на вопросах точности отображения в растровых моделях. В растровых форматах

в большинстве случаев неясно, относятся ли координаты к центральной точке пикселя или к одному из его углов. Поэтому точность привязки элемента растра определяют как 1/2 ширины и высоты ячейки.

Растровые модели имеют следующие достоинства:

1) растр не требует предварительного знакомства с явлениями, данные собираются с равномерно расположенной сети точек, что позволяет в дальнейшем на основе статистических методов обработки получать объективные характеристики исследуемых объектов. Благодаря этому растровые модели могут использоваться для изучения новых явлений, о которых не накоплен материал. В силу простоты этот способ получил наибольшее распространение;

2) растровые данные проще для обработки по параллельным алгоритмам и этим обеспечивают более высокое быстродействие по сравнению с векторными;

3) некоторые задачи, например создание буферной зоны, много проще решать в растровом виде;

4) многие растровые модели позволяют вводить векторные данные, в то время как обратная процедура весьма затруднительна для векторных моделей;

5) процессы растеризации много проще алгоритмически, чем процессы векторизации, которые зачастую требуют экспертных решений.

Наиболее часто растровые модели применяют при обработке аэрокосмических снимков для получения данных дистанционных исследований Земли.

Основой визуального представления данных при использовании ГИС-технологий является графическая среда, основу которой составляют векторные и растровые (ячеистые) модели.

Векторные модели основаны на представлении геометрической информации с помощью векторов, занимающих часть пространства, что требует при реализации меньшего объема памяти. Используются векторные модели в транспортных, коммунальных, маркетинговых приложениях ГИС.

Цифровая карта может быть организована в виде множества слоев (покрытий или карт подложек). Слои в ГИС

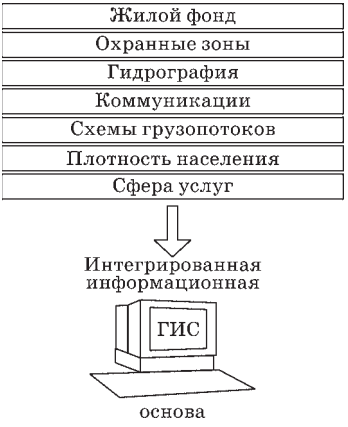


Рис. 3.5
Пример слоев интегрированной ГИС

представляют набор цифровых картографических моделей, построенных на основе объединения (типизации) пространственных объектов, имеющих общие функциональные признаки. Совокупность слоев образует интегрированную основу графической части ГИС. Пример слоев интегрированной ГИС представлен на рисунке 3.5.

Важным моментом при проектировании ГИС является размерность модели. Применяют двухмерные (2D) и трехмерные (3D) модели координат. Двухмерные модели используются при построении карт, а трехмерные — при моделировании геологических процессов, проектировании инженерных сооружений (плотин, водохранилищ, карьеров и др.), моделировании потоков газов и жидкостей. Существуют два типа трехмерных моделей:

- псевдотрехмерные, когда фиксируется третья координата;
- истинное трехмерное представление.

Большинство современных ГИС осуществляют комплексную обработку информации:

- сбор первичных данных;
- накопление и хранение информации;
- различные виды моделирования (семантическое, имитационное, геометрическое, эвристическое);
- автоматизированное проектирование;
- документационное обеспечение.

Множество задач, возникающих в жизни, привело к созданию различных ГИС, которые могут классифицироваться по следующим признакам.

По функциональным возможностям:

- полнофункциональные ГИС общего назначения;

- специализированные ГИС ориентированы на решение конкретной задачи в какой-либо предметной области;
- информационно-справочные системы для домашнего и информационно-справочного пользования.

Функциональные возможности ГИС определяются также архитектурным принципом их построения:

- закрытые системы — не имеют возможностей расширения, они способны выполнять только тот набор функций, который однозначно определен на момент покупки;
- открытые системы отличаются легкостью приспособления, возможностями расширения, так как могут быть достроены самим пользователем при помощи специального аппарата (встроенных языков программирования).

По пространственному (*территориальному*) *охвату*:

- глобальные (планетарные);
- общенациональные;
- региональные;
- локальные (в том числе муниципальные).

По проблемно-тематической ориентации:

- общегеографические;
- экологические и природопользовательские;
- отраслевые (водных ресурсов, лесопользования, геологические, туризма и т. д.).

По способу организации географических данных:

- векторные;
- растровые;
- векторно-растровые ГИС.

В качестве *источников данных* для формирования ГИС служат:

1) *картографические материалы* (топографические и общегеографические карты, карты административно-территориального деления, кадастровые планы и др.). Сведения, получаемые с карт, имеют территориальную привязку, поэтому их удобно использовать в качестве базового слоя ГИС. Если нет цифровых карт на исследуемую территорию, тогда графические оригиналы карт преобразуются в цифровой вид;

2) *данные дистанционного зондирования* (ДДЗ) все шире используются для формирования баз данных ГИС. К ДДЗ прежде всего относят материалы, получаемые с космических носителей. Для дистанционного зондирования применяют разнообразные технологии получения изображений и передачи их на Землю, носители съемочной аппаратуры (космические аппараты и спутники) размещают на разных орбитах, оснащают разной аппаратурой. Благодаря этому получают снимки, отличающиеся разным уровнем обзорности и детальности отображения объектов природной среды в разных диапазонах спектра (видимый и ближний инфракрасный, тепловой инфракрасный и радиодиапазон). Все это обусловливает широкий спектр экологических задач, решаемых с применением ДДЗ. К методам дистанционного зондирования относятся и аэро- и наземные съемки и другие неконтактные методы, например гидроакустические съемки рельефа морского дна. Материалы таких съемок обеспечивают получение как количественной, так и качественной информации о различных объектах природной среды;

3) *материалы полевых изысканий территорий* включают данные топографических, инженерно-геодезических изысканий, кадастровой съемки, геодезические измерения природных объектов, выполняемые нивелирами, теодолитами, электронными тахеометрами, GPS-приемниками, а также результаты обследования территорий с применением геоботанических и других методов, например исследования по перемещению животных, анализ почв и др.;

4) *статистические данные* содержат данные государственных статистических служб по самым разным отраслям народного хозяйства, а также данные стационарных измерительных постов наблюдений (гидрологические и метеорологические данные, сведения о загрязнении окружающей среды и т. д));

5) *литературные данные* (справочные издания, книги, монографии и статьи, содержащие разнообразные сведения по отдельным типам географических объектов).

В ГИС редко используется только один вид данных, чаще всего это сочетание разнообразных данных на какую-либо территорию.

Основные области использования ГИС:

- электронные карты;
- городское хозяйство;
- государственный земельный кадастр;
- экология;
- дистанционное зондирование;
- экономика;
- специальные системы военного назначения.

На практике наиболее хорошо себя зарекомендовали для работы с мелкомасштабными «природными» картами (геология, сельское хозяйство, навигация, экология и т. п.) такие ГИС, как ArcInfo и ArcView GIS. Обе системы разработаны американской компанией ESRI (www.esri.com, www.dataplus.ru) и весьма распространены в мире.

Из относительно простых западных ГИС, которые начинали свою родословную с анализа территорий в объеме, необходимом для бизнеса и относительно простых применений, можно назвать систему MapInfo, которая также распространена в мире весьма широко. Эта система очень быстро прогрессирует и сегодня может составить конкуренцию самым развитым ГИС.

Корпорацией Intergraph (www.intergraph.com) поставляется ГИС MGE, базирующаяся на основе AutoCAD-подобной системы MicroStation, выпускаемой в свою очередь компанией Bently. Система MGE представляет собой целое семейство различных программных продуктов, помогающих решать набольшее множество задач, существующих в области геоинформатики.

Все указанные продукты имеют и Интернет-ГИС-серверы, позволяющие публиковать цифровые карты в Интернете. Правда, приходится говорить только о вьюерах, поскольку обеспечить сегодня редактирование топологических карт со стороны удаленного клиента Интернета нельзя по причине недостаточной развитости как ГИС-, так и Интернет-технологий.

Буквально недавно вышла на рынок ГИС и Microsoft, подтвердив тем самым, что ГИС станет в ближайшем будущем такой системой, которую должен иметь на своем компьютере всякий мало-мальски уважающий себя пользователь, как он имеет сегодня у себя Excel или Word. Microsoft выпустила продукт MapPoint (Microsoft MapPoint 2000 Business Mapping Software), который войдет в состав Office 2000. Эта компонента офисного продукта будет ориентирована в основном на бизнес-планирование и анализ.

Повторением концепции ArcInfo, но сильно уступающей последней по функциональной полноте, является отечественная система GeoDraw, разработанная в ЦГИ ИГРАН (г. Москва). Возможности ее ограничены сегодня в основном мелкомасштабными картами. С нашей точки зрения, значительно «сильнее» здесь выглядит «старейшина» отечественной геоинформатики — ГИС Sinteks ABRIS. В последней хорошо представлены функции по анализу пространственной информации.

В геологии сильны позиции ГИС ПАРК (Ланэко, г. Москва), в которой также реализованы уникальные методы моделирования соответствующих процессов.

Наиболее «продвинутыми» в области представления и дежурства крупномасштабных насыщенных карт городов и генпланов крупных предприятий можно считать две отечественные системы: GeoCosm (ГЕОИД, г. Геленджик) и «ИнГео» (ЦСИ «Интегро», г. Уфа, www.integro.ru). Эти системы — одни из самых молодых и потому разрабатывались сразу с использованием самых современных технологий. А систему «ИнГео» разрабатывали даже не столько геодезисты, сколько специалисты, относящие себя к профессионалам в области имитационного моделирования и кадастровых систем.

В целом в России едва ли не в каждой организации создают свою ГИС. Однако этот процесс — весьма непростой, и вероятность его завершения неудачно несравненно более высока, чем вероятность безпроблемной реализации, не говоря уже о возможности выхода коммерческого продукта, допускающего отчуждение от разработчиков.

3.3. ТЕХНОЛОГИИ ЗАЩИТЫ ИНФОРМАЦИИ

Наряду с позитивным влиянием на все стороны человеческой деятельности широкое внедрение информационных технологий привело к появлению новых угроз безопасности людей. Это связано с тем обстоятельством, что информация, создаваемая, хранимая и обрабатываемая средствами вычислительной техники, стала определять действия большей части людей и технических систем. В связи с этим резко возросло количество предпосылок к нанесению ущерба, связанных с хищением информации, так как воздействовать на любую систему (социальную, биологическую или техническую) с целью ее уничтожения, снижения эффективности функционирования или воровства ее ресурсов (денег, товаров, оборудования) возможно только в том случае, когда известна информация о ее структуре и принципах функционирования.

Все виды информационных угроз можно разделить на две большие группы:

- отказы и нарушения работоспособности программных и технических средств;
- преднамеренные угрозы, заранее планируемые злоумышленниками для нанесения вреда.

Выделяют следующие основные группы причин сбоев и отказов в работе компьютерных систем:

- нарушения физической и логической целостности хранящихся в оперативной и внешней памяти структур данных, возникающие по причине старения или преждевременного износа их носителей;
- нарушения, возникающие в работе аппаратных средств из-за их старения или преждевременного износа;
- нарушения физической и логической целостности хранящихся в оперативной и внешней памяти структур данных, возникающие по причине некорректного использования компьютерных ресурсов;
- нарушения, возникающие в работе аппаратных средств из-за неправильного использования или повреждения, в том числе из-за неправильного использования программных средств;

- неустраненные ошибки в программных средствах, не выявленные в процессе отладки и испытаний, а также оставшиеся в аппаратных средствах после их разработки.

Помимо естественных способов выявления и своевременного устранения указанных выше причин, используют следующие специальные способы защиты информации от нарушений работоспособности компьютерных систем:

- внесение структурной, временной, информационной и функциональной избыточности компьютерных ресурсов;
- защита от некорректного использования ресурсов компьютерной системы;
- выявление и своевременное устранение ошибок на этапах разработки программно-аппаратных средств.

Структурная избыточность компьютерных ресурсов достигается за счет резервирования аппаратных компонентов и машинных носителей данных, организации замены отказавших и своевременного пополнения резервных компонентов. Структурная избыточность составляет основу остальных видов избыточности.

Внесение информационной избыточности выполняется путем периодического или постоянного (фонового) резервирования данных на основных и резервных носителях. Зарезервированные данные обеспечивают восстановление случайно или преднамеренно уничтоженной и искаженной информации. Для восстановления работоспособности компьютерной системы после появления устойчивого отказа, кроме резервирования обычных данных, должна заблаговременно резервироваться и системная информация, а также подготавливаться программные средства восстановления.

Функциональная избыточность компьютерных ресурсов достигается дублированием функций или внесением дополнительных функций в программно-аппаратные ресурсы вычислительной системы для повышения ее защищенности от сбоев и отказов, например периодическое тестирование и восстановление, а также самотестирование и самовосстановление компонентов компьютерной системы.

Защита от некорректного использования информационных ресурсов заключается в корректном функционировании программного обеспечения с позиции использования ресурсов вычислительной системы. Программа может четко и своевременно выполнять свои функции, но некорректно использовать компьютерные ресурсы из-за отсутствия всех необходимых функций (например, изолирование участков оперативной памяти для операционной системы и прикладных программ, защита системных областей на внешних носителях, поддержка целостности и непротиворечивости данных).

Выявление и устранение ошибок при разработке программно-аппаратных средств достигается путем качественного выполнения базовых стадий разработки на основе системного анализа концепции, проектирования и реализации проекта.

Однако основным видом угроз целостности и конфиденциальности информации являются преднамеренные угрозы, заранее планируемые злоумышленниками для нанесения вреда, которые можно разделить на две группы:

- угрозы, реализация которых выполняется при постоянном участии человека;
- угрозы, реализация которых после разработки злоумышленником соответствующих компьютерных программ выполняется этими программами без непосредственного участия человека.

Задачи по защите от угроз каждого вида одинаковы:

- запрещение несанкционированного доступа к ресурсам вычислительных систем;
- невозможность несанкционированного использования компьютерных ресурсов при осуществлении доступа;
- своевременное обнаружение факта несанкционированных действий, устранение их причин и последствий.

Основным способом запрещения несанкционированного доступа к ресурсам вычислительных систем является подтверждение подлинности пользователей и разграничение их доступа к информационным ресурсам, включающего следующие этапы:

- идентификация;
- установление подлинности (аутентификация);
- определение полномочий для последующего контроля и разграничения доступа к компьютерным ресурсам.

Идентификация необходима для указания компьютерной системе уникального идентификатора обращающегося к ней пользователя. Идентификатор может представлять собой любую последовательность символов и должен быть заранее зарегистрирован в системе администратора службы безопасности. В процессе регистрации заносится следующая информация:

- фамилия, имя, отчество (при необходимости другие характеристики пользователя);
- уникальный идентификатор пользователя;
- имя процедуры установления подлинности;
- эталонная информация для подтверждения подлинности (например, пароль);
- ограничения на используемую эталонную информацию (например, время действия пароля);
- полномочия пользователя по доступу к компьютерным ресурсам.

Установление подлинности (аутентификация) заключается в проверке истинности полномочий пользователя.

Общая схема идентификации и установления подлинности пользователя представлена на рисунке 3.6.

Для особо надежного опознания для идентификации используются технические средства, определяющие индивидуальные характеристики человека (голос, отпечатки пальцев, структура значка). Однако такие методы требуют значительных затрат и поэтому используются редко. Наиболее массово используемыми являются парольные методы проверки подлинности пользователей, которые можно разделить на две группы:

- простые пароли;
- динамически изменяющиеся пароли.

Простой пароль не изменяется от сеанса к сеансу в течение установленного времени его существования.

Во втором случае пароль изменяется по правилам, определяемым используемым методом. Выделяют следу-

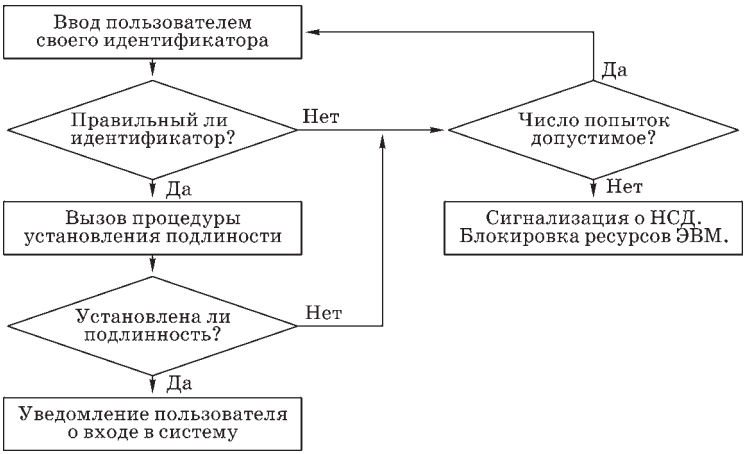


Рис. 3.6
Схема идентификации и установления подлинности пользователя

ющие методы реализации динамически изменяющихся паролей:

- методы модификации простых паролей, например случайная выборка символов пароля и одноразовое использование паролей;
- метод «запрос — ответ», основанный на предъявлении пользователю случайно выбираемых запросов из имеющегося массива;
- функциональные методы, основанные на использовании некоторой функции F с динамически изменяющимися параметрами (дата, время, день недели и др.), с помощью которой определяется пароль.

Для защиты от несанкционированного входа в компьютерную систему используются как общесистемные, так и специализированные программные средства защиты.

После идентификации и аутентификации пользователя система защиты должна определить его полномочия для последующего контроля санкционированного доступа к компьютерным ресурсам (разграничение доступа). В качестве компьютерных ресурсов рассматриваются:

- программы;
- внешняя память (файлы, каталоги, логические диски);

- информация, разграниченная по категориям в базах данных;
- оперативная память;
- время (приоритет) использования процессора;
- порты ввода-вывода;
- внешние устройства.

Различают следующие виды прав пользователей по доступу к ресурсам:

- всеобщее право (полное предоставление ресурса);
- функциональное или частичное право;
- временное право.

Наиболее распространенными способами разграничения доступа являются:

- разграничение доступа по спискам (пользователей или ресурсов);
- использование матрицы установления полномочий (строки матрицы — идентификаторы пользователей, столбцы — ресурсы компьютерной системы);
- разграничение доступа по уровням секретности и категориям (например, общий доступ, конфиденциально, секретно);
- парольное разграничение доступа.

Защита информации от исследования и копирования предполагает криптографическое закрытие защищаемых от хищения данных. Задачей криптографии является обратимое преобразование некоторого понятного исходного текста (открытого текста) в кажущуюся случайной последовательность некоторых знаков, часто называемых шифротекстом или криптограммой. В шифре выделяют два основных элемента — алгоритм и ключ. Алгоритм шифрования представляет собой последовательность преобразований обрабатываемых данных, зависящих от ключа шифрования. Ключ задает значения некоторых параметров алгоритма шифрования, обеспечивающих шифрование и дешифрование информации. В криптографической системе информация I и ключ K являются входными данными для шифрования (рис. 3.7) и дешифрования (рис. 3.8) информации. При похищении информации необходимо знать ключ и алгоритм шифрования.

Рис. 3.7
Процесс шифрования

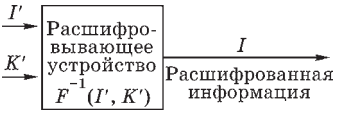


Рис. 3.8
Процесс дешифрования

По способу использования ключей различают два типа криптографических систем: симметрические и асимметрические.

В симметрических (одноключевых) криптографических системах ключи шифрования и дешифрования либо одинаковы, либо легко выводятся один из другого.

В асимметрических (двухключевых или системах с открытым ключом) криптографических системах ключи шифрования и дешифрования различаются таким образом, что с помощью вычислений нельзя вывести один ключ из другого.

Скорость шифрования в двухключевых криптографических системах намного ниже, чем в одноключевых. Поэтому асимметрические системы используют в двух случаях:

- для шифрования секретных ключей, распределенных между пользователями вычислительной сети;
- для формирования цифровой подписи.

Одним из сдерживающих факторов массового применения методов шифрования является потребление значительных временных ресурсов при программной реализации большинства хорошо известных шифров (DES, FEAL, REDOC, IDEA, ГОСТ).

Одной из основных угроз хищения информации является угроза доступа к остаточным данным в оперативной и внешней памяти компьютера. Под остаточной информацией понимаются данные, оставшиеся в освободившихся участках оперативной и внешней памяти:

- данные, оставшиеся после удаления файлов пользователя;
- данные, оставшиеся после удаления временных файлов без ведома пользователя;
- данные, находящиеся в неиспользуемых хвостовых частях последних кластеров, занимаемых файлами;

- данные, оставшиеся в кластерах, освобожденных после уменьшения размеров файлов;
- данные, оставшиеся после форматирования дисков.

Основным способом защиты от доступа к конфиденциальным остаточным данным является своевременное уничтожение данных в следующих областях памяти компьютера:

- в рабочих областях оперативной и внешней памяти, выделенных пользователю, после окончания им сеанса работы;
- в местах расположения файлов после выдачи запросов на их удаление.

Уничтожение остаточных данных может быть реализовано либо средствами операционных сред, либо с помощью специализированных программ. Использование специализированных программ (автономных или в составе системы защиты) обеспечивает гарантированное уничтожение информации.

Подсистема защиты от компьютерных вирусов (специально разработанных программ для выполнения несанкционированных действий) является одним из основных компонентов системы защиты информации и процесса ее обработки в вычислительных системах.

Выделяют три уровня защиты от компьютерных вирусов:

- защита от проникновения в вычислительную систему вирусов известных типов;
- углубленный анализ на наличие вирусов известных и неизвестных типов, преодолевших первый уровень защиты;
- защита от деструктивных действий и размножения вирусов, преодолевших первые два уровня.

Поиск и обезвреживание вирусов осуществляются как автономными антивирусными программными средствами (сканеры), так и в рамках комплексных систем защиты информации.

Среди транзитных сканеров, которые загружаются в оперативную память, наибольшей популярностью в нашей стране пользуются антивирусные программы Aidstest Дмитрия Лозинского и DrWeb Игоря Данилова. Эти про-

граммы просты в использовании и для детального ознакомления с руководством по каждой из них следует прочитать файл, поставляемый вместе с антивирусным средством.

Широкое внедрение в повседневную практику компьютерных сетей, их открытость, масштабность делают проблему защиты информации исключительно сложной. Выделяют две базовые подзадачи:

- обеспечение безопасности обработки и хранения информации в каждом из компьютеров, входящих в сеть;
- защита информации, передаваемой между компьютерами сети.

Решение первой задачи основано на многоуровневой защите автономных компьютерных ресурсов от несанкционированных и некорректных действий пользователей и программ, рассмотренных выше.

Безопасность информации при сетевом обмене данными требует также обеспечения их конфиденциальности и подлинности. Защита информации в процессе передачи достигается на основе защиты каналов передачи данных, а также криптографического закрытия передаваемых сообщений. В идеальном случае защита каналов передачи данных должна обеспечивать их защиту как от нарушений работоспособности, так и несанкционированных действий (например, подключения к линиям связи). По причине большой протяженности каналов связи, а также возможной доступности их отдельных участков (например, при беспроводной связи) защита каналов передачи данных от несанкционированных действий экономически неэффективна, а в ряде случаев невозможна. Поэтому реально защита каналов передачи данных строится на основе защиты нарушений их работоспособности. На рисунке 3.9 представлены цели и способы защиты передаваемых данных.

В силу актуальности проблемы защиты информации рынок насыщен значительным количеством средств защиты информации, которые можно классифицировать следующим образом:

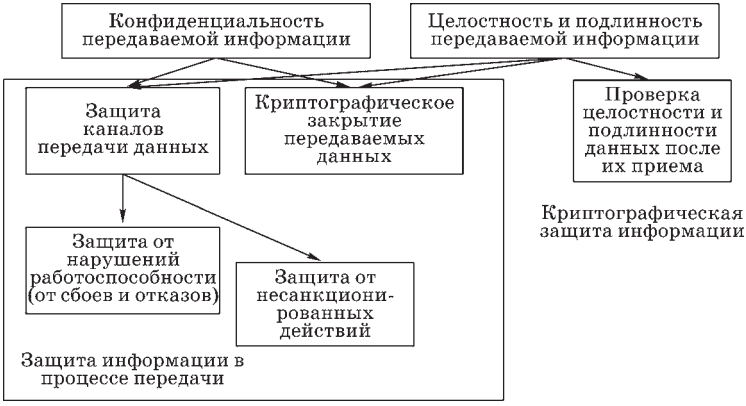


Рис. 3.9
Цели и способы защиты передаваемых данных

1) средства защиты от несанкционированного доступа (НСД): средства авторизации, мандатное управление доступом, избирательное управление доступом, управление доступом на основе ролей, журналирование (также называется аудит);

2) системы анализа и моделирования информационных потоков (CASE-системы);

3) системы мониторинга сетей: системы обнаружения и предотвращения вторжений (IDS/IPS), системы предотвращения утечек конфиденциальной информации (DLP-системы);

4) анализаторы протоколов;

5) антивирусные средства;

6) межсетевые экраны;

7) криптографические средства: шифрование; цифровая подпись;

8) системы резервного копирования;

9) системы бесперебойного питания: источники бесперебойного питания, резервирование нагрузки, генераторы напряжения;

10) системы аутентификации: пароль, ключ доступа (физический или электронный), сертификат, биометрия;

11) средства предотвращения взлома корпусов и краж оборудования;

12) средства гарантированного уничтожения информации;

13) средства контроля доступа в помещения;

14) инструментальные средства анализа систем защиты: мониторинговый программный продукт.

3.4. CASE-ТЕХНОЛОГИИ

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности ИС, создаваемых в различных областях экономики. Современные крупные проекты ИС характеризуются, как правило, следующими особенностями:

- сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;
- наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);
- отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;
- необходимость интеграции существующих и вновь разрабатываемых приложений;
- функционирование в неоднородной среде на нескольких аппаратных платформах;
- разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;
- существенная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков и, с другой

стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИС.

Перечисленные факторы способствовали появлению программно-технологических средств специального класса — CASE-средств, реализующих CASE-технологию создания и сопровождения ИС. Термин CASE (Computer Aided Software Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т. д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

- подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;
- широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;

- внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств [16].

При использовании методологий структурного анализа появился ряд ограничений (сложность понимания, большая трудоемкость и стоимость использования, неудобство внесения изменений в проектные спецификации и т. д.). С самого начала CASE-технологии развивались с целью преодоления этих ограничений путем автоматизации процессов анализа и интеграции поддерживающих средств. Они обладают достоинствами и возможностями, перечисленными ниже.

Единый графический язык. CASE-технологии обеспечивают всех участников проекта, включая заказчиков, единым строгим, наглядным и интуитивно понятным графическим языком, позволяющим получать обозримые компоненты с простой и ясной структурой. При этом программы представляются двумерными схемами (которые проще в использовании, чем многостраничные описания), позволяющими заказчику участвовать в процессе разработки, а разработчикам — общаться с экспертами предметной области, разделять деятельность системных аналитиков, проектировщиков и программистов, облег-

чая им защиту проекта перед руководством, а также обеспечивая легкость сопровождения и внесения изменений в систему.

Единая БД проекта. Основа CASE-технологии — использование базы данных проекта (репозитория) для хранения всей информации о проекте, которая может разделяться между разработчиками в соответствии с их правами доступа. Содержимое репозитория включает не только информационные объекты различных типов, но и отношения между их компонентами, а также правила использования или обработки этих компонентов. Репозиторий может хранить свыше 100 типов объектов: структурные диаграммы, определения экранов и меню, проекты отчетов, описания данных, логика обработки, модели данных, их организации и обработки, исходные коды, элементы данных и т. п.

Интеграция средств. На основе репозитория осуществляется интеграция CASE-средств и разделение системной информации между разработчиками. При этом возможности репозитория обеспечивают несколько уровней интеграции: общий пользовательский интерфейс по всем средствам, передачу данных между средствами, интеграцию этапов разработки через единую систему представления фаз жизненного цикла, передачу данных и средств между различными платформами.

Поддержка коллективной разработки и управления проектом. CASE-технология поддерживает групповую работу над проектом, обеспечивая возможность работы в сети, экспорт-импорт любых фрагментов проекта для их развития и/или модификации, а также планирование, контроль, руководство и взаимодействие, т. е. функции, необходимые в процессе разработки и сопровождения проектов. Эти функции также реализуются на основе репозитория. В частности, через репозиторий может осуществляться контроль безопасности (ограничения и привилегии доступа), контроль версий и изменений и др.

Макетирование. CASE-технология дает возможность быстро строить макеты (прототипы) будущей системы, что позволяет заказчику на ранних этапах разработки

оценить, насколько она приемлема для будущих пользователей и устраивает его.

Генерация документации. Вся документация по проекту генерируется автоматически на базе репозитория (как правило, в соответствии с требованиями действующих стандартов). Несомненное достоинство CASE-технологии заключается в том, что документация всегда отвечает текущему состоянию дел, поскольку любые изменения в проекте автоматически отражаются в репозитории (известно, что при традиционных подходах к разработке ПО документация в лучшем случае запаздывает, а ряд модификаций вообще не находит в ней отражения).

Верификация проекта. CASE-технология обеспечивает автоматическую верификацию и контроль проекта на полноту и состоятельность на ранних этапах разработки, что влияет на успех разработки в целом — по статистическим данным анализа пяти крупных проектов фирмы TRW (США) ошибки проектирования и кодирования составляют соответственно 64 и 32% от общего числа ошибок, а ошибки проектирования в 100 раз труднее обнаружить на этапе сопровождения ПО, чем на этапе анализа требований.

Автоматическая генерация объектного кода. Генерация программ в машинном коде осуществляется на основе репозитория и позволяет автоматически построить до 85–90% объектного кода или текстов на языках высокого уровня.

Сопровождение и реинжиниринг. Сопровождение системы в рамках CASE-технологии характеризуется сопровождением проекта, а не программных кодов. Средства реинжиниринга и обратного инжиниринга позволяют создавать модель системы из ее кодов и интегрировать полученные модели в проект, автоматически обновлять документацию при изменении кодов и т. п.

При использовании CASE-технологий изменяются все фазы жизненного цикла ИС, причем наибольшие изменения касаются фаз анализа и проектирования. В таблице 3.1 приведены основные изменения жизненного цикла ИС при использовании CASE-технологий по сравнению с традиционной технологией разработки.

Таблица 3.1

Сравнительная характеристика основных изменений жизненного цикла ИС

Традиционная технология разработки	Разработка с помощью CASE-технологий
Основные усилия — на кодирование и тестирование	Основные усилия — на анализ и проектирование
«Бумажные» спецификации	Быстрое итеративное макетирование
Ручное кодирование	Автоматическая генерация машинного кода
Тестирование ПО	Автоматический контроль проекта
Сопровождение программного кода	Сопровождение проекта

В таблице 3.2 приведены оценки трудозатрат по фазам жизненного цикла программного обеспечения (ПО) в зависимости от используемой технологии разработки.

Таблица 3.2

Оценка трудозатрат по фазам жизненного цикла программного обеспечения (ПО) в зависимости от используемой технологии разработки

Технология разработки	Анализ	Проектирование	Программирование	Тестирование
Традиционная	20%	15%	20%	45%
Структурная методология (вручную)	30%	30%	15%	25%
CASE-технология	40%	40%	5%	15%

Перейдем к характеристике современных CASE-систем, которые охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО [16].

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 западны-

ми фирмами. Полный комплекс CASE-средств, обеспечивающий поддержку жизненного цикла 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 (потоков данных, «сущность — связь» 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 по следующим признакам:

- функциональной ориентации;
- применяемым методологиям и моделям систем и баз данных (БД);
- степени интегрированности с системами управления базами данных (СУБД);
- доступным платформам.

Классификация по функциональной ориентации представляет наибольший интерес и в основном совпадает с компонентным составом CASE-средств и включает следующие основные типы:

1) средства анализа (Upper CASE), предназначенные для построения и анализа моделей предметной области (Design/IDEF, BPwin);

2) средства анализа и проектирования (Middle CASE), поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций (Vantage Team Builder,

Designer/2000, Silverrun, PRO-IV, CASE.Аналитик). Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;

3) средства проектирования БД, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin, S-Designor и DataBase Designer (ORACLE). Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;

4) средства разработки приложений. К ним относятся средства 4GL (Uniface, JAM, PowerBuilder, Developer/2000, New Era, SQLWindows, Delphi и др.) и генераторы кодов, входящие в состав Vantage Team Builder, PRO-IV и частично — в Silverrun;

5) 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, Object Team).

Российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами:

- Vantage Team Builder (Westmount I-CASE);
- Designer/2000;
- Silverrun;
- ERwin + BPwin;
- S-Designor;
- CASE.Аналитик;
- Rational Rose.

Кроме того, на рынке постоянно появляются как новые для отечественных пользователей системы, так и новые версии и модификации перечисленных систем. Наиболь-

ший интерес представляет CASE-средство фирмы Rational Software Corporation (США) Rational Rose, которое предназначено для автоматизации этапов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации. Rational Rose использует синтез-методологию объектно-ориентированного анализа и проектирования, основанную на подходах трех ведущих специалистов в данной области: Буча, Рамбо и Джекобсона. Разработанная ими универсальная нотация для моделирования объектов (UML — Unified Modeling Language) претендует на роль стандарта в области объектно-ориентированного анализа и проектирования.

Ориентация на объектно-ориентированные методы объясняется следующими причинами:

- возможность сборки программной системы из готовых повторно используемых компонентов;
- возможность накопления проектных решений в виде библиотек классов на основе механизмов наследования;
- простота внесения изменений в проекты за счет инкапсуляции данных в объектах;
- быстрая адаптация приложений к изменяющимся условиям за счет использования свойств наследования и полиформизма;
- возможность организации параллельной работы аналитиков, проектировщиков и программистов.

Концепция объектно-ориентированного подхода и концепция распределенных вычислений стали базой для создания консорциума Object Management Group (OMG), членами которой являются более 500 ведущих компьютерных компаний (Sun, DEC, IBM, HP, Motorola и др.). Основным направлением деятельности консорциума является разработка спецификаций и стандартов для создания распределенных объектных систем в разнородных средах. Базисом стали спецификации под названием Object Management Architecture (OMA).

OMA состоит из четырех основных компонентов, представляющих спецификации различных уровней поддержки приложений (рис. 3.10):

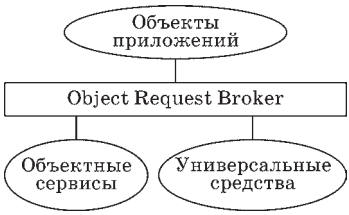


Рис. 3.10
Спецификация OMA

- архитектура брокера запросов объектов (CORBA — Common Object Request Broker Architecture) определяет механизмы взаимодействия объектов в разнородной сети;
- объектные сервисы (Object Services) являются основными системными сервисами, используемыми разработчиками для создания приложений;
- универсальные средства (Common Facilities) являются высокоуровневыми системными сервисами, ориентированными на поддержку пользовательских приложений (электронная почта, средства печати и др.);
- прикладные объекты (Application Object) предназначены для решения конкретных прикладных задач.

Исходя из основных положений объектно-ориентированного подхода, рассмотрим концепцию идеального объектно-ориентированного CASE-средства.

Существует несколько объектно-ориентированных методов, авторами наиболее распространенных из них являются Г. Буч, Д. Рамбо, И. Джекобсон. В настоящее время наблюдается процесс сближения объектно-ориентированных методов. В частности, указанные выше авторы создали и выпустили несколько версий унифицированного метода UML (Unified Modeling Language — унифицированный язык моделирования).

Классическая постановка задачи разработки программной системы (инжиниринг) представляет собой спиральный цикл итеративного чередования этапов объектно-ориентированного анализа, проектирования и реализации (программирования).

В реальной практике в большинстве случаев имеется предыстория в виде совокупности разработанных и внедренных программ, которые целесообразно использовать при разработке новой системы. Процесс проектирования в таком случае основан на реинжиниринге программных кодов, при котором путем анализа тек-

стов программ восстанавливается исходная модель программной системы.

Современные CASE-средства поддерживают процессы инжиниринга и автоматизированного реинжиниринга.

Идеальное объектно-ориентированное CASE-средство (рис. 3.11) должно содержать четыре основных блока: анализ, проектирование, разработка и инфраструктура [17].

Основные требования к блоку анализа:

- возможность выбора выводимой на экран информации из всей совокупности данных, описывающих модели;

<table>
<tr><th colspan="2">Анализ</th><th>Проектирование</th><th>Реализация</th></tr>
<tr><td>Возможность добавлять пояснительные надписи к диаграммам и в документацию</td><td>Возможность создавать различные представления и скрывать ненужные в данный момент слои системы</td><td>Возможность просматривать и выбирать элементы и бизнес-объекты для использования в системе</td><td>Возможность генерировать заготовки программного кода на нескольких объектно-ориентированных языках</td></tr>
<tr><td colspan="2">Среда для создания диаграмм разнообразных моделей</td><td>Возможность создания пользовательского интерфейса (поддержка OLE, ActiveX, OpenDoc, HTML)</td><td>Возможность проверки кода на синтаксическую корректность</td></tr>
<tr><td>Поддержка различных нотаций</td><td>Возможность динамического моделирования событий в системе</td><td>Возможности определения бизнес-модели и бизнес-правил</td><td>Возможность генерировать код для 4GL и клиент-серверных продуктов (PowerBuilder, Forte, VisualAge, VisualWorks)</td></tr>
<tr><td>Возможность генерации документации для печати</td><td>Возможность динамической коррекции одной диаграммы из другой</td><td>Возможность связи с объектно-ориентированными базами данных и распределенными модулями (поддержка CORBA, DCOM, IIOP, HTML)</td><td></td></tr>
<tr><td colspan="4">Инфраструктура</td></tr>
<tr><td colspan="2">Контроль версий. Блокирование и согласование частей системы при групповой разработке</td><td>Репозиторий</td><td>Возможность реинженеринга программного кода, 4GL, клиент-серверных систем в диаграммы моделей</td></tr>
</table>

Рис. 3.11
Идеальное объектно-ориентированное CASE-средство

- согласованность диаграмм при хранении их в репозитарии;
- внесение комментариев в диаграммы и соответствующую документацию для фиксации проектных решений;
- возможность динамического моделирования в терминах событий;
- поддержка нескольких нотаций (хотя бы три нотации Г. Буча, И. Джекобсона и OMT).

Основные требования к блоку проектирования:

- поддержка всего процесса проектирования приложения;
- возможность работы с библиотеками, средства поиска и выбора;
- возможность разработки пользовательского интерфейса;
- поддержка стандартов OLE, ActiveX и доступ к библиотекам HTML или Java;
- поддержка разработки распределенных или двух- и трехзвенных клиент-серверных систем (работа с CORBA, DCOM, Интернетом).

Основные требования к блоку реализации:

- генерация кода полностью из диаграмм;
- возможность доработки приложений в клиент-серверных CASE-средствах типа Power Builder;
- реинжиниринг кодов и внесение соответствующих изменений в модель системы;
- наличие средств контроля, которые позволяют выявлять несоответствие между диаграммами и генерируемыми кодами и обнаруживать ошибки как на стадии проектирования, так и на стадии реализации.

Основные требования к блоку инфраструктуры:

- наличие репозитория на основе базы данных, отвечающего за генерацию кода, реинжиниринг, отображение кода на диаграммах, а также обеспечивающего соответствие между моделями и программными кодами;
- обеспечение командной работы (многопользовательской работы и управление версиями) и реинжиниринга.

Сравнительный анализ CASE-систем показывает, что на сегодняшний день одним из наиболее приближенных к идеальному варианту CASE-средств является семейство Rational Rose фирмы Rational Software Corporation. Следует отметить, что именно здесь работают авторы унифицированного языка моделирования — Г. Буч, Д. Рамбо и И. Джекобсон, под руководством которых ведется разработка нового CASE-средства, поддерживающего UML.

Конкретный вариант Rational Rose определяется языком, на котором генерируются коды программ (C++, Smalltalk, PowerBuilder, Ada, SQLWindows и ObjectPro). Основной вариант — Rational Rose/C++ — позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций, а также генерировать программные коды на C++. Кроме того, Rational Rose содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонент в новых проектах.

В основе работы Rational Rose лежит построение различного рода диаграмм и спецификаций, определяющих логическую и физическую структуры модели, ее статические и динамические аспекты. В их число входят диаграммы классов, состояний, сценариев, модулей, процессов. Rational Rose функционирует на различных платформах: IBM PC (в среде Windows), Sun SPARC stations (UNIX, Solaris, SunOS), Hewlett-Packard (HP UX), IBM RS/6000 (AIX).

Выделим основные критерии оценки и выбора CASE-средств:

1) функциональные характеристики:

- среда функционирования: проектная среда, программное обеспечение/технические средства, технологическая среда;
- функции, ориентированные на фазы жизненного цикла: моделирование, реализация, тестирование;
- общие функции: документирование, управление конфигурацией, управление проектом;

2) надежность;

3) простота использования;

4) эффективность;

5) сопровождаемость;

6) переносимость;

7) общие критерии (стоимость, затраты, эффект внедрения, характеристики поставщика).

Данные критерии подробно изложены в стандартах IEEE Std 1348-1995. IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools и IEEE Std 1209-1992 Recommended Practice for the Evaluation and Selection of CASE Tools.

3.5. ТЕЛЕКОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

Спектр телекоммуникационных технологий чрезвычайно широк, что во многом объясняется высоким уровнем стандартизации и унификации, способствовавшим их распространению. Из всего многообразия остановимся на телекоммуникационных технологиях, обеспечивающих удаленный доступ и распределенную обработку.

Задача разработчиков распределенных систем — спроектировать программное и аппаратное обеспечение так, чтобы предоставить все необходимые характеристики распределенной системы. А для этого требуется знать преимущества и недостатки различных архитектур распределенных систем.

Выделяют следующие виды распределенных архитектур ИС [18]:

- архитектура «файл-сервер»;
- двухзвенная архитектура «клиент-сервер»;
- многозвенная архитектура «клиент-сервер»;
- архитектура веб-приложений;
- сервис-ориентированная архитектура.

Рассмотрим каждую из этих архитектур.

Архитектура «файл-сервер». Файл-серверные приложения — приложения, схожие по своей структуре с локальными приложениями и использующие сетевой ресурс для хранения программы и данных.

Классическое представление информационной системы в архитектуре «файл-сервер» представлено на рисунке 3.12.

Организация информационных систем на основе использования выделенных файл-серверов все еще является распространенной в связи с наличием большого количества персональных компьютеров разного уровня развитости и сравнительной дешевизны связывания PC в локальные сети. Конечно, основным достоинством данной архитектуры является простота организации. Проектировщики и разработчики информационной системы находятся в привычных и комфортных условиях IBM PC в среде MS-DOS, Windows или какого-либо облегченного варианта Windows Server. Имеются удобные и развитые средства разработки графического пользовательского интерфейса, простые в использовании средства разработки систем баз данных и/или СУБД.

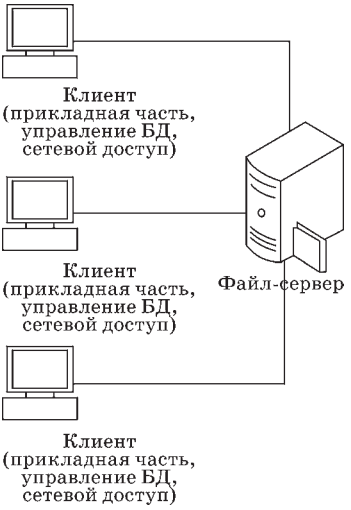


Рис. 3.12
Классическое представление архитектуры «файл-сервер»

Достоинства такой архитектуры:

- многопользовательский режим работы с данными;
- удобство централизованного управления доступом;
- низкая стоимость разработки;
- высокая скорость разработки;
- невысокая стоимость обновления и изменения ПО.

Недостатки:

- проблемы многопользовательской работы с данными: последовательный доступ, отсутствие гарантии целостности;
- низкая производительность (зависит от производительности сети, сервера, клиента);
- плохая возможность подключения новых клиентов;
- ненадежность системы.

Архитектура «клиент-сервер» (Client-server) — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением.

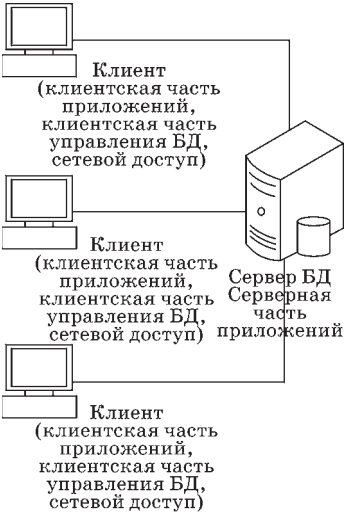


Рис. 3.13
Классическое представление архитектуры «клиент-сервер»

Первоначально системы такого уровня базировались на классической двухуровневой клиент-серверной архитектуре (Two-tier architecture). Под клиент-серверным приложением в этом случае понимается информационная система, основанная на использовании *серверов баз данных*.

Схематически такую архитектуру можно представить, как показано на рисунке 3.13.

На стороне клиента выполняется код приложения, в который обязательно входят компоненты, поддерживающие интерфейс с конечным пользователем, производящие отчеты, выполняющие другие специфичные для приложения функции.

Клиентская часть приложения взаимодействует с клиентской частью программного обеспечения управления базами данных, которая фактически является индивидуальным представителем СУБД для приложения.

Заметим, что интерфейс между клиентской частью приложения и клиентской частью сервера баз данных, как правило, основан на использовании языка SQL. Поэтому такие функции, как, например, предварительная обработка форм, предназначенных для запросов к базе

данных, или формирование результирующих отчетов выполняются в коде приложения.

Наконец, клиентская часть сервера баз данных, используя средства сетевого доступа, обращается к серверу баз данных, передавая ему текст оператора языка SQL.

Посмотрим теперь, что же происходит на стороне сервера баз данных. В продуктах практически всех компаний сервер получает от клиента текст оператора на языке SQL.

Сервер производит компиляцию полученного оператора.

Далее (если компиляция завершилась успешно) происходит выполнение оператора.

Преимуществами данной архитектуры являются:

- возможность в большинстве случаев распределить функции вычислительной системы между несколькими независимыми компьютерами в сети;
- все данные хранятся на сервере, который, как правило, защищен гораздо лучше большинства клиентов, а также на сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа;
- поддержка многопользовательской работы;
- гарантия целостности данных.

Недостатки:

- неработоспособность сервера может сделать неработоспособной всю вычислительную сеть;
- администрирование данной системы требует квалифицированного профессионала;
- высокая стоимость оборудования;
- бизнес-логика приложений осталась в клиентском ПО.

При проектировании информационной системы, основанной на архитектуре «клиент-сервер», большее внимание следует обращать на грамотность общих решений. Технические средства пилотной версии могут быть минимальными (например, в качестве аппаратной основы сервера баз данных может использоваться одна из рабочих станций). После создания пилотной версии нужно провести дополнительную исследовательскую работу, чтобы выяснить узкие места системы. Только после это-

го необходимо принимать решение о выборе аппаратуры сервера, которая будет использоваться на практике.

Увеличение масштабов информационной системы не порождает принципиальных проблем. Обычным решением является замена аппаратуры сервера (и, может быть, аппаратуры рабочих станций, если требуется переход к локальному кешированию баз данных). В любом случае практически не затрагивается прикладная часть информационной системы.

Также данный вид архитектуры называют архитектурой с «толстым» клиентом.

Многозвенная архитектура. В случае большого числа пользователей возникают проблемы своевременной и синхронной замены версий клиентских приложений на рабочих станциях. Такие проблемы решаются в рамках многозвенной архитектуры (рис. 3.14). Часть общих приложений переносится на специально выделенный сервер приложений. Тем самым понижаются требования к ресурсам рабочих станций, которые будут называться «тонкими» клиентами. Данный способ организации вычислительного процесса является разновидностью архитектуры «клиент-сервер».

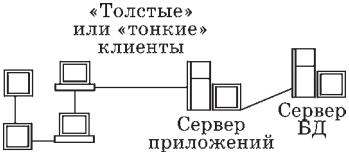


Рис. 3.14
Многозвенная архитектура

Использование многозвенной архитектуры может быть рекомендовано также в случае, если некоторая программа требует для своей работы много ресурсов, то может оказаться дешевле построить тонкую сеть с одним очень мощным сервером, чем использовать несколько мощных клиентных рабочих станций. Особенно это имеет значение, если данной программой пользуются не постоянно, а время от времени. На рисунке 3.15 представлена архитектура многозвенного приложения.

Разумное сочетание производительности сервера приложений и производительности рабочих станций позволят построить сеть, более дешевую при установке и эксплуатации.

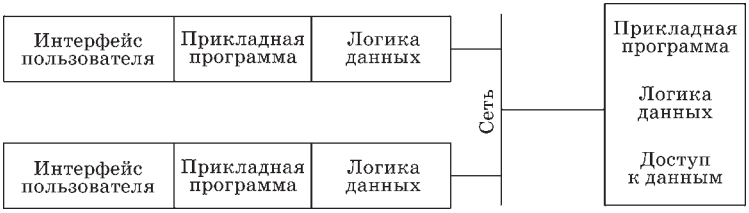


Рис. 3.15
Архитектура многозвенного приложения

Архитектура веб-приложений. Эта архитектура широко применяется в настоящее время и носит также название *архитектуры веб-сервисов*. Веб-сервис — это приложение, доступное через Интернет и предоставляющее некоторые услуги, форма которых не зависит от поставщика (так как используется универсальный формат данных — XML) и платформы функционирования. В настоящее время существуют три различные технологии, поддерживающие концепцию распределенных объектных систем. Это технологии EJB, CORBA и DCOM.

В основе веб-сервисов лежат открытые стандарты и протоколы: SOAP, UDDI и WSDL.

1. SOAP (Simple Object Access Protocol), разработанный консорциумом W3C, определяет формат запросов к веб-сервисам. Сообщения между веб-сервисом и его пользователем пакуются в так называемые SOAP-конверты (SOAP envelopes, иногда их еще называют XML-конвертами). Само сообщение может содержать либо запрос на осуществление какого-либо действия, либо ответ — результат выполнения этого действия.

2. WSDL (Web Service Description Language). Интерфейс веб-сервиса описывается в WSDL-документах (а WSDL — это подмножество XML). Перед развертыванием службы разработчик составляет ее описание на языке WSDL, указывает адрес веб-сервиса, поддерживаемые протоколы, перечень допустимых операций, форматы запросов и ответов.

3. UDDI (Universal Description, Discovery and Integration) — протокол поиска веб-сервисов в Интернете (http://www.uddi.org/). Представляет собой бизнес-реестр, в ко-

тором провайдеры веб-сервисов регистрируют службы, а разработчики находят необходимые сервисы для включения в свои приложения.

EJB. Основная идея, лежавшая в разработке технологии Enterprise JavaBeans, — создать такую инфраструктуру для компонент, чтобы они могли бы легко «вставляться» (plug in) и удаляться из серверов, тем самым увеличивая или снижая функциональность сервера. Технология Enterprise JavaBeans похожа на технологию JavaBeans в том смысле, что она использует ту же самую идею (а именно создание новой компоненты из уже существующих, готовых и настраиваемых компонент, аналогично RAD-системам), но во всем остальном Enterprise JavaBeans — совершенно иная технология.

Достоинства EJB:

- быстрое и простое создание;
- Java-оптимизация;
- кроссплатформенность;
- динамическая загрузка компонент-переходников;
- возможность передачи объектов по значению;
- встроенная безопасность.

Недостатки EJB:

- поддержка только одного языка — Java;
- трудность интегрирования с существующими приложениями;
- плохая масштабируемость;
- производительность;
- отсутствие международной стандартизации.

Благодаря своей легко используемой Java-модели EJB является самым простым и самым быстрым способом создания распределенных систем. EJB — хороший выбор для создания RAD-компонент и небольших приложений на языке Java. Конечно, EJB не такая мощная технология, как DCOM или CORBA. Тем самым роль RMI в создании больших, масштабируемых промышленных систем снижается.

DCOM. Distributed Component Object Model (DCOM) — программная архитектура, разработанная компанией Microsoft для распределения приложений между несколь-

кими компьютерами в сети. Программный компонент на одной из машин может использовать DCOM для передачи сообщения (его называют удаленным вызовом процедуры) к компоненту на другой машине. DCOM автоматически устанавливает соединение, передает сообщение и возвращает ответ удаленного компонента.

Для того чтобы различные фрагменты сложного приложения могли работать вместе через Интернет, необходимо обеспечить между ними надежные и защищенные соединения, а также создать специальную систему, которая направляет программный трафик.

Для решения этой задачи компания Microsoft создала распределенную компонентную объектную модель Distributed Component Object Model (DCOM), которая встраивается в операционные системы Windows NT 4.0 и Windows 98 и выше.

Достоинства DCOM:

- независимость от языка;
- динамический/статический вызов;
- динамическое нахождение объектов;
- масштабируемость;
- открытый стандарт (контроль со стороны TOG);
- множественность Windows-программистов.

Недостатки DCOM:

- сложность реализации;
- зависимость от платформы;
- нет именования через URL;
- нет проверки безопасности на уровне выполнении ActiveX компонент;
- отсутствие альтернативных разработчиков.

DCOM является лишь частным решением проблемы распределенных объектных систем. Он хорошо подходит для Microsoft-ориентированных сред. Как только в системе возникает необходимость работать с архитектурой, отличной от Windows, DCOM перестает быть оптимальным решением проблемы. Конечно, вскоре это положение может измениться, так как Microsoft стремится перенести DCOM и на другие платформы. Например, фирмой Software AG уже выпущена версия DCOM для Solaris

UNIX и планируется выпуск версий и для других версий UNIX. Но все-таки на сегодняшний день DCOM хорош лишь в качестве решения для систем, ориентированных исключительно на продукты Microsoft. Большие нарекания вызывает также отсутствие безопасности при исполнении ActiveX компонент, что может привести к неприятным последствиям.

CORBA. В конце 1980-х — начале 1990-х гг. многие ведущие фирмы-разработчики были заняты поиском технологий, которые принесли бы ощутимую пользу на все более изменчивом рынке компьютерных разработок. В качестве такой технологии была определена область распределенных компьютерных систем. Необходимо было разработать единообразную архитектуру, которая позволяла бы осуществлять повторное использование и интеграцию кода, что было особенно важно для разработчиков. Поэтому в мае 1989 г. была сформирована OMG (Object Managment Group). Как уже отмечалось, сегодня OMG насчитывает более 700 членов (в OMG входят практически все крупнейшие производители ПО, за исключением Microsoft).

Задачей консорциума OMG является определение набора спецификаций, позволяющих строить интероперабельные информационные системы. Спецификация OMG — The Common Object Request Broker Architecture (CORBA) является индустриальным стандартом, описывающим высокоуровневые средства поддерживания взаимодействия объектов в распределенных гетерогенных средах.

CORBA специфицирует инфраструктуру взаимодействия компонент (объектов) на представительском уровне и уровне приложений модели OSI. Она позволяет рассматривать все приложения в распределенной системе как объекты. Причем объекты могут одновременно играть роль и клиента, и сервера: роль клиента, если объект является инициатором вызова метода у другого объекта; роль сервера, если другой объект вызывает на нем какой-нибудь метод. Объекты-серверы обычно называют «реализацией объектов». Практика показывает, что большин-

ство объектов одновременно исполняют роль и клиентов, и серверов, попеременно вызывая методы на других объектах и отвечая на вызове извне. Имеется возможность, используя CORBA, строить гораздо более гибкие системы, чем системы клиент-сервер, основанные на двух- и трехуровневой архитектуре.

Достоинства CORBA:

- платформенная независимость;
- языковая независимость;
- динамические вызовы;
- динамическое обнаружение объектов;
- масштабируемость;
- CORBA-сервисы;
- широкая индустриальная поддержка.

Недостатки CORBA:

- нет передачи параметров «по значению»;
- отсутствует динамическая загрузка компонент-переходников;
- нет именования через URL.

К основным достоинствам CORBA можно отнести межъязыковую и межплатформенную поддержку. Хотя CORBA-сервисы и отнесены к достоинствам технологии CORBA, их в равной степени можно одновременно отнести и к недостаткам CORBA, ввиду практически полного отсутствия их реализации.

Среда Internet

Интернет — бурно разросшаяся совокупность компьютерных сетей, опутывающих земной шар, связывающих правительственные, военные, образовательные и коммерческие институты, а также отдельных граждан.

Как и многие другие великие идеи, Сеть сетей возникла из проекта, который предназначался совершенно для других целей: из сети ARPAnet, разработанной и созданной в 1969 г. по заказу Агентства передовых исследовательских проектов (ARPA — Advanced Research Project Agency) Министерства обороны США. ARPAnet была сетью, объединяющей учебные заведения, военных и военных подрядчиков; она была создана для помощи исследо-

вателям в обмене информацией, а также (что было одной из главных целей) для изучения, каким образом поддерживать связь в случае ядерного нападения.

В модели ARPAnet между компьютером-источником и компьютером-адресатом всегда существует связь. Сама сеть считается ненадежной; любой ее отрезок может в любой момент исчезнуть (после бомбежки или в результате неисправности на кабеле). Сеть была построена так, чтобы потребность в информации от компьютеров-клиентов была минимальной. Для пересылки сообщения по сети компьютер должен был просто помещать данные в конверт, называемый «пакетом межсетевого протокола» (IP, Internet Protocol), правильно «адресовать» такие пакеты. Взаимодействующие между собой компьютеры (а не только сама сеть) также несли ответственность за обеспечение передачи данных. Основополагающий принцип заключался в том, что каждый компьютер в сети мог общаться в качестве узла с любым другим компьютером с широким выбором компьютерных услуг, ресурсов, информации. Комплекс сетевых соглашений и общедоступных инструментов Сети сетей разработан с целью создания одной большой сети, в которой компьютеры, соединенные воедино, взаимодействуют, имея множество различных программных и аппаратных платформ.

В настоящее время направление развития Интернета в основном определяет Общество Интернета, или ISOC (Internet Society). ISOC — это организация на общественных началах, целью которой является содействие глобальному информационному обмену через Интернет. Она назначает совет старейшин IAB (Internet Architecture Board), который отвечает за техническое руководство и ориентацию Интернета (в основном это стандартизация и адресация в Интернете). Пользователи Интернета выражают свои мнения на заседаниях инженерной комиссии IETF (Internet Engineering Task Force). IETF — еще один общественный орган; он собирается регулярно для обсуждения текущих технических и организационных проблем Интернета.

Финансовая основа Интернета заключается в том, что каждый платит за свою часть. Представители отдельных сетей собираются и решают, как соединяться и как финансировать эти взаимные соединения. Учебное заведение или коммерческое объединение платят за подключение к региональной сети, которая, в свою очередь, платит за доступ к Интернету поставщику на уровне государства. Таким образом, каждое подключение к Интернету кем-то оплачивается.

Рассмотрим кратко основные компоненты Интернета.

World Wide Web (WWW, просто Web, Всемирная паутина) представляет совокупность веб-серверов, на которых хранятся данные, реализованные в виде текстовых и/или графических страниц с гипертекстовыми ссылками на другие страницы или веб-серверы. Если ссылка заинтересовала пользователя, то он может перейти на нужную страницу независимо от ее местонахождения, вернуться на предыдущую просмотренную, поставить закладку. В этом заключается основное преимущество WWW — пользователя не интересует, как организовано и где находится огромное структурированное хранилище данных. Графическое представление подключения различных серверов представляет собой сложную невидимую электронную паутину.

Веб-серверы — специальные компьютеры, осуществляющие хранение страниц с информацией и обработку запросов от других машин. Пользователь, попадая на какой-нибудь веб-сервер, получает страницу с данными. На компьютере пользователя специальная программа (браузер) преобразует полученный документ в удобный для просмотра и чтения вид, отображаемый на экране. Веб-серверы устанавливаются, как правило, в фирмах и организациях, желающих распространить свою информацию среди многих пользователей, и отличаются специфичностью информации. Организация и сопровождение собственного сервера требует значительных затрат. Поэтому в WWW встречаются «разделяемые» (shared) серверы, на которых публикуют свои данные различные пользователи и организации. Это самый дешевый способ

опубликования своей информации для обозрения. Такие серверы зачастую представляют своеобразные информационные свалки.

Серверы FTP представляют собой хранилища различных заархивированных файлов и программ. На этих серверах может храниться как полезная информация (дешевые условно-бесплатные утилиты, программы, картинки), так и информация сомнительного характера, например порнографическая.

Электронная почта является неотъемлемой частью Интернета и одной из самых полезных вещей. С ее помощью можно посылать и получать любую корреспонденцию (письма, статьи, деловые бумаги и др.). Время пересылки зависит от объема, обычно занимает минуты, иногда часы. Каждый абонент электронной почты имеет свой уникальный адрес. Надо отметить, что подключение к электронной почте может быть организовано и без подключения к Интернету. Необходимый интерфейс пользователя реализуется с помощью браузера, который получив от него запрос с Интернет-адресом, преобразовывает его в электронный формат и посылает на определенный сервер. В случае корректности запроса он достигает вебсервера, и последний посылает пользователю в ответ информацию, хранящуюся по заданному адресу. Браузер, получив информацию, делает ее читабельной и отображает на экране. Современные браузеры имеют также встроенную программу для электронной почты.

Подсоединение к Интернету для каждого конкретного пользователя может быть реализовано различными способами: от полного подсоединения по локальной вычислительной сети (ЛВС) до доступа к другому компьютеру для работы с разделением и использованием программного пакета эмуляции терминала.

Диапазон предлагаемых Интернетом услуг достаточно широк. Можно воспользоваться: электронной почтой, электронными досками объявлений, пересылкой файлов, удаленным доступом, каталогизирующими программами и т. д. Для получения полного набора услуг у пользователя должно быть подсоединение по протоколу TCP/IP. Это

необходимо для того, чтобы компьютер пользователя был частью сети и мог устанавливать контакт с любой сервисной программой, имеющейся в Интернете.

Фактически выход в Интернет может быть реализован несколькими видами подключений:

- доступ по выделенному каналу;
- доступ по ISDN (Integrated Services Digital Network — цифровая сеть с интегрированными услугами);
- доступ по коммутируемым линиям;
- с использованием протоколов SLIP и PPP.

Корпорациям и крупным организациям лучше всего использовать доступ по выделенному каналу. В этом случае предоставляется возможность наиболее полно использовать все средства Интернета. Поставщик сетевых услуг при этом сдает в аренду выделенную телефонную линию с указанной скоростью передачи и устанавливает специальный компьютер-маршрутизатор для приема и передачи сообщений от телекоммуникационного узла организации. Это дорогостоящее подключение. Однако, установив такое соединение, каждый компьютер ЛВС организации является полноценным членом Internet и может выполнять любую сетевую функцию.

ISDN — это использование цифровой телефонной линии, соединяющей домашний компьютер или офис с коммутатором телефонной компании. Преимущество ISDN в том, что предоставляется возможность доступа с очень высокими скоростями при относительно низкой стоимости. При этом сервис Интернета предоставляется таким же, как и по коммутируемым линиям. Услуги телефонных компаний, предоставляющих сервис ISDN, доступны не на всей территории России.

Доступ по коммутируемым линиям — наиболее простой и дешевый способ получения доступа к сети (Dial-up Access). В этом случае пользователь приобретает права доступа к компьютеру, который подсоединен к Интернету (хост-компьютеру или узлу Интернета). Войдя по телефонной линии (при этом используется модем и программное обеспечение для работы в коммутируемом режиме) с помощью эмулятора терминала в удаленную систему,

необходимо в ней зарегистрироваться и далее уже можно пользоваться всеми ресурсами Интернета, предоставленными удаленной системе. Пользователь в таком режиме арендует дисковое пространство и вычислительные ресурсы удаленной системы. Если требуется сохранить важное сообщение электронной почты или другие данные, то это можно сделать в удаленной системе, но не на диске пользовательского компьютера: сначала нужно записать файл на диск удаленной системы, а затем с помощью программы передачи данных перенести этот файл на свой компьютер. При таком доступе пользователь не может работать с прикладными программами, для которых нужен графический дисплей, так как в такой конфигурации компьютер, подсоединенный к Интернету, не имеет возможности передать графическую информацию на компьютер пользователя.

При дополнительных финансовых затратах и в коммутируемом режиме можно получить полный доступ к Интернету. Это достигается применением протоколов SLIP и PPP. Один называется межсетевой протокол последовательного канала (Serial Line Internet Protocol, SLIP), а другой — протокол точка — точка (Point-to-Point Protocol, PPP). Одно из главных достоинств SLIP и PPP состоит в том, что они обеспечивают полноценное соединение с Интернетом. Пользовательский компьютер не использует какую-то систему как «точку доступа», а непосредственно подключается к Интернету. Но для подключения средних и больших сетей к Интернету эти протоколы не подходят, поскольку их быстродействия недостаточно для одновременной связи со многими пользователями.

Современные сети создаются по многоуровневому принципу. Передача сообщений в виде последовательности битов начинается на уровне линий связи и аппаратуры, причем линий связи не всегда высокого качества. Затем добавляется уровень базового программного обеспечения, управляющего работой аппаратуры. Следующий уровень программного обеспечения позволяет наделить базовые программные средства дополнительными необ-

ходимыми возможностями. Расширение необходимых функциональных возможностей сети путем добавления уровня за уровнем приводит к тому, что пользователь в конце концов получает по-настоящему дружественный и полезный инструментарий.

Аналогом информационной модели Интернета можно назвать почтовое ведомство, представляющее собой сеть с коммутацией пакетов. Там корреспонденция конкретного пользователя смешивается с другими письмами, отправляется в ближайшее почтовое отделение, где сортируется и направляется в другие почтовые отделения до тех пор, пока не достигнет адресата.

Для передачи данных в Интернете используются Интернет-протокол (IP) и протокол управления передачей (TCP).

С помощью Интернет-протокола (IP) обеспечивается доставка данных из одного пункта в другой. Различные участки Интернета связываются с помощью системы компьютеров (называемых маршрутизаторами), соединяющих между собой сети. Это могут быть сети Ethernet, сети с маркерным доступом, телефонные линии. Правила, по которым информация переходит из одной сети в другую, называются протоколами. Межсетевой протокол (Internet Protocol, IP) отвечает за адресацию, т. е. гарантирует, что маршрутизатор знает, что делать с данными пользователя, когда они поступят. Некоторая адресная информация приводится в начале каждого пользовательского сообщения. Она дает сети достаточно сведений для доставки пакета данных, так как каждый компьютер в Интернете имеет свой уникальный адрес.

Для более надежной передачи больших объемов информации служит протокол управления передачей (Transmission Control Protocol, TCP). Информацию, которую пользователь хочет передать, TCP разбивает на порции. Каждая порция нумеруется, у нее подсчитывается контрольная сумма, чтобы можно было на приемной стороне проверить, вся ли информация получена правильно, а также расположить данные в правильном порядке. На каждую порцию добавляется информация протокола IP,

таким образом получается пакет данных в Интернете, составленный по правилам TCP/IP.

По мере развития Интернета и увеличения числа компьютерных узлов, сортирующих информацию, в сети была разработана доменная система имен — DNS и способ адресации — способ адресации по доменному принципу. DNS иногда еще называют региональной системой наименований.

Доменная система имен — это метод назначения имен путем передачи сетевым группам ответственности за их подмножество имен. Каждый уровень этой системы называется доменом. Домены в именах отделяются друг от друга точками: inr.msk.su. В имени может быть различное количество доменов, но практически их не больше пяти. По мере движения по доменам слева направо в имени количество имен, входящих в соответствующую группу, возрастает.

Все компьютеры Интернета способны пользоваться доменной системой. Работающий в сети компьютер всегда знает свой собственный сетевой адрес. Когда используется доменное имя, например mx.ihep.su, компьютер преобразовывает его в числовой адрес. Для этого он начинает запрашивать помощь у DNS-серверов. Это узлы, рабочие машины, обладающие соответствующей базой данных, в число обязанностей которых входит обслуживание такого рода запросов. DNS-сервер начинает обработку имени с правого его конца и двигается по нему влево, т. е. сначала производится поиск адреса в самой большой группе (домене), потом постепенно сужает поиск. Но для начала опрашивается на предмет наличия нужной информации местный узел. Если местный сервер адрес не знает, он связывается с корневым сервером. Это сервер, который знает адреса серверов имен высшего уровня (самых правых в имени), здесь это уровень государства (ранга домена su). У него запрашивается адрес компьютера, ответственного за зону su. Местный DNS-сервер связывается с этим более общим сервером и запрашивает у него адрес сервера, ответственного за домен ihep.su. Теперь уже запрашивается этот сервер, и у него выясняется адрес рабочей машины mx.

Дальнейшим развитием Интернета явилась Интранет-технология. *Интранет* представляет собой технологию управления корпоративными коммуникациями, в отличие от Интернета, который является технологией глобальных коммуникаций. В телекоммуникационных технологиях выделяют три уровня реализации: аппаратный, программный и информационный. С этой точки зрения Интранет отличается от Интернета только информационными аспектами, где выделяются три уровня: универсальный язык представления корпоративных знаний, модели представления, фактические знания.

Архитектура Интранета явилась естественным развитием информационных систем: от систем с централизованной архитектурой, через системы «клиент-сервер» к Интранету.

Идея централизованной архитектуры была классически реализована в мейнфреймах, отличительной чертой которых была концентрация вычислительных ресурсов в едином комплексе, где осуществлялось хранение и обработка огромных массивов информации. Достоинства: простота администрирования, защита информации.

С появлением персональных компьютеров появилась возможность переноса части информационной системы непосредственно на рабочее место. Таким образом, возникла необходимость построения распределенной информационной системы. Этим целям соответствует архитектура «клиент-сервер», основанная на модели взаимодействия компьютеров и программ в сети, рассмотренная выше.

Однако системам «клиент-сервер» присущ ряд серьезных недостатков:

- трудность администрирования, вследствие территориальной разобщенности и неоднородности компьютеров на рабочих местах;
- недостаточная степень защиты информации от несанкционированных действий;
- закрытый протокол для общения клиентов и сервера, специфичный для данной информационной системы.

Как следствие указанных недостатков была разработана архитектура систем Интранета, сконцентрировавших

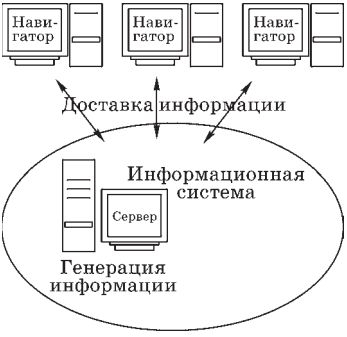


Рис. 3.16
Архитектура систем Интранета

и объединивших в себе лучшие качества централизованных систем и традиционных систем «клиент-сервер» (рис. 3.16).

Вся информационная система находится на центральном компьютере. На рабочих местах находятся простейшие устройства доступа (навигаторы), предоставляющие возможность управления процессами в информационной системе. Все процессы осуществляются на центральной ЭВМ, с которой устройство доступа общается посредством простого протокола, путем передачи экранов и кодов нажатых клавиш на пульте.

Основные достоинства систем Интранета:

- на сервере вырабатывается информация (а не данные) в форме, удобной для представления пользователю;
- использование для обмена информацией между клиентом и сервером протокола открытого типа;
- концентрация прикладной системы на сервере, на клиентах размещается только программа-навигатор;
- облегченное централизованное управление серверной частью и рабочими местами;
- унифицированность интерфейса, не зависящего от программного обеспечения, используемого пользователем (операционная система, СУБД и др.).

Важным преимуществом Интранета является открытость технологии. Существующее программное обеспечение, основанное на закрытых технологиях, когда решения, разработанные одной фирмой для одного приложения, возможно, кажутся более функциональными и удобными, однако резко ограничивают возможности развития информационных систем. В настоящее время в Интранете широко используются открытые стандарты по следующим направлениям:

- управление сетевыми ресурсами (SMTP, IMAP, MIME);
- телеконференции (NNTP);
- информационный сервис (HTPP, HTML);
- справочная служба (LDAP);
- программирование (Java).

Тенденции дальнейшего развития Интранета:

- интеллектуальный сетевой поиск;
- высокая интерактивность навигаторов за счет применения Java-технологии;
- сетевые компьютеры;
- превращение интерфейса навигатора в универсальный интерфейс с компьютером.

3.6. ТЕХНОЛОГИИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Современная информационная система — это набор информационных технологий, направленных на поддержку жизненного цикла информации и включающих три основных процесса: обработку данных, управление информацией и управление знаниями. В условиях резкого увеличения объемов информации переход к работе со знаниями на основе искусственного интеллекта является, по всей вероятности, единственной альтернативой информационного общества.

Воспользуемся определением «интеллектуальной системы» профессора Д. А. Поспелова [19]: «Система называется интеллектуальной, если в ней реализованы следующие основные функции, позвляющие:

- накапливать знания об окружающем систему мире, классифицировать их и оценивать их с точки зрения прагматической полезности и непротиворечивости, инициировать процессы получения новых знаний, осуществлять соотнесение новых знаний с ранее хранимыми;
- пополнять поступившие знания с помощью логического вывода, отражающего закономерности в окружающем систему мире или в накопленных ею ранее знаниях, получать обобщенные знания на основе бо-

лее частных знаний и логически планировать свою деятельность;

- общаться с человеком на языке, максимально приближенном к естественному человеческому языку, и получать информацию от каналов, аналогичных тем, которые использует человек при восприятии окружающего мира, уметь формировать для себя или по просьбе человека (пользователя) объяснение собственной деятельности, оказывать пользователю помощь за счет тех знаний, которые хранятся в памяти, и тех логических средств рассуждений, которые присущи системе».

Перечисленные функции можно назвать функциями представления и обработки знаний, рассуждения и общения. Наряду с обязательными компонентами, в зависимости от решаемых задач и области применения в конкретной системе, эти функции могут быть реализованы в различной степени, что определяет индивидуальность архитектуры. На рисунке 3.17 в наиболее общем виде представлена архитектура интеллектуальной системы (ИнС) в виде совокупности блоков и связей между ними [12].

База знаний представляет собой совокупность сред, хранящих знания различных типов. Рассмотрим кратко их назначение.

База фактов (данных) хранит конкретные данные, а база правил — элементарные выражения, называемые в теории искусственного интеллекта продукциями.

База процедур содержит прикладные программы, с помощью которых выполняются все необходимые преобразования и вычисления над данными. База закономерностей включает различные сведения, относящиеся к особенностям той среды, в которой действует система. База метазнаний (база знаний о себе) содержит описание самой системы и способов ее функционирования: сведения о том, как внутри системы представляются единицы информации различного типа, как взаимодействуют различные компоненты системы, как было получено решение задачи, т. е. она хранит.

Рис. 3.17
Общая структура интеллектуальной системы

База целей содержит целевые структуры, называемые сценарями, позволяющие организовать процессы движения от исходных фактов, правил, процедур к достижению той цели, которая поступила в систему от пользователя либо была сформулирована самой системой в процессе ее деятельности в проблемной среде.

Управление всеми базами, входящими в базу знаний, и организацию их взаимодействия осуществляет система управления базами знаний. С ее же помощью реализуют-

ся связи баз знаний с внешней средой. Таким образом, машина базы знаний осуществляет первую функцию интеллектуальной системы.

Выполнение второй функции обеспечивает часть интеллектуальной системы, называемая решателем и состоящая из ряда блоков, управляемых системой управления решателя. Часть из блоков реализует логический вывод. Блок дедуктивного вывода осуществляет в решателе дедуктивные рассуждения, с помощью которых из закономерностей из базы знаний, фактов из базы фактов и правил из базы правил выводятся новые факты.

Кроме этого, данный блок реализует эвристические процедуры поиска решений задач как поиск путей решения задачи по сценариям при заданной конечной цели. Для реализации рассуждений, которые не носят дедуктивного характера, т. е. поиск по аналогии, по прецеденту и пр., используются блоки индуктивного и правдоподобного выводов. Блок планирования используется в задачах планирования решений совместно с блоком дедуктивного вывода. Назначение блока функциональных преобразований состоит в решении задач расчетно-логического и алгоритмического типа.

Третья функция — функция общения — реализуется как с помощью компоненты естественно-языкового интерфейса, так и с помощью рецепторов и эффекторов, которые осуществляют так называемое невербальное общение и используются в интеллектуальных роботах.

Важное место в теории ИИ занимает проблема представления знаний. В настоящее время выделяют следующие основные типы моделей представления знаний:

1) семантические сети, в том числе функциональные семантические сети;

2) фреймы и сети фреймов;

3) продукционные модели.

Семантические сети определяют как граф общего вида, в котором можно выделить множество вершин и ребер. Каждая вершина графа представляет некоторое понятие, а дуга — отношение между парой понятий. Метка и направление дуги конкретизируют семантику. Метки

вершин семантической нагрузки не несут, а используются как справочная информация.

Различные разновидности семантических сетей обладают различной семантической мощностью, следовательно, можно описать одну и ту же предметную область более компактно или громоздко.

Фреймом называют структуру данных для представления и описания стереотипных объектов, событий или ситуаций. Фреймовая модель представления знаний состоит из двух частей:

- набор фреймов, составляющих библиотеку внутри представляемых знаний;
- механизмы их преобразования, связывания и т. д.

Существуют два типа фреймов:

- образец (прототип) — интенсиональное описание некоторого множества экземпляров;
- экземпляр (пример) — экстенсиональное представление фрейм-образца.

В общем виде фрейм может быть представлен следующим кортежем:

<ИФ, (ИС, ЗС, ПП), …, (ИС, ЗС, ПП)>,

где ИФ — имя фрейма; ИС — имя слота; ЗС — значение слота; ПП — имя присоединенной процедуры (необязательный параметр).

Слоты — это некоторые незаполненные подструктуры фрейма, заполнение которых приводит к тому, что данный фрейм ставится в соответствие некоторой ситуации, явлению или объекту.

В качестве данных фрейм может содержать обращения к процедурам (так называемые присоединенные процедуры). Выделяют два вида процедур: процедуры-демоны и процедуры-слуги. Процедуры-демоны активизируются при каждой попытке добавления или удаления данных из слота. Процедуры-слуги активизируются только при выполнении условий, определенных пользователем при создании фрейма.

Продукционные модели — это набор правил вида «условия — действие», где условиями являются утверж-

дения о содержимом базы данных, а действия представляют собой процедуры, которые могут изменять содержимое базы данных.

Формально продукция определяется следующим образом:

$$(i); Q; P; C; A \to B; N,$$

где (i) — имя продукции (правила); Q — сфера применения правила; P — предусловие (например, приоритетность); C — предикат (отношение); $A \to B$ — ядро; N — постусловия (изменения, вносимые в систему правил).

Практически продукции строятся по схеме «ЕСЛИ» (причина или, иначе, посылка), «ТО» (следствие или, иначе, цель правила).

Полученные в результате срабатывания продукций новые знания могут использоваться в следующих целях:

- понимание и интерпретация фактов и правил с использованием продукций, фреймов, семантических цепей;
- решение задач с помощью моделирования;
- идентификация источника данных, причин несовпадений новых знаний со старыми, получение метазнаний;
- составление вопросов к системе;
- усвоение новых знаний, устранение противоречий, систематизация избыточных данных.

Процесс рассмотрения компьютером набора правил (выполнение программы) называют консультацией. Ее наиболее удобная для пользователя форма — дружественный диалог с компьютером. Интерфейс может быть в форме меню, на языке команд и на естественном языке.

Диалог может быть построен на системе вопросов, задаваемых пользователем, компьютером, или фактов — данных, хранящихся в базе данных. Возможен смешанный вариант, когда в базе данных недостаточно фактов.

При прямом поиске пользователь может задавать две группы вопросов, на которые компьютер дает объяснения:

1) КАК получено решение. При этом компьютер должен выдать на экран трассу в виде ссылок на использованные правила;

2) ПОЧЕМУ компьютер задал какой-то вопрос, при этом на экран выдается своеобразная трасса, которую компьютер хотел бы использовать для вывода после получения ответа на задаваемый вопрос. Вопрос ПОЧЕМУ может быть задан как в процессе консультации, так и после выполнения программы.

Специфичен алгоритм поиска, реализуемый логическими языками: он является фактически последовательным перебором по дереву «сверху — вниз — слева — направо».

Сравнительные характеристики методов описания знаний представлены в таблице 3.3 [12], [20].

Таблица 3.3

Сравнительные характерискити метода описания знаний

Метод	Достоинства	Недостатки
Семантические сети	Возможность представления смысла фраз и ограничений. Наличие внутренней структуры. Определение операций над объектами (естественный язык)	Нет средств зависимости от времени. Произвольность структуры. Сложность сочетания с методами логики
Фреймы	Наличие внутренней структуры и связности	Жесткость структуры. Трудности построения модулей и модификации. Сложность сочетания с методами логики
Продукции	Модульность. Простота модификации правил. Отделение предметных правил от управления. Простота сочетания с методами логики	Отсутствие внутренней структуры. Зависимость шагов вывода от принятой стратегии вывода

Из приведенных структур методов описания знаний нетрудно заметить, что построить таблицу 3.3, позволяющую наглядно увидеть правила и обнаружить ошибки в их формировании, удобнее всего для продукций. К тому же продукции легко трансформируются при изменении

правил. Перечисленные факторы определили более широкое использование продукций.

Помимо описания знаний, важное значение имеет теория логического вывода.

Выделяют следующие методы логики вывода:

1) традиционная логика;

2) булева алгебра (алгебра логики);

3) исчисление предикатов.

Традиционная логика возникла еще во времена Аристотеля и была предназначена для уменьшения количества логических ошибок в различного рода диспутах и спорах. Отправной точкой этой логики служат следующие понятия.

Алгебра логики (*булева алгебра*) оперирует с высказываниями. Чаще всего используют одно- $y = f(x)$ и двуместные $z = f(x, y)$ высказывания.

Предикаты первого порядка. Пусть имеется n исходных множеств X, что запишем как X^n.

Если осуществляется преобразование: $X^n \to X$, то говорят об *алгебре*.

Если осуществляется преобразование: $X^n \to \{0, 1\}$, то говорят об *отношениях*.

Исчисление — совокупность правил оперирования с каким-либо символом (предикатами).

Предикат (сказуемое) — отношение между множествами элементов соответствующих аргументов. Например: $Q(a_1, a_2, \ldots, a_n)$, где a_n — аргументы; Q — предикат.

Предикаты первого порядка «наследуют» кванторы от традиционной логики и строгую логику алгебры логики. Предикат является предикатом первого порядка, *если квантор не входит в сам предикат*. Предикаты первого порядка охватывают большую часть отношений. Доказано, что в ряде частных случаев предикаты более высокого порядка могут быть сведены к предикатам первого порядка. Хотя предикаты первого порядка не позволяют описывать возможность и необходимость, убеждения, намерения, цели, метазнания (знания о знаниях), они охватывают широкий спектр возможностей и получили широкое распространение в построении ЭС.

Для любой системы выделяют процесс проектирования и эксплуатации. Процесс проектирования по своему назначению является интеллектуальным и включает формирование цели, технологию и инструментарий. В проектировании присутствует много ручных операций, хотя в последнее время все шире используются системы автоматизированного проектирования.

Когда говорят об интеллектуальных системах, то имеют в виду интеллектуальность, проявляющуюся в процессе эксплуатации системы. Общая схема ИнС представлена на рисунке 3.18.

Блок «интерфейс пользователя» предназначен для связи компьютера с пользователем, для которого предпочтительным языком «разговора» является естественный язык или близкий к нему.

Следует отметить специфическое понятие «естественный язык», под которым понимается фактически искус-

Рис. 3.18
Схема интеллектуальной системы:
МЛВ — машина логического вывода.

ственный язык, полученный на этапе концептуализации из естественного человеческого языка (русского, английского или любого другого) путем удаления неоднозначностей (синонимов, омонимов, идиоматических выражений). Следует заметить, что такой язык в теории автоматизированных систем носит название «искусственный язык» или «информационный язык».

Основу ИС составляют блоки «база данных — БД», «база правил — БП», «машина логического вывода — МЛВ», «объяснения». БД хранит исходные данные. В базе правил фиксируются знания и опыт эксперта. МЛВ выводит результат, взаимодействуя с БД и базой правил. Объяснения выводят систему правил, использованных для получения конкретного результата, на естественном языке.

Все три «компьютерных» блока должны быть описаны математически. Правила записываются через интерфейс эксперта (рис. 3.18) в виде сложных правильно построенных формул (ППФ) при посредничестве инженера по знаниям. Форма, в которой записываются ППФ, определяется *синтаксисом*, а истинностное значение алгоритмов логического вывода — *семантикой* или смыслом.

По запросу пользователя компьютер с помощью блока «объяснения» может дать ответ на вопрос КАК (с помощью набора каких правил) получен результат и ПОЧЕМУ компьютер задает пользователю уточняющие вопросы, связанные, как правило, с данными.

Для стыковки интерфейса пользователя, работающего на естественном языке, с перечисленными блоками используют лингвистический и лексический процессоры.

Лингвистический процессор непосредственно работает с естественным языком, преобразуя результат в «машинный вид».

Лексический процессор — словарный состав информационного языка.

Иногда включают и синтаксический процессор, при этом под *синтаксисом* понимают правила сочетания слов внутри предложения и построения предложений.

В блоке «Получение знаний» выделяют два понятия:

- если информация поступает из книг (документов), то говорят о *выявлении* знаний;
- если она получается на основе работы эксперта, то говорят об *извлечении* знаний.

Преобразование полученной информации инженером по знаниям называется *получением знаний*.

Сложность здесь заключается в том, что эксперт может не владеть языками программирования, в то время как инженер по знаниям может недостаточно ориентироваться в данной предметной области.

При этом в наиболее развитых моделях выделяют интенсиональную и экстенсиональную компоненты.

Экстенсиональная — различные сведения о предметной области (данные).

Интенсиональная — схемы связей между атрибутами, отражающие основные закономерности системы в предметной области решаемых функциональных задач. Для разработчиков моделей данные — это схема БД. Для представителей ИИ — это знания о проблемной области.

Понятие «интеллект» многогранно и потому существует значительное количество разновидностей интеллектуальных систем, которые целесообразно классифицировать.

Классификация — разделение множества на подмножества по принятому классификационному признаку.

Из множеств классификаций интеллектуальных систем воспользуемся классификацией, показанной на рисунке 3.19.

Системы на естественном языке (СЕЯ) специфичны и предназначены преимущественно для таких целей, как машинный перевод, генерация документов, автоматическое аннотирование и реферирование.

Экспертные системы (ЭС) предполагают высокую степень формализации процессов на этапе концептуализации.

Разновидностью экспертных систем можно считать расчетно-логические системы (РЛС), оперирующие функциями вместо правил.

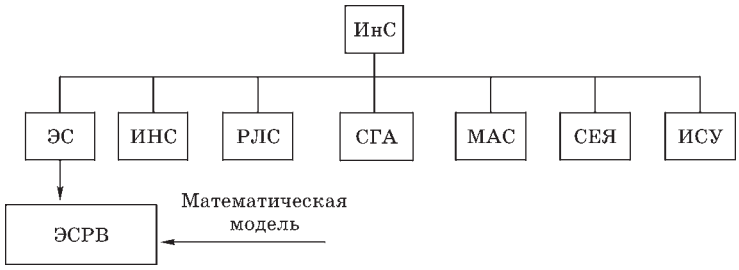


Рис. 3.19
Классификация интеллектуальных систем:

ЭС — экспертные системы; ИНС — искусственные нейронные сети; РЛС — расчетно-логические системы; СГА — системы с генетическими алгоритмами; МАС — многоагентные системы; СЕЯ — системы на естественном языке; ИСУ — интеллектуальные системы управления; ЭСРВ — экспертные системы реального времени.

Искусственные нейронные сети (ИНС) представляют собой фактически разновидность систем автоматического управления, использующие свойства нейрона.

Системы с генетическими алгоритмами (СГА) относятся к разновидности эволюционных эвристических методов.

Многоагентные системы (МАС) преследуют цель согласования теорий баз данных и баз знаний.

Интеллектуальные системы управления (ИСУ) используют режим адаптации к изменяющимся параметрам и целям эксплуатации (функционирования) систем.

По характеру математического описания интеллектуальные системы возможно разделить на непрерывные и дискретные (рис. 3.20).

Непрерывные процессы имеют место в ИНС, ИСУ, РЛС и описываются дифференциальными уравнениями с соответствующими критериями, функциональными (формульными) зависимостями. В этом случае используются преимущественно числовые данные, хотя они могут быть искусственно организованы в виде такой, например, структуры, как базы данных.

Интеллектуальные процессы человека чаще связаны с дискретными данными и знаниями, опирающимися на информационные языки.

В связи с этим ИнС оперируют с дискретными величинами (классы ЭС, СГА, МАС, СЕЯ). Однако и в этих

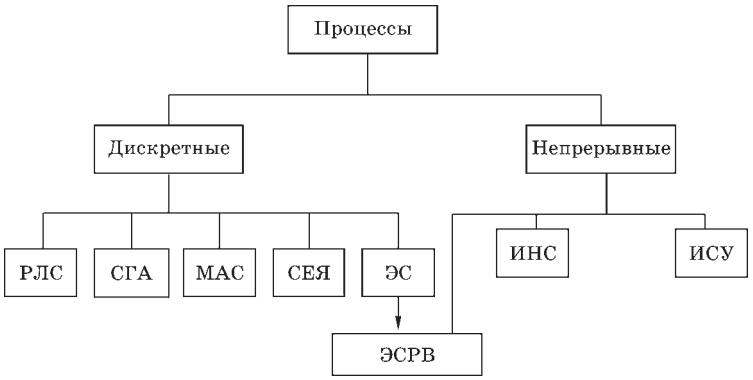


Рис. 3.20
Методы математического описания

классах не существует однородного математического описания. Здесь используются такие методы, как логическое описание, нечеткие множества, функциональные семантические сети, контекстно зависимые грамматики.

Имеется к тому же все большая необходимость связи дискретных и непрерывных величин, что привело к появлению гибридных моделей. Сюда относятся прежде всего экспертные системы реального времени (ЭСРВ), в которых имеется непрерывная составляющая, описываемая дифференциальными или разностными уравнениями.

Экспертная система (статическая) — вычислительная система, использующая знания эксперта и процедуры логического вывода и позволяющая дать объяснения полученным результатам. Синонимом ЭС является термин «система, базирующаяся на знаниях». *Знания* — совокупность данных и связывающих их правил.

На начальных этапах развития рассматривались статические ЭС. В таких системах время напрямую не фигурировало.

Изначально предполагалось, что схема ЭС будет иметь вид, показанный на рисунке 3.18. Однако выяснилось, особенно после появления оболочки экспертной системы GURU, что построение «переводчика» с естественного языка на язык компьютера является сложной, вполне са-

мостоятельной проблемой. Появление декларативных логических языков типа ПРОЛОГ (Prolog), а затем — языков, использующих ПРОЛОГ-идеи, позволило упростить задачу «перевода» при условии использования текстовых элементов из заранее составленного естественного языка после выполнения этапа концептуализации. При этом структура интеллектуальной системы трансформировалась — исчезли блоки лингвистических и лексических процессоров (рис. 3.21).

Статическая ЭС без блока «база правил» называется *оболочкой ЭС* (аналог СУБД). Наиболее известная оболочка GURU. В рамках языка ПРОЛОГ также имеется оболочка ЭС.

В то же время системы управления имеют, как правило, замкнутую структуру.

Очень скоро выяснилось, что статические системы имеют ограниченную сферу применения.

Рис. 3.21
Схема экспертной системы:
МЛВ — машина логического вывода.

Статические ЭС не позволяют учесть, сохранить и провести анализ изменяющихся во времени данных, поступающих от внешних источников; работать с асинхронными процессами; обеспечить механизм рассуждений при ограниченных ресурсах; обеспечить прогноз последствий принимаемых решений-советов; обеспечить создание и поддержку интерфейсов пользователей, уровень их защиты для различных категорий пользователей.

Реализация ЭС в области управления называется ЭС реального времени (ЭСРВ).

Следует отметить, что первоначально ЭСРВ работала с моделью объекта управления (рис. 3.18). В настоящее время в ЭСРВ включают реальный объект управления, информация о состоянии которого поставляется системой датчиков и отражается на табло.

Искусственные нейронные сети. Начиная с 1990-х гг. работы велись в плане поисков методов обучения и построения адаптивных систем. На определенном этапе становления искусственные нейронные сети (ИНС) развивались в противовес ЭС реального времени с целью устранения следующих недостатков:

- сложное сочетание непрерывных и дискретных процессов;
- сложность учета нелинейного характера процесса управления;
- недостаточная адаптация ЭС реального времени к различного рода неопределенностям (параметров структуры).

В дальнейшем выяснилось, что сами ИНС имеют достаточно сложную структуру. Их применение для малоразмерных систем управления проблематично.

Основой ИНС является базовый процессорный элемент, который построен как имитация естественному биологическому нейрону.

Системы с генетическими алгоритмами появились в процессе поиска новых методов математических расчетов, поскольку возможности как детерминированных, так и статистических методов оказались недостаточными для описания вновь появляющихся классов систем.

В методах эффективного поиска выделяются три группы:

1) методы, основанные на математических вычислениях, — это все виды программирования. Серьезным ограничением здесь являются требования дифференцируемости функции. Если это требование не соблюдено, тогда используются специальные методы программирования. Сложным в общем случае может оказаться многоэкстремальный характер кривой выбора решения;

2) перечислительные или переборные методы (целочисленное, динамическое программирование) характеризуются высокой размерностью задач, большим объемом вычислений;

3) методы, использующие элемент случайности (случайный поиск), не дают гарантии оптимальности решений и получения более хорошего результата, чем в предыдущих методах.

В целом ряде случаев нужно не оптимальное, а более хорошее решение. Тогда оказалось полезным сочетать детерминированную составляющую алгоритмов со случайной составляющей. Это направление получило название *эволюционные вычисления*.

Методы эволюционных вычислений возможно классифицировать следующим образом:

- эволюционное программирование базируется на теории конечных автоматов, которые должны реагировать на изменения внешней среды;
- эволюционные стратегии по своей сути похожи на метод генетических алгоритмов, однако целевая функция может меняться от поколения к поколению (основной признак изменения — мутация).

Общая цель названных методов — адаптация к изменяющимся условиям. В то же время общая цель генетических алгоритмов — описание процесса развития популяций.

В генетических алгоритмах используются приемы из генетики. Процесс заменяется некоторым кодом. Используется специфическая терминология.

Код и его структура называются *генотипом*. Конкретная интерпретация кода, т. е. переход от задачи к коду,

называется *фенотипом*. Набор конкретных кодов — *популяцией*. Индивидуальный код называют *хромосомой* (особью, индивидом). Каждый шаг в алгоритме называют *поколением*. Особи предыдущего поколения — *родители*, последующего поколения — *потомки*.

ГА является вариантом решения в виде битовой строки фиксированной длины, манипулирование с которой производится в отсутствие связей с ее смысловой интерпретацией.

Теория *мультиагентных систем* только начала складываться. Показанные на рисунке 3.22 предметные области развивались практически автономно.

Область систем искусственного интеллекта (с позиции принятия решений) сильно ограничена. При этом рассматриваются дискретные процессы. В то же время появились адаптивные непрерывные системы.

Адаптация возможна к изменению параметров объекта управления (самонастраивающаяся система) к изменению структуры ОУ (самонастраивающаяся система).

Возникла задача интеграции предметных областей, характеризующихся кривой *1* на рисунке 3.22. Напря-

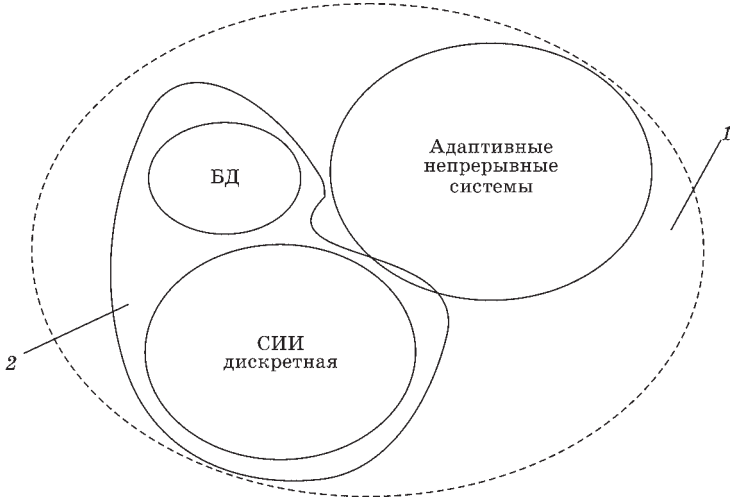


Рис. 3.22
Соотношение предметных областей

мую решить такую задачу оказалось сложно. Поэтому пока изучают интеграцию предметных областей, ограниченных кривой *2*.

Считается, что эта интеграция является началом решения первого понимания термина «искусственный интеллект». Одним из таких методов интеграции является метод *мультиагентных* (*многоагентных*) *систем* (*МАС*) [12].

В мультиагентных системах используют модальную логику — логику достоверности. Достоверность может иметь, как отмечалось, три разновидности:

- аподиктическая (наибольшая достоверность) — целое больше части;
- ассерторическая (утвердительная логика) — типа яблоко разделено на части;
- проблематичная (фактор уверенности, нечетные переменные).

Термин «агент» происходит от латинского глагола agere, что означает «действовать», «двигать», «править», «управлять».

С компьютерной точки зрения *агент* — вычислительный процесс, который реализует автономную коммуникационную функциональность приложений. С точки зрения систем искусственного интеллекта целесообразно иметь интеллектуального агента.

Отталкиваясь от определений данных, будем считать, что ИА — это программный или аппаратный объект (сущность), автономно функционирующий для достижения целей, поставленных перед ним владельцем или пользователем, и обладающий определенными интеллектуальными способностями. Уровень этих способностей, необходимых для достижения поставленных перед ИА целей, можно определить, пользуясь классификацией Д. А. Поспелова.

Вид математического аппарата, позволяющего описать поведение агента в соответствующей среде, и является мерой его интеллектуальной сложности (или разумности).

Представим эту классификацию в виде таблицы 3.4.

Таблица 3.4

Мера интеллектуальной разумности

Тип среды функционирования	Метод математического описания	Вид ИА
Замкнутая детерминированная	Автоматные грамматики, конечные автоматы	Автоматные агенты
Замкнутая вероятностная	Вероятностные автоматы	Вероятностные автоматные агенты
Нетрансформируемая открытая	Контекстно-свободные грамматики (КСГ), сценарии, магазинные автоматы	КСГ-агенты
Трансформируемая замкнутая	Контекстно-зависимые грамматики (линейные автоматы)	КЗГ-агенты
Трансформируемая открытая	Семиотические системы	Семиотические агенты

Две базовые характеристики — автономность и целенаправленность — позволяют отличать ИА от других программных и аппаратных объектов (модулей, подпрограмм, процедур). Наличие целесообразности поведения требует, чтобы ИА обладал свойством реактивности. Такой уровень интеллекта соответствует рефлекторному поведению животного. Если же ИА обладает знаниями о среде, собственных целях и способах их достижения, то такой агент может быть назван разумным. Следовательно, может быть проведена граница между интеллектуальными и неинтеллектуальными агентами.

Значительные усилия по стандартизации агентных систем и технологий предпринимает организация Foundation for Intelligent Physical Agents (FIPA), которая является международной организацией, выполняющей работу по разработке открытых спецификаций, поддерживающих интероперабельность агентов и агентных приложений.

МАС может рассматриваться как сильно связанная сеть решателей, совместно работающих над проблемами, которые могут выходить за рамки возможностей индивидуальных агентов.

Исходя из изложенного, МАС — совокупность взаимосвязанных агентов, как программных, так и аппа-

ратных, способных взаимодействовать друг с другом и окружающей средой, обладающих определенными интеллектуальными способностями и возможностью индивидуальных и совместных действий.

Интеллектуальными МАС будем называть МАС, обладающие всеми признаками ИНС в соответствии с определением К. А. Пупкова.

Открытые МАС могут содержать переменные множества агентов, входящих в систему и выходящих из нее, причем возможно своекорыстное (столкновение интересов) поведение агентов.

Понятие «агентно-ориентированная система» (АОС) представляется более широким, чем понятие МАС. M. Wooldridge характеризует АОС как систему, в которой ключевой используемой абстракцией является агент.

Агентно-ориентированные системы трактуем как гибридные системы, содержащие наряду с МАС и другие системы (ЭС, обучающие и тестирующие системы, распределенные объектные приложения).

АОС в настоящее время являются глобально распределенными взаимосвязанными совокупностями программных и аппаратных систем, содержащими сотни и тысячи компонентов, что и определяет возрастающую сложность их разработки, внедрения и эксплуатации.

Системы на естественном языке (СЕЯ) реализуют процедуры поиска, автоматического аннотирования и реферирования. В свою очередь, в поисковой системе следует выделить ручной режим, характерный для библиотек, и автоматизированный режим для электронных документов. Правда, в современных библиотеках все шире используется автоматизированный поиск.

Понятие интеллектуальной системы управления (*ИСУ*) связано с современными адаптивными автоматизированными системами управления. Адаптивные системы отличаются от неадаптивных тем, что они в процессе функционирования получают дополнительную информацию о поведении или влиянии среды на параметры, структуру и цели системы и компенсируют это влияние.

3.7. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Развитие технологии программирования происходит на фоне возрастающих требований к качеству ПО, что определяется увеличением сложности и ответственности выполняемых им задач. Если на начальном этапе создание качественного ПО во многом зависело от опыта и квалификации разработчика, то на современном этапе акцент смещается в сторону инструмента, которым он пользуется, и средств автоматизации.

На протяжении развития технологий программирования разработчики стремились к тому, чтобы по некоторому неформальному описанию объекта программирования автоматически генерировался бы текст синтаксически и семантически корректной программы. Однако на данном этапе процесс далек от такого идеального варианта.

В настоящее время любые технологии разработки ПО представляют процесс, состоящий из ряда последовательных преобразований одного описания решаемой задачи в другое, начиная от постановки задачи и заканчивая программой, реализованной в кодах конкретной ЭВМ.

Рассмотрим краткую ретроспективу развития методов и средств разработки ПО [21], [22].

Установка ключевых переключателей на передней панели вычислительных устройств — предпосылка первых программ.

Машинный язык — возможность задавать команды, оперируя с ячейками памяти, полностью используя возможности компьютера. Однако использование большинства ЭВМ на уровне машинного языка затруднительно, особенно это касается ввода-вывода. Поэтому от его использования пришлось отказаться.

Специализированные, или *проблемно- (предметно)-ориентированные*, *языки* — построены для решения конкретных задач, упрощают процесс разработки программных продуктов в выбранной предметной области, но в то же время для новых языков характерна проблема преемственности, а для специализированных — ограниченности области применения. На этом этапе следует выделить

развитие таких языков программирования, как КОБОЛ, СИМУЛА, СИМСКРИПТ, АСПИД, АНАЛИТИК и т. д.

Постепенное усложнение решаемых задач, а вместе с ним и кодов и объемов разрабатываемых программных средств, заставило критически переосмыслить сложившиеся к концу 1960-х гг. стиль и технику программирования. В связи с чем, начиная с 1970-х гг. усилиями рабочей группы Технического комитета по программированию Международной федерации по обработке информации (ИФИП), формируются основы методологии и теории программирования и утверждается *принцип структурного программирования* как наиболее производительного и привлекательного стиля программирования. Развитие концепции *структуризации* привело к осознанию необходимости структуризации данных и определило появление в языках программирования *механизмов абстрагирования типов данных* (Клу, Модула и др.). Последнее легло в основу *техники модульного программирования*, содержательной основой которой является интерпретация типа как алгебры над множеством объектов и множеством операций, а модуля как программного эквивалента абстрактного типа. В модульном программировании акцент делается на разбиение программы на модули таким образом, чтобы данные (обрабатываемые модулем) были спрятаны в нем. Эта доктрина, известная как «принцип ограничения доступа к данным», в значительной степени повысила модифицируемость и эффективность порождаемого кода.

Развитие техники модульного программирования привело к появлению *объектно-ориентированного стиля программирования*, который во многом унифицировал процесс создания ПО. К достоинствам этого метода относится:

- более полная реализация технологии структурного программирования;
- упрощение процесса создания сложных иерархических систем;
- возможность создания пользовательских библиотек объектов в различных областях применения.

Параллельно с развитием процедурного стиля программирования в начале 1970-х гг. появляются непроцедурные *языки логического программирования и программирования искусственного интеллекта* (LISP, PROLOG, РЕФАЛ, ПРИЗ). Программа в таких языках представляет собой совокупность *правил* (определяющих отношения между объектами) и *цели* (запроса). Процесс выполнения программы трактуется как процесс установления общезначимости логической формулы по правилам, установленным семантикой того или иного языка. Результат вычислений является побочным продуктом процедуры вывода. Такой метод являет собой полную противоположность программирования на каком-либо из процедурных языков. Сегодня PROLOG — язык, предназначенный для программирования приложений, использующих средства и методы искусственного интеллекта, создания экспертных систем и представления знаний.

Программный продукт является результатом некоего технологического процесса, включающего следующие основные этапы:

- спецификация;
- проектирование;
- кодирование;
- отладка;
- документирование;
- тестирование;
- сопровождение;
- реинжиниринг.

Спецификация

Формирование спецификации предназначено для определения сервисов, которыми будет обладать проектируемое ПО, а также ограничений, накладываемых на функциональные возможности и разработку программной системы. Этот процесс в настоящее время обычно называют разработкой требований (Requirements Engineering). Разработка требований часто является критическим этапом в создании ПО, поскольку ошибки, допущенные на

этом этапе, ведут к возникновению проблем на этапах проектирования и разработки.

Процесс разработки требований включает четыре основных этапа.

1. Предварительные исследования. Оценивается степень удовлетворенности пользователей существующими программными продуктами и аппаратными средствами, а также экономическая эффективность будущей системы и возможность уложиться в существующие бюджетные ограничения при ее разработке. Этот этап должен быть по возможности коротким и дешевым.

2. Формирование и анализ требований. Формируются системные требования путем изучения существующих аналогичных систем, обсуждения будущей системы с потенциальными пользователями и заказчиками, анализа задач, которые должна решать система, и т. п. Этот этап может включать разработку нескольких моделей системы и ее прототипов, что помогает сформировать функциональные требования к системе.

3. Специфицирование требовании. Осуществляется перевод всей совокупности информации, собранной на предыдущем этапе, в документ, определяющий множество требований. Этот документ обычно содержит два типа требований: пользовательские — обобщенные представления заказчиков и конечных пользователей о системе; системные — детальное описание функциональных показателей системы.

4. Утверждение требований. Проверяется выполнимость, согласованность и полнота множества требований. В процессе формирования ограничений неизбежно возникновение каких-либо ошибок. На этом этапе они должны быть по возможности выявлены и устранены.

Проектирование

Целью проектирования является определение внутренних свойств системы и детализации ее внешних (видимых) свойств на основе выданных заказчиком требований к ПО (исходные условия задачи). Эти требования подвергаются анализу. Первоначально программа рас-

сматривается как «черный ящик». Ход процесса проектирования и его результаты зависят не только от состава требований, но и от выбранной модели процесса, опыта проектировщика.

В зависимости от класса создаваемого ПО процесс проектирования может обеспечиваться как «ручным» проектированием, так и различными средствами его автоматизации. Проектированию обычно подлежат:

- архитектура ПО;
- устройство компонентов (модулей) ПО;
- пользовательские интерфейсы.

В ходе проектирования архитектором или опытным программистом создается проектная документация, включающая текстовые описания, диаграммы, модели будущей программы.

Проектирование программного обеспечения — этап жизненного цикла программного обеспечения, во время которого исследуется структура и взаимосвязи элементов разрабатываемой системы. Результатом этого этапа является проект, содержащий достаточное количество информации для реализации системы. Жизненный цикл программного обеспечения — непрерывный процесс, который начинается с принятия решения о необходимости создания ПО и заканчивается при полном изъятии его из эксплуатации.

Результат процесса проектирования — *дизайн*. Рассматриваемое как процесс проектирование есть инженерная деятельность в рамках жизненного цикла (в данном контексте — программного обеспечения), в которой надлежащим образом анализируются требования для создания описания внутренней структуры ПО, являющейся основой для конструирования программного обеспечения как такового. Программный дизайн (как результат деятельности по проектированию) должен описывать архитектуру программного обеспечения, т. е. представлять декомпозицию программной системы в виде организованной структуры компонент и интерфейсов между компонентами. Важнейшей характеристикой готовности дизайна является тот уровень детализации компонентов,

который позволяет заняться их конструированием. Термины «дизайн» и «архитектура» могут использоваться взаимозаменяемым образом, но чаще говорят о дизайне как о целостном взгляде на архитектуру системы.

Проектирование играет важную роль в процессах жизненного цикла создания программного обеспечения (Software Development Life Cycle), например IEEE и ISO/IEC (ГОСТ Р ИСО.МЭК) 12207. Проектирование программных систем можно рассматривать как деятельность, результат которой состоит из двух составных частей:

1) архитектурный или высокоуровневый дизайн (Software Architectural Design, Top-level Design) — описание высокоуровневой структуры и организации компонентов системы;

2) детализированная архитектура (Software Detailed Design) — описывающая каждый компонент в том объеме, который необходим для конструирования.

Кодирование

Это процесс написания программного кода, скриптов, с целью реализации определенного алгоритма на определенном языке программирования.

Разработка программ начинается с постановки технического задания. От того, насколько грамотным и точным оно будет, во многом зависит продуктивный результат. Требования к будущему ПО должны быть тщательно проанализированы и точно сформулированы. На этом этапе главная задача — собрать как можно больше информации, для этого стоит пообщаться и с руководством, и с теми, кто будет непосредственно пользоваться софтом. Итогом этого этапа должно стать техническое задание, детально описывающее действия программного продукта и ожидаемые результаты.

Программирование — процесс и искусство создания компьютерных программ и/или программного обеспечения с помощью языков программирования. Программирование сочетает в себе элементы искусства, фундаментальных наук (прежде всего информатика и математика), инженерии, спорта и ремесла. Кодирование программно-

го обеспечения — собственно разработка программы или написание кода программы. Используемые технологии различны, и у каждого специалиста в этой сфере есть свои предпочтения. На данном этапе осуществляется создание ПО с учетом выставленного технического задания. Некоторые путают такие понятия, как «программирование» и непосредственно «кодирование». Кодирование является лишь частью программирования наряду с анализом, проектированием, компиляцией, тестированием и отладкой, сопровождением. В узком смысле слова, программирование рассматривается как кодирование алгоритмов на заданном языке программирования. Под программированием также может пониматься разработка логической схемы для программируемой логической интегральной схемы, а также процесс записи информации в ПЗУ. В более широком смысле программирование — процесс создания программ, т. е. разработка программного обеспечения.

Процесс кодирования информации может производиться либо ручным, либо автоматическим способом.

Отладка

Это процесс локализации и исправления ошибок, обнаруженных при тестировании программного обеспечения [23].

Локализацией называют процесс определения оператора программы, выполнение которого вызвало нарушение нормального вычислительного процесса. Для исправления ошибки необходимо определить ее причину, т. е. определить оператор или фрагмент, содержащие ошибку. Причины ошибок могут быть как очевидны, так и очень глубоко скрыты. В целом сложность отладки обусловлена следующими причинами:

- требует от программиста глубоких знаний специфики управления используемых технических средств, операционной системы, среды и языка программирования, реализуемых процессов, природы и специфики различных ошибок, методик отладки и соответствующих программных средств;

- психологически дискомфортна, так как необходимо искать собственные ошибки и, как правило, в условиях ограниченного времени;
- возможно взаимовлияние ошибок в разных частях программы, например за счет затирания области памяти одного модуля другим из-за ошибок адресации;
- отсутствуют четко сформулированные методики отладки.

Различают следующие виды ошибок:

- *синтаксические ошибки* — ошибки, фиксируемые компилятором (транслятором, интерпретатором) при выполнении синтаксического и частично семантического анализа программы;
- *ошибки компоновки* — ошибки, обнаруженные компоновщиком (редактором связей) при объединении модулей программы;
- *ошибки выполнения* — ошибки, обнаруженные операционной системой, аппаратными средствами или пользователем при выполнении программы.

Синтаксические ошибки относят к группе самых простых, так как синтаксис языка, как правило, строго формализован, и ошибки сопровождаются развернутым комментарием с указанием ее местоположения. Определение причин таких ошибок, как правило, труда не составляет, и даже при нечетком знании правил языка за несколько прогонов удается удалить все ошибки данного типа. Следует иметь в виду, что чем лучше формализованы правила синтаксиса языка, тем больше ошибок из общего количества может обнаруживать компилятор и соответственно меньше ошибок будет обнаруживаться на следующих этапах. В связи с этим говорят о языках программирования с защищенным и с незащищенным синтаксисом.

Ошибки компоновки, как следует из названия, связаны с проблемами, обнаруженными при разрешении внешних ссылок. Например, предусмотрено обращение к подпрограмме другого модуля, а при объединении модулей данная подпрограмма не найдена или не стыкуются списки параметров. В большинстве случаев ошибки такого рода также удается быстро локализовать и устранить.

Ошибки выполнения относятся к самой непредсказуемой группе. Прежде всего они могут иметь разную природу и соответственно по-разному проявляться. Часть ошибок обнаруживается и документируется операционной системой. Выделяют четыре способа проявления таких ошибок:

- появление сообщения об ошибке, зафиксированной схемами контроля выполнения машинных команд, например переполнение разрядной сетки, ситуации «деление на ноль», нарушение адресации и т. п.;
- появление сообщения об ошибке, обнаруженной операционной системой, например, нарушение защиты памяти, попытка записи на устройства, защищенные от записи, отсутствие файла с заданным именем и т. п.;
- «зависание» компьютера, как простое, когда удается завершить программу без перезагрузки операционной системы, так и «тяжелое», когда для продолжения работы необходима перезагрузка;
- несовпадение полученных результатов с ожидаемыми.

Причины ошибок выполнения очень разнообразны, а потому и локализация может оказаться крайне сложной. Все возможные причины ошибок можно разделить на следующие группы:

- неверное определение исходных данных (ошибки передачи, преобразования, перезаписи, неправильные данные);
- логические ошибки (проектирования, кодирования);
- накопление погрешностей результатов вычислений (некорректное использование приближенных методов вычислений).

Отладка программы в любом случае предполагает обдумывание и логическое осмысление всей имеющейся информации об ошибке. Большинство ошибок можно обнаружить по косвенным признакам посредством тщательного анализа текстов программ и результатов тестирования без получения дополнительной информации. При этом используют различные методы.

Метод ручного тестирования — самый простой и естественный способ. При обнаружении ошибки необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которыми была обнаружена ошибка. Метод очень эффективен, но неприменим для больших программ, программ со сложными вычислениями и в тех случаях, когда ошибка связана с неверным представлением программиста о выполнении некоторых операций. Данный метод часто используют как составную часть других методов отладки.

Метод индукции основан на тщательном анализе симптомов ошибки, которые могут проявляться как неверные результаты вычислений или как сообщение об ошибке. Если компьютер просто «зависает», то фрагмент проявления ошибки вычисляют исходя из последних полученных результатов и действий пользователя. Полученную таким образом информацию организуют и тщательно изучают, просматривая соответствующий фрагмент программы. В результате этих действий выдвигают гипотезы об ошибках, каждую из которых проверяют. Если гипотеза верна, то детализируют информацию об ошибке, иначе — выдвигают другую гипотезу.

Метод дедукции основан на формировании множества причин, которые могли бы вызвать данное проявление ошибки. Затем, анализируя причины, исключают те, которые противоречат имеющимся данным. Если все причины исключены, то следует выполнить дополнительное тестирование исследуемого фрагмента. В противном случае наиболее вероятную гипотезу пытаются доказать. Если гипотеза объясняет полученные признаки ошибки, то ошибка найдена, иначе — проверяют следующую причину.

Метод обратного прослеживания эффективен для небольших программ. Начинают с точки вывода неправильного результата. Для этой точки строится гипотеза о значениях основных переменных, которые могли бы привести к получению имеющегося результата. Далее, исходя из этой гипотезы, делают предложения о значениях переменных в предыдущей точке. Процесс продолжают, пока не обнаружат причину ошибки.

Отладочный вывод. Метод требует включения в программу дополнительного отладочного вывода в узловых точках. Узловыми считают точки алгоритма, в которых основные переменные программы меняют свои значения. Данный метод не очень эффективен и в настоящее время практически не используется, так как в сложных случаях в процессе отладки может потребоваться вывод большого количества значений многих переменных, которые выводятся при каждом изменении. Кроме того, внесение в программы дополнительных операторов может привести к изменению проявления ошибки.

Большинство современных сред программирования (Delphi, Builder C++, Visual Studio и т. д.) включают средства отладки, которые обеспечивают максимально эффективную отладку. Они позволяют:

- выполнять программу по шагам, причем как с заходом в подпрограммы, так и выполняя их целиком;
- предусматривать точки останова;
- выполнять программу до оператора, указанного курсором;
- отображать содержимое любых переменных при пошаговом выполнении;
- отслеживать поток сообщений и т. п.

При отладке программ иногда используют специальные программы — *отладчики*, которые позволяют выполнить любой фрагмент программы в пошаговом режиме и проверить содержимое интересующих программиста переменных. Как правило, такие отладчики позволяют отлаживать программу только в машинных командах, представленных в 16-ричном коде.

Суммируя все сказанное выше, можно предложить следующую методику отладки программного обеспечения, написанного на универсальных языках программирования:

1-й этап — изучение проявления ошибки — если выдано какое-либо сообщение или неправильные или неполные результаты, то необходимо их изучить и попытаться понять, какая ошибка могла так проявиться. При этом используют индуктивные и дедуктивные ме-

тоды отладки. В результате выдвигают версии о характере ошибки, которые необходимо проверить. Для этого можно применить методы и средства получения дополнительной информации об ошибке. Если ошибка не найдена или система просто «зависла», переходят ко второму этапу.

2-й этап — локализация ошибки — определение конкретного фрагмента, при выполнении которого произошло отклонение от предполагаемого вычислительного процесса. Локализация может выполняться:

- путем отсечения частей программы, причем если при отсечении некоторой части программы ошибка пропадает, то это может означать как то, что ошибка связана с этой частью, так и то, что внесенное изменение изменило проявление ошибки;
- с использованием отладочных средств, позволяющих выполнить интересующий нас фрагмент программы в пошаговом режиме и получить дополнительную информацию о месте проявления и характере ошибки, например уточнить содержимое указанных переменных.

При этом если были получены неправильные результаты, то в пошаговом режиме проверяют ключевые точки процесса формирования данного результата. Как подчеркивалось выше, ошибка не обязательно допущена в том месте, где она проявилась. Если в конкретном случае это так, то переходят к следующему этапу.

3-й этап — определение причины ошибки — изучение результатов второго этапа и формирование версий возможных причин ошибки. Эти версии необходимо проверить, возможно, используя отладочные средства для просмотра последовательности операторов или значений переменных.

4-й этап — исправление ошибки — внесение соответствующих изменений во все операторы, совместное выполнение которых привело к ошибке.

5-й этап — повторное тестирование — повторение всех тестов с начала, так как при исправлении обнаруженных ошибок часто вносят в программу новые.

Документирование

Это важная часть в разработке программного обеспечения, но часто ей уделяется недостаточно внимания. Существуют четыре основных типа документации на ПО [24], [25]:

- архитектурная/проектная — обзор программного обеспечения, включающий описание рабочей среды и принципов, которые должны быть использованы при создании ПО;
- техническая — документация на код, алгоритмы, интерфейсы, API;
- пользовательская — руководства для конечных пользователей, администраторов системы и другого персонала;
- маркетинговая.

Архитектурная/проектная документация. Проектная документация обычно описывает продукт в общих чертах, не описывая того, как что-либо будет использоваться, она скорее отвечает на вопрос: «Почему именно так?» Например, в проектном документе программист может описать обоснование того, почему структуры данных организованы именно таким образом. Описываются причины, почему какой-либо фрагмент сконструирован определенным образом, выделяются паттерны, в некоторых случаях могут быть даны предложения по улучшению ПО в дальнейшем.

Техническая документация. При создании программы одного лишь кода, как правило, недостаточно. Должен быть предоставлен некоторый дополнительный материал, описывающий различные аспекты реализации кода. Такая документация часто включается непосредственно в исходный код или предоставляется вместе с ним. Подобная документация имеет сильно выраженный технический характер и в основном используется для определения и описания API, структур данных и алгоритмов. Часто при составлении технической документации используются автоматизированные средства — генераторы документации, такие как Doxygen, javadoc, NDoc и др. Они получают информацию из специальным образом

оформленных комментариев в исходном коде и создают справочные руководства в каком-либо формате, например в виде текста или HTML. Использование генераторов документации и документирующих комментариев многими программистами признается удобным средством по различным причинам. В частности, при таком подходе документация является частью исходного кода, и одни и те же инструменты могут использоваться для сборки программы и одновременной сборки документации к ней. Это также упрощает поддержку документации в актуальном состоянии.

Пользовательская документация. В отличие от технической документации, сфокусированной на коде и том, как он работает, пользовательская документация описывает лишь то, как использовать программу. В случае, если продуктом является программная библиотека, пользовательская документация и документация на код становятся очень близкими, почти эквивалентными понятиями. Но в общем случае это не так. Обычно пользовательская документация представляет собой руководство пользователя, которое описывает каждую функцию программы, а также шаги, которые нужно выполнить для использования этой функции. Хорошая пользовательская документация идет еще дальше и предоставляет инструкции о том, что делать в случае возникновения проблем. Очень важно, чтобы документация не вводила в заблуждение и была актуальной. Руководство должно иметь четкую структуру; очень полезно, если имеется сквозной предметный указатель. Логическая связность и простота также имеют большое значение. Существуют три подхода к организации пользовательской документации. Вводное руководство (*англ.* tutorial), наиболее полезное для новых пользователей, последовательно проводит по ряду шагов, служащих для выполнения каких-либо типичных задач. Тематический подход, при котором каждая глава руководства посвящена какой-то отдельной теме, больше подходит для совершенствующихся пользователей. В последнем, третьем, подходе команды или задачи организованы в виде алфавитного справочника — часто это

хорошо воспринимается продвинутыми пользователями, знающими, что они ищут. Жалобы пользователей обычно относятся к тому, что документация охватывает только один из этих подходов и поэтому хорошо подходит лишь для одного класса пользователей. Во многих случаях разработчики программного продукта ограничивают набор пользовательской документации лишь встроенной системой помощи (*англ.* on-line help), содержащей справочную информацию о командах или пунктах меню. Работа по обучению новых и поддержке совершенствующихся пользователей перекладывается на частных издателей, часто оказывающих значительную помощь разработчикам.

Маркетинговая документация. Для многих приложений необходимо располагать рядом рекламные материалы, с тем чтобы заинтересовать людей, обратив их внимание на продукт. Такая форма документации имеет цель: подогреть интерес к продукту у потенциальных пользователей, информировать их о том, что именно делает продукт, с тем чтобы их ожидания совпадали с тем, что они получат, объяснить положение продукта по сравнению с конкурирующими решениями. Одна из хороших маркетинговых практик — предоставление слогана — простой запоминающейся фразы, иллюстрирующей то, что мы хотим донести до пользователя, а также характеризующей ощущение, которое создает продукт. Часто бывает так, что коробка продукта и другие маркетинговые материалы дают более ясную картину о возможностях и способах использования программы, чем все остальное.

Самым главным документом является ТЗ (техническое задание), в котором описываются цели и задачи работы, заказчик и исполнители, технические требования, сроки и этапы, требования секретности, форс-мажорные обстоятельства и правила предъявления результатов. Технические требования, в свою очередь, делятся на функциональные, экономические, требования по надежности, эффективности, защищенности от несанкционированного доступа и др.

Следующим по важности документом является ПМИ (программа и методика испытаний). Структурно ПМИ по-

добна ТЗ — практически для каждого пункта ТЗ в ПМИ говорится, как этот пункт будет проверяться. Способы проверки могут быть самыми разными — от пропуска специального теста до изучения исходных текстов программы, но они должны быть предусмотрены заранее.

Руководство системного программиста на современном русском языке называется руководством по инсталляции. В нем описывается порядок установки системы на ЭВМ, как проверить корректность поставленной системы, как вносить изменения и т. п. Обычно это простой короткий документ; в противном случае система, наверное, плохая, сделанная непрофессионалами.

Руководство оператора (пользователя) — это основной документ, описывающий, как пользоваться системой. В хорошем руководстве сначала описывается идея системы, основные функции и как ими пользоваться, а уже потом идет описание всех клавиш и меню.

Руководство программиста. Часто это самый объемный документ, описывающий внутреннюю организацию программы. Обычно этот документ идет в паре с документом «текст программы». Руководство программиста дает заказчику возможность дописать какие-то новые фрагменты программы или переделать старые. В современной литературе этот документ называется SDK (Software Development Kit). Продукт, снабженный SDK, может стоить на порядок дороже, чем такой же продукт без оного, так что можно понять, насколько трудно создать действительно полезный SDK. Сейчас не очень принято продавать исходные тексты программ — проблемы с интеллектуальной собственностью, даже при наличии SDK трудно «влезть» в чужую программу. Поэтому большое распространение получили API (Application Program Interface). Программа передается только в виде DLL (библиотека двоичных кодов), но известно, как обратиться к каждой функции из других программ, т. е. известно имя точки входа, количество, типы и значения параметров. Наличие множества API, конечно, хуже, чем наличие исходных текстов (например, нельзя переделать что-то в середине функции), зато много проще в исполь-

зовании. С другой стороны, все большую популярность приобретает FSF (Free Software Foundation). Основателем этого движения был Ричард Столман, который забил тревогу по поводу попыток крупных фирм запатентовать многие основные алгоритмы и программы: «Дойдет до того, что они запатентуют понятия „цикл“ и „подпрограмма“, что мы будем тогда делать?» FSF представляет собой собрание программ в исходных текстах. Любой программист может свободно использовать их в своих целях, но все добавления и улучшения, которые он сделал, тоже следует положить в FSF. Таким образом, FSF представляет собой одно из самых больших доступных хранилищ программ.

Остальные документы, перечисленные в ЕСПД, носят формальный или необязательный характер.

Тестирование

К сожалению, наличие ошибок считается нормальным явлением, реальную рабочую программу невозможно создать безошибочно с первой попытки. Во время тестирования наиболее важны: умение тестировщика найти в программе слабые места и умение программиста исправить ошибочную часть кода, не затронув при этом рабочие участки. Наиболее важные участки многократно проверяются различными методами. Программа полного тестирования может помочь обнаружить проблемы до того, как они проявят себя в эксплуатируемом программном продукте. Кроме того, соответствующая документация необходима тем, кто проводит тестирование, чтобы было понятно, что нужно проверять. Поэтому такие правила должны начинаться с требований по тестированию и документированию. Формулировка может выглядеть следующим образом. Все собственные разработки должны быть протестированы и задокументированы перед их сдачей в промышленную эксплуатацию.

Важно отметить, что эта формулировка устанавливает требование, но не тип тестирования или формат документации. Данное предложение необходимо включить в правила для всего программного обеспечения и в процедуры.

Многие организации, занимающиеся созданием программного обеспечения, до 50% средств, выделенных на разработку программ, тратят на тестирование, что составляет миллиарды долларов по всему миру в целом. И все же, несмотря на громадные капиталовложения, знаний о сути тестирования явно не хватает и большинство программных продуктов неприемлемо ненадежно даже после «основательного тестирования».

Многими тестирование трактуется как процесс, подтверждающий правильность программы и демонстрирующий, что ошибок в программе нет. Такое определение логически некорректно. Тестирование надо рассматривать как процесс выполнения программы с намерением найти ошибки.

Психологические эксперименты показывают, что большинство людей, поставив цель (например, показать, что ошибок нет), ориентируются в своей деятельности на достижение этой цели [26]. Тестовик подсознательно не позволит себе действовать против цели, т. е. подготовить тест, который выявил бы одну из оставшихся в программе ошибок. Поскольку мы все признаем, что совершенство в проектировании и кодировании любой программы недостижимо и поэтому каждая программа содержит некоторое количество ошибок, самым плодотворным применением тестирования будет найти некоторые из них. Если мы хотим добиться этого и избежать психологического барьера, мешающего нам действовать против поставленной цели, наша цель должна состоять в том, чтобы найти как можно больше ошибок. Сформулируем основополагающий вывод.

Если ваша цель — показать отсутствие ошибок, вы их найдете не слишком много. Если же ваша цель — показать наличие ошибок, вы найдете значительную их часть.

Надежность невозможно внести в программу в результате тестирования, она определяется правильностью этапов проектирования. Наилучшее решение проблемы надежности — с самого начала не допускать ошибок в программе. Однако вероятность того, что удастся безупречно спроектировать большую программу, бесконечно мала. Роль тестирования состоит как раз в том, чтобы опреде-

лить местонахождение немногочисленных ошибок, оставшихся в хорошо спроектированной программе. Попытки с помощью тестирования достичь надежности плохо спроектированной программы совершенно бесплодны.

Рассмотрим классификацию различных форм тестирования.

Тестирование (testing), как мы уже выяснили, — процесс выполнения программы (или части программы) с намерением (или целью) найти ошибки.

Доказательство (proof) — попытка найти ошибки в программе безотносительно к внешней для программы среде. Большинство методов доказательства предполагает формулировку утверждений о поведении программы и затем вывод и доказательство математических теорем о правильности программы. Доказательства могут рассматриваться как форма тестирования, хотя они и не предполагают прямого выполнения программы. Многие исследователи считают доказательство альтернативой тестированию — взгляд во многом ошибочный; более подробно это обсуждается.

Контроль (verification) — попытка найти ошибки, выполняя программу в тестовой, или моделируемой, среде.

Испытание (validation) — попытка найти ошибки, выполняя программу в заданной реальной среде.

Аттестация (certification) — авторитетное подтверждение правильности программы, аналогичное аттестации электротехнического оборудования Underwriters Laboratories. При тестировании с целью аттестации выполняется сравнение с некоторым заранее определенным стандартом.

Отладка (debugging) не является разновидностью тестирования. Хотя слова «отладка» и «тестирование» часто используются как синонимы, под ними подразумеваются разные виды деятельности. Тестирование — деятельность, направленная на обнаружение ошибок; отладка направлена на установление точной природы известной ошибки, а затем — на исправление этой ошибки. Эти два вида деятельности связаны — результаты тестирования являются исходными данными для отладки.

Тестирование модуля, или автономное тестирование (Module Testing, Unit Testing) — контроль отдельного программного модуля, обычно в изолированной среде (т. е. изолированно от всех остальных модулей). Тестирование модуля иногда включает также математическое доказательство.

Тестирование сопряжений (Integration Testing) — контроль сопряжений между частями системы (модулями, компонентами, подсистемами).

Тестирование внешних функций (External Function Testing) — контроль внешнего поведения системы, определенного внешними спецификациями.

Комплексное тестирование (System Testing) — контроль и/или испытание системы по отношению к исходным целям. Комплексное тестирование является процессом контроля, если оно выполняется в моделируемой среде, и процессом испытания, если выполняется в среде реальной, жизненной.

Тестирование приемлемости (Acceptance Testing) — проверка соответствия программы требованиям пользователя.

Тестирование настройки (Installation Testing) — проверка соответствия каждого конкретного варианта установки системы с целью выявить любые ошибки, возникшие в процессе настройки системы.

Тестирование программного обеспечения охватывает целый ряд видов деятельности, весьма аналогичный последовательности процессов разработки программного обеспечения. Сюда входят постановка задачи для теста, проектирование, написание тестов, тестирование тестов и, наконец, выполнение тестов и изучение результатов тестирования. Решающую роль играет проектирование теста.

Различают тестирование по типам:

- черный ящик (без просмотра исходного текста);
- белый ящик (с изучением исходного текста);

Сторонники первого подхода проектирует свои тесты, исследуя внешние спецификации или спецификации сопряжения программы или модуля, которые он тестирует.

Позиция их такова: «Нас не интересует, как выглядит эта программа и выполнил ли я все команды или все пути. Мы будем удовлетворены, если программа будет вести себя так, как указано в спецификациях». Их идеал — проверить все возможные комбинации и значения на входе.

Приверженцы второго подхода проектируют свои тесты, изучают логику программы. Они начинают с того, что стремятся подготовить достаточное количество тестов для того, чтобы каждая команда была выполнена по крайней мере один раз. Если они искушеннее, то проектируют тесты так, чтобы каждая команда условного перехода выполнялась в каждом направлении хотя бы раз. Их идеал — проверить каждый путь, каждую ветвь алгоритма. При этом их совсем (или почти совсем) не интересуют спецификации.

Ни один из этих подходов не является хорошей стратегией. К сожалению, они страдают тем недостатком, что в идеале совершенно неосуществимы из-за огромного числа тестов.

Поскольку исчерпывающее тестирование невозможно, мы должны ограничиться чем-то меньшим.

По объему можно выделить следующие тесты:

- маленький тест, типа печати «hello, world», чтобы понять, есть ли вообще о чем говорить;
- получасовое тестирование (американцы говорят «Тест на одну сигарету»), обычно проверяют по одному тесту на каждую функцию программы перед серьезным тестированием;
- модульное тестирование;
- комплексное тестирование.

Каждый тест должен давать максимальную отдачу по сравнению с нашими затратами. Эта отдача измеряется вероятностью тою, что тест выявит необнаруженную прежде ошибку. Затраты измеряются временем и стоимостью подготовки, выполнения и проверки результатов теста. Считая, что затраты ограничены бюджетом и графиком, можно утверждать, что искусство тестирования, по существу, представляет собой искусство отбора тестов с максимальной отдачей. Более того, каждый тест должен быть

представителем некоторого класса входных значений, так чтобы его правильное выполнение создавало у нас некоторую убежденность в том, что для определенного класса входных данных программа будет выполняться правильно. Это обычно требует некоторого знания алгоритма и структуры программы, и мы, таким образом, смещаемся к правому концу спектра.

Сопровождение

Процесс изменения приложения после поставки с целью исправления ошибок, повышения производительности или для адаптации к изменившимся условиям.

Начинается этап сопровождения, который длится, пока программа живет. В процессе эксплуатации возникают новые требования, всплывают ошибки, объемы обрабатываемых данных растут, соответственно растут и требования к эффективности. Основными задачами сопровождения являются:

- исправление ошибок;
- регулярное проведение замеров производительности (профилирование);
- улучшение «времяпожирающих» мест в программе (еще говорят «бутылочное горлышко»);
- улучшение документации;
- расширение функциональности;
- принятие решения о прекращении эксплуатации программы или ее реинжиниринге (переводе программы на другую платформу).

Очень трудно сопровождать программу, в процессе разработки которой о сопровождении никто не думал. Такие простые вещи, как разумные комментарии, выбор идентификаторов и ступенчатое расположение строк исходного текста, заметно облегчают сопровождение. Только на этапе разработки можно предусмотреть средства для профилирования, снятия трасс, отладочных печатей и т. п.

Сопровождение — это особое искусство. Старые программы (*legacy* — унаследованные) обычно состоят из сотен или даже тысяч модулей общим объемом в миллионы

строк исходного кода. Вряд ли кто-то возьмется все прочитать, понять и запомнить. Обычно сопровождающий программист, исправляя ошибку, лишь примерно знает, где ее искать. Более того, после нахождения неправильного фрагмента не всегда ясно, как повлияет исправление на другие участки кода. Часто бывает, что, исправив одну ошибку, получаешь две других.

Реинжиниринг

Особый интерес вызывает вопрос, когда же следует прекращать сопровождение [27]. На этот вопрос отвечает диаграмма оценки качества ПО (рис. 3 23).

Здесь по горизонтали растет важность программы для бизнеса компании, а по вертикали — сложность внесения исправлений. Так вот, если исправлять несложно и важность небольшая, то надо просто сопровождать, исправляя ошибки (corrective fixing), если же программа важна для компании, то можно и улучшать ее (adaptive fixing). Если исправлять сложно, а важность небольшая, то легче программу выкинуть, если же программа важна, содержит определенные знания бизнес-процессов, но сопровождать ее трудно, например потому, что авторы программы покинули компанию или компания вынуждена сменить платформу, то программу следует подвергнуть реинжинирингу.

Попросту говоря, реинжиниринг — это перевод программы со старых языков (Кобол, ПЛ1, Адабас-Натурал и т. п.) на новые, такие как Java, Visual Basic, C++. На самом деле все гораздо сложнее. Смена операционной системы, используемой базы данных, стиля организации

Рис. 3.23
Диаграмма оценки качества ПО

диалога с пользователем — каждая из этих задач весьма нетривиальна. Если же добавить естественное желание реструктурировать программу, удалить отмершие части, перейти к объектно-ориентированной организации программы и т. д., да еще вспомнить, что все эти задачи требуют перебора и анализа путей в графе управления программы, число которых растет экспоненциально относительно числа вершин, то задача вообще выглядит нерешаемой.

Относительные затраты ресурсов на создание ПО по основным этапам: спецификацию (10%), проектирование (10%), кодирование (10%), отладку (20%) и сопровождение (50%).

Систематический, обоснованный подход к выбору и применению технологий создания программного обеспечения может сократить время и повысить качество разработки программного обеспечения, обеспечить высокую степень его независимости от конкретных разработчиков, а также снизить затраты на разработку и сопровождение программного обеспечения.

Рассмотрим существующие технологии создания ПО.

Восходящее программирование (*программирование «снизу вверх»*) — методика разработки программ, при которой крупные блоки собираются из ранее созданных мелких блоков. Восходящее программирование начинается с разработки ключевых процедур и подпрограмм, которые затем постоянно модифицируются.

Метод расширения ядра — метод восходящего программирования, при котором основное внимание уделяется выявлению множества вспомогательных модулей, а не определению функции всей программы в целом.

Диаграмма функционального моделирования (Structured analysis and design technique, SADT) — инструмент разработки функциональных спецификаций в виде диаграмм, фрагментов текста и глоссария, связанных перекрестными ссылками. В состав диаграммы входят:

- блоки, изображающие активность моделируемой системы;
- дуги, связывающие блоки вместе и изображающие взаимодействия и взаимосвязи между ними.

Место соединения дуги с блоком определяет тип интерфейса:

- управляющая информация входит в блок сверху;
- входная информация, подвергающаяся обработке, показывается с левой стороны блока;
- выходная информация показывается с правой стороны;
- механизм, осуществляющий операцию, представляется дугой, входящей в блок снизу.

Императивное программирование — технология, характеризующаяся принципом последовательного изменения состояния вычислителя пошаговым образом. При этом управление изменениями полностью определено и полностью контролируемо.

Компонентное сборочное программирование — объектно-ориентированное сборочное программирование, основанное на распространении классов в бинарном виде и предоставление доступа к методам класса через строго определенные интерфейсы.

Компонентное сборочное программирование поддерживают технологические подходы COM, CORBA, .Net.

Компьютерный дарвинизм — подход к разработке программных систем, основанный на принципе восходящей разработки при интенсивном тестировании. Подход состоит из трех основных процессов: макетирования, тестирования и отладки.

Логическое программирование — программирование в терминах фактов и правил вывода, с использованием языка, основанного на формальных исчислениях.

Модульное программирование — метод разработки программ, предполагающий разбиение программы на независимые модули. Считается, что:

- оптимальный по размерам модуль целиком помещается на экране дисплея;
- разделение большой программы на модули облегчает ее разработку, отладку и сопровождение.

Сборочное программирование — технология, при которой программа собирается посредством повторного использования уже известных фрагментов программ.

Модульное сборочное программирование — разновидность сборочного программирования, основанная на процедурах и функциях методологии структурного императивного программирования.

Нисходящее программирование (*программирование «сверху вниз»*) — методика разработки программ, при которой разработка начинается с определения целей решения проблемы, после чего идет последовательная детализация, заканчивающаяся детальной программой.

Объектно-ориентированное программирование — технология программирования, при которой программа рассматривается как набор дискретных объектов, содержащих, в свою очередь, наборы структур данных и процедур, взаимодействующих с другими объектами.

Объектно-ориентированное сборочное программирование — разновидность сборочного программирования:

- основанная на методологии объектно-ориентированного программирования;
- предполагающая распространение библиотек классов в виде исходного кода или упаковку классов в динамически компонуемую библиотеку.

Синтезирующее программирование — программирование, предполагающее синтез программы по ее спецификации.

Структурное программирование — методология и технология разработки программных комплексов, основанная на принципах:

- программирования «сверху вниз»;
- модульного программирования.

При этом логика алгоритма и программы должны использовать три основные структуры: последовательное выполнение, ветвление и повторение.

Инструментарий технологии программирования — программные продукты, предназначенные для поддержки технологии программирования. К ним относятся рассмотренные ранее (п. 3.4) CASE-технологии.

3.8. ОБЛАЧНЫЕ ТЕХНОЛОГИИ

Впервые концепция облачных технологий (вычислений) или облачной (рассеянной) обработки данных была предложена в 1970-х гг. Официально идея была опубликована и научно обоснована в 2006 г., когда компания Amazon представила свою инфраструктуру веб-сервисов (Web Services), обеспечивающую не только хостинг, но и предоставление клиенту удаленных вычислительных мощностей. Аналогичные сервисы вслед за Amazon представили Google, Sun и IBM, Microsoft. Причем Microsoft представила не просто сервис, а полноценную облачную операционную систему Windows Azure.

Облачные вычисления (*англ.* cloud computing) — технология распределенной обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как интернет-сервис. Как правило, используемый сегодня термин «облачные вычисления» применим для любых сервисов, которые предоставляются через сеть Интернет. Сам термин «облако» появился из принятого графического обозначения Интернета, который изображается в виде облачков. Таким образом, облачные вычисления — это новая парадигма, предполагающая распределенную и удаленную обработку и хранение данных [28]–[32].

На сегодняшний день облачный сервис включает три основных характеристики, которые отличают его от обычного сервиса:

- режимность «ресурсы по запросу»;
- эластичность;
- независимость от элементов управления инфраструктурой.

В качестве простого примера, отражающего различия облачных систем и обычных, можно привести услуги выдачи ресурсов на веб-серверах. В случае с обычной системой, провайдер взимает плату за предоставленные мощности и ресурсы вне зависимости от их использования. Что же касается облачных структур, то плата взимается лишь за использованные мощности и ресурсы, тем самым экономя деньги пользователя.

Облачные сервисы можно разделить на пять категорий:

- программное обеспечение как услуга;
- платформа как услуга;
- инфраструктура как услуга;
- данные как услуга;
- рабочее место как услуга.

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

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

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

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

С точки зрения инфраструктуры выделяют следующие модели развертывания:

- частное облако;
- публичное облако;
- гибридное облако;
- общественное облако.

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

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

ски существует в юрисдикции владельца — поставщика услуг.

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

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

Наряду с очевидными преимуществами концепция облачных технологий не лишена недостатков. Главные претензии связаны с безопасностью и необходимостью надежного широкополосного доступа в Интернет.

Рассмотрим примеры реализации облачных технологий (помимо веб-почты).

Например, США был запущен облачный сервис OnLive, предоставляющий возможность играть в современные игры даже на самом простом оборудовании. Технически это выглядит следующим образом: сама игра располагается на удаленном сервере и там же производится обработка графики, которая на компьютер конечному пользователю поступает уже в «готовом» виде. Другими словами, вычисления, предназначенные для выполнения на видеокарте и процессоре вашего компьютера, здесь выполняются на сервере, а ваш компьютер используется лишь как монитор.

Также Apple развивает у себя облачную технологию в виде сервиса под названием MobileMe. Сервис включает

в себя почтовый клиент, календарь, адресную книгу, файловое хранилище, альбом фотографий и инструмент для обнаружения утерянного iPhone. Этот сервис является платным, но главное здесь — другое. Apple обеспечивает такой уровень взаимодействия своего набора интернет-сервисов и приложений на компьютере, телефоне, плеере и iPad, что необходимость в использовании браузера пропадает. Вы пользуетесь привычными программами на своем Mac, iPhone и iPad, однако все данные хранятся не на них, а в облаке, что позволяет забыть о необходимости синхронизации, а также о доступности. При этом, оговоримся, не обязательно использовать именно приложения — можно и просто через браузер с любого компьютера зайти в свой аккаунт.

Разрабатываемая Google операционная система Chrome OS представляет собой фактически один браузер, через который пользователь взаимодействует с разветвленной сетью веб-сервисов. ОС ориентирована на нетбуки, отмечаются очень низкие системные требования и отсутствие необходимости самостоятельной установки программ. То есть Google предоставляет преимущества облачной концепции обычным пользователям. Правда, минус этого подхода заключен в том, что без Интернета нетбук на базе Chrome OS будет совершенно бесполезен.

Отметим преимущества и недостатки облачных технологий.

Преимущества облачных вычислений:

- снижение требований к вычислительной мощности пользовательского компьютера (любой компьютер, способный открыть окно браузера, получает огромный потенциал настоящей рабочей станции);
- экономия затрат на приобретении, поддержке, модернизации ПО и оборудования;
- масштабируемость, отказоустойчивость и безопасность — автоматическое выделение и освобождение необходимых ресурсов в зависимости от потребностей приложения. Техническое обслуживание, обновление ПО производит провайдер услуг;

- удаленный доступ к данным в облаке — работать можно из любой точки на планете, где есть доступ в сеть Интернет;
- высокая скорость обработки данных;
- оплата услуг по мере необходимости и только за то, что используется;
- экономия дискового пространства (данные и программы хранятся на удаленных серверах).

Недостатки облачных вычислений:

- зависимость целостности пользовательских данных от компаний, предоставляющих услугу;
- необходимость наличия надежного и быстрого доступа в сеть Интернет;
- отсутствие общепринятых стандартов в направлении безопасности облачных технологий;
- возможность появления облачных монополистов;
- опасность хакерских атак на сервер (при хранении данных на компьютере в любое время можно отключиться от сети и очистить систему с помощью антивируса).

Несмотря на всю критику, у облачных технологий большое будущее. Самым простым доказательством этому служит то, что как бы ни соревновались и ни противоречили друг другу три основных гиганта (Microsoft, Apple и Google), все они практически одновременно устремились в эту новую технологию и уходить оттуда не собираются. Более того, именно с облачными технологиями все три компании связывают свое будущее. Еще несколько лет назад концепция cloud computing казалась лишь красивой идеей, «приманкой», странным экспериментом. Сегодня же преимущества облачных технологий могут почувствовать даже те люди, которые не связаны с разработкой программ, веб-технологиями и прочими узкоспециализированными вещами (Xbox Live, Windows Live, MobileMe, OnLive, Google Docs — яркие тому примеры).

Ниже приведена краткая характеристика нескольких проектов облачных технологий с открытым исходным кодом.

Chef — профессионально поддерживается и спонсируется компанией Opscode. Он актино разрабатывается, что видно по частоте использования регистраций кода. Это фреймворк сборки для управления конфигурацией всех типов IT-инфраструктуры, в частности среди облачных разработок. Сначала пишется исходный код, описывающий, как будет построена инфраструктура, а затем эти описания применяются к серверам. В результате получается полностью автоматизированная инфраструктура. Chef профессионально поддерживается и спонсируется компанией Opscode.

Eucalyptus — это инфраструктура Open Source программного обеспечения для реализации облачных технологий на кластерах. Текущий интерфейс Eucalyptus совместим с интерфейсами Amazon's EC2, S3 и EBS, однако инфраструктура разработана для поддержки множественных клиентских интерфейсов. Eucalyptus реализуется с использованием, как правило, доступных инструментариев Linux и базовых веб-сервисных технологий, что облегчает установку и поддержку системы. Eucalyptus Systems обеспечивают услуги консультирования, обучения и поддержки.

В первую очередь облако интересно конечным пользователям информационных систем. Простейший случай: есть веб-сервис, обслуживающий запросы от пользователей. Сервис реализован в облаке. По мере того как количество запросов растет и сервис перестает справляться с нагрузкой, в систему можно добавить (динамически или по требованию) новые узлы и перераспределить нагрузку между ними.

Во вторую очередь Eucalyptus полезен в непосредственной разработке программных систем. В облаке можно объединить аппаратные ресурсы всех мастей и оттенков, удовлетворяющих требованиям Eucalyptus.

OpenNebula — это, возможно, самый интересный и значимый проект в списке облачных технологий, рекламирующий себя как *Open Source инструментарий для облачных вычислений*. OpenNebula — это инструмент, который может быть использован для любого типа облачного

внедрения и для управления виртуальной инфраструктурой в информационном центре или кластере или для объединения локальной инфраструктуры с публичной облачно-ориентированной инфраструктурой. OpenNebula также поддерживает публичные облака, предоставляя возможность облачным интерфейсам раскрывать их функциональность для виртуальной машины, управления памятью и сетью.

Zenoss — имеет возможность отслеживать Amazon Web Services и все виды другой облачной и виртуальной инфраструктуры.

Enomaly's Elastic Computing Platform (ECP) — программируемая виртуальная облачная инфраструктура для всех типов предприятий. ECP помогает в разработке, управлении и внедрении виртуальных приложений в облаке и значительно снижает административный и системный объем работы. Веб-ориентированная инструментальная панель дает возможность IT-сотрудникам упростить и эффективно спланировать внедрения, автоматизировать масштабирование и балансировку загрузки виртуальных машин, анализировать, настраивать и оптимизировать облачные возможности простых в использовании сервисных программ. ECP была создана для работы с виртуальным центром обработки данных, обеспечивая дополнительную ценность и снижение затрат.

Ubuntu Enterprise Cloud — включает в себя Ubuntu Server Edition и интегрирует несколько Open Source проектов, включая Eucalyptus. UEC предоставляет пользователям пакеты под ключ для внедрения частного облака.

3.9. ТЕХНОЛОГИЯ БОЛЬШИХ ДАННЫХ

Стремительное развитие инструментальных средств создало условия для накопления и хранения больших объемов данных, однако технологии их использования отстают от потребностей пользователей.

Большие данные (Big Data) в информационных технологиях означают серию подходов, инструментов и методов обработки структурированных и неструктурирован-

ных данных огромных объемов и значительного многообразия для получения воспринимаемых пользователем результатов [33]–[35].

В качестве определяющих характеристик для больших данных отмечают *три V*: объем (volume), в смысле величины физического объема, скорость (velocity), в смысле как скорости прироста, так и необходимости высокоскоростной обработки и получения результатов, многообразие (variety), в смысле возможности одновременной обработки различных типов структурированных и полуструктурированных данных.

Введение термина «большие данные» приписывают К. Линчу (2008), с 2009 г. термин широко распространился в деловой прессе, а к 2010 г. относят появление первых продуктов и решений, относящихся исключительно и непосредственно к проблеме обработки больших данных. В настоящее время большинство крупнейших поставщиков информационных технологий для организаций в своих деловых стратегиях используют понятие о больших данных, в том числе IBM, Oracle, Microsoft, Hewlett-Packard, EMC.

На практике данных действительно становится все больше, но проблема вызвана не столько обрушившимися в невероятном количестве данными, а неспособностью старыми методами справиться с новыми объемами. Способность порождать данные оказалась сильнее, чем способность их перерабатывать. Причина возникновения такого дисбаланса связана с недостаточным вниманием к отслеживанию связей цепочки «данные — информация — знание». В современной теории информации недостаточное внимание уделено раскрытию понятий «данные» и «информация», отдавая предпочтение технологии работы с данными. Данные должны обрабатываться для получения информации, необходимой пользователю для превращения в знание.

Принято делить подходы к Big Data на три группы: «быстрые данные» (Fast Data), их объем измеряется терабайтами; «большая аналитика» (Big Analytics) — петабайтные данные и «глубокое проникновение» (Deep

Insight) — экзабайты, зеттабайты. Группы различаются между собой не только оперируемыми объемами данных, но и качеством решения по их обработке.

Обработка для Fast Data не предполагает получения новых знаний, ее результаты соотносятся с априорными знаниями и позволяют судить о том, как протекают те или иные процессы, она позволяет лучше и детальнее увидеть происходящее, подтвердить или отвергнуть какие-то гипотезы. Небольшая часть из существующих сейчас технологий подходит для решения задач Fast Data, среди них некоторые технологии работы с хранилищами (продукты Teradata, Netezza, Greenplum, СУБД типа Verica и kdb). Скорость работы этих технологий должна возрастать синхронно с ростом объемов данных.

Задачи, решаемые средствами Big Analytics, заметно отличаются, причем не только количественно, но и качественно, а соответствующие технологии должны помогать в получении новых знаний — они служат для преобразования зафиксированной в данных информации в новое знание. Однако на этом среднем уровне не предполагается наличие искусственного интеллекта при выборе решений или каких-либо автономных действий аналитической системы — она строится по принципу «обучения с учителем». Иначе говоря, весь ее аналитический потенциал закладывается в нее в процессе обучения. Классическими представителями такой аналитики являются продукты MATLAB, SAS, Revolution R, Apache Hive, SciPy Apache и Mahout.

Высший уровень, Deep Insight, предполагает обучение без учителя (Unsupervised learning) и использование современных методов аналитики, а также различные способы визуализации. На этом уровне возможно обнаружение знаний и закономерностей, априорно неизвестных.

С течением времени компьютерные приложения становятся все ближе к реальному миру во всем его многообразии.

Потенциальные практические сферы применения больших данных отличаются широким разнообразием: здравоохранение и медицина, государственные услуги, розничная торговля, различные производственные от-

расли, телекоммуникационная отрасль, информационно-технологическая отрасль, финансы и страхование, энергетика, маркетинг и риск-менеджмент, издательское дело, медиа и развлечения, биотехнологии, транспорт и логистика, коммунальные услуги. Со временем это разнообразие будет только нарастать. Применительно к коммерческим структурам большие данные призваны стать дополнительным источником конкурентных преимуществ.

Методы анализа больших данных. Рост объемов входных данных и потребность в их аналитике, причем в режиме, максимально приближенном к реальному времени, привели к возникновению направления *аналитика больших данных* (Big Data Analytics).

На текущий момент большие данные опираются на многочисленные методы анализа (табл. 3.5), создававшиеся на протяжении многих десятилетий во многих областях информационных технологий (искусственный интеллект, анализ данных, распознавание и др.). В процессе развития технологии больших данных будут созданы новые методы анализа либо существующие методы будут видоизменяться.

Аппаратное и программное обеспечение больших данных. Выше отмечалось, что появление больших данных обусловлено бурным развитием информационно-телекоммуникационных технологий, что сопровождалось в том числе количественным наращиванием объемов данных, их качественным усложнением, стремительным нарастанием разнообразия. В какой-то момент стало очевидно, что устоявшиеся технологические решения не в состоянии справиться с новым вызовом вовлечения накапливаемых массивом данных в экономическую деятельность. Пренебрежительное отношение к таким данным ведет к потерям, которых хотелось бы избежать. Для решения этой задачи нужны новые технологические подходы.

Ввиду разнообразия рассмотрим одну из технологий — Hadoop. Ее стоит затронуть хотя бы на самом общем уровне, так как на текущий момент она находит наибольше распространение и наиболее обсуждаема в сообществе специалистов.

Таблица 3.5

Методы анализа, применяемые в больших данных

№ п/п	Метод	Краткая характеристика
1	А/Б тестирование (A/B Testing)	Метод, предусматривающий сравнение реакций на контрольный (А) и тестируемый (Б) варианты некоторых стимулов. Тестируемых вариантов может быть несколько. На основании сравнения реакций выбирается наилучшее для данной цели решение
2	Анализ с использованием ассоциативных правил (Association Rule Learning)	Совокупность методов для выявления характерных взаимосвязей (ассоциативных правил) между переменными в больших базах данных. Такие методы представляют собой набор алгоритмов для генерации и тестирования возможных правил.Одно из применений метода ассоциативных правил — анализ рыночной корзины (Market Basket Analysis), применение которого позволяет, например, установить, что покупатель товара А с высокой вероятностью купит и товар Б. Это дает возможность продавцам принимать соответствующие маркетинговые решения
3	Классификация (Classification)	Совокупность методов рубрикации новых данных на основе рубрикации тренировочных данных (Training Data Set). Эти методы относят к группе методов управляемого (направляемого) обучения (Supervised Learning), так как изначально наличествующие тренировочные данные направляют процесс обучения (см. далее)
4	Кластерный анализ (Cluster Analysis)	Статистический метод классификации объектов, предполагающий разбиение всей совокупности объектов на более мелкие группы (кластеры), объединяющие похожие между собой объекты и в то же время сильно отличающиеся от объектов других групп. Характеристики, на основе которых данный объект относится к тому или иному кластеру, заблаговременно неизвестны. Этот метод относят к числу методов ненаправляемого обучения (Unsupervised Learning)
5	Краудсорсинг (Crowdsourcing)	Метод сбора данных, предоставляемых большими группами людей («толпой») в ответ на открытый запрос на такие данные. Как правило, осуществляется с использованием сетевых методов социального взаимодействия
6	Интеграция данных (Data Fusion and Data Integration)	Совокупность методов интеграции и анализа данных, полученных из множественных источников, что позволяет делать более точные выводы по сравнению с ситуацией, когда такие данные используется раздельно

7	Глубинный анализ данных (Data Mining)	К числу методов, применяемых для глубинного анализа данных, можно отнести, например, анализ с использованием ассоциативных правил, кластерный анализ, классификацию, регрессионный анализ и др.
8	Ансамблевое обучение (Ensemble Learning)	Подразумевает использование множественных прогностических моделей (Predictive Models), каждая из которых разработана с использованием статистических методов или машинного обучения, для получения более адекватных прогнозов
9	Генетические (эволюционные) алгоритмы (Genetic Algorithms)	Метод оптимизации, вдохновленный изучением процессов естественной эволюции, которая, как известно, предполагает выживание наиболее приспособленных особей. В этом методе потенциальные решения интерпретируются как «хромосомы», которые могут объединяться и мутировать. «Хромосомы» приводятся в соприкосновение с условиями «окружающей среды». «Выжившие хромосомы» рассматриваются в качестве наилучшего решения. Такого рода методы наиболее ценны при решении нелинейных задач
10	Машинное обучение (Machine Learning)	Подраздел компьютерной науки (относимый к области искусственного интеллекта), занимающийся разработкой алгоритмов, позволяющих компьютерам изменять свое поведение на основе поступающих эмпирических данных. Важнейшее направление исследований — автоматическое распознавание сложных устойчивых моделей и принятие адекватных решений на основе данных
11	Обработка естественных языков (Natural Language Processing)	Составная часть машинного обучения, подразумевающая совместное использование компьютерных технологий и лингвистики для создания алгоритмов, позволяющих анализировать естественные (человеческие) языки
12	Искусственные нейросети (Neural Networks)	Модели вычислений, предназначенные для распознавания устойчивых моделей в массивах данных. Их создание вдохновлено структурой и работой биологических нейросетей
13	Сетевой анализ (Network Analysis)	Совокупность методов, применяемых для описания взаимоотношений между отдельными узлами, объединенных в граф или сеть

Продолжение табл. 3.5

№ п/п	Метод	Краткая характеристика
14	Оптимизация (Optimization)	Совокупность численных методов, применяемых для изменения (to redesign) сложных систем и процессов с целью улучшения их результатов работы, определяемых на основе одного или нескольких объективных численных показателей (например, затраты, скорость, надежность и др.)
15	Распознавание устойчивых моделей (Pattern Recognition)	Совокупность методов машинного обучения, которые в соответствии с конкретным алгоритмом присваивают некоторой входной величине некоторую выходную величину — метку (label). Классификация относится к числу таких методов
16	Прогностическое моделирование (Predictive Modeling)	Совокупность методов, подразумевающих создание или выбор математических моделей для предсказания вероятности некоторого результата (параметра выхода)
17	Регрессионный анализ (Regression Analysis)	Совокупность статистических методов для определения изменений зависимой величины вследствие изменений одной или нескольких независимых величин
18	Анализ настроений (мнений) (Sentiment Analysis)	Применение методов обработки естественных языков и других аналитических методов для выявления и извлечения из анализируемого текста субъективной информации, характеризующей настроения, мнения, отношение людей к проблеме
19	Статистическая обработка сигналов (Signal Processing)	Эти методы заимствованы из электротехники и прикладной математики. Изначально они применялись для анализа дискретных и непрерывных сигналов. В современном анализе данных применяются для статистического вычленения из данных полезного сигнала и шума
20	Пространственный анализ (Spatial Analysis)	Совокупность методов анализа топологических, геометрических или географических свойств объектов, содержащихся в данных
21	Статистика (Statistics)	Наука о сборе, структурировании и интерпретации данных, включая планирование опросов (design of surveys) и планирование экспериментов (design of experiments)

22	Направляемое обучение (Supervised Learning)	Совокупность методов машинного обучения, позволяющих делать выводы о наличии некоторых функций или отношений на основе тренировочных данных. Пример — метод классификации
23	Имитационное моделирование (Simulation)	Создание моделей поведения сложных систем, применяемых для прогнозирования возможных будущих состояний и/или сценарного планирования
24	Анализ временных рядов (Time Series Analysis)	Совокупность методов для анализа последовательностей данных, характеризующих изменение некоторых параметров объекта во времени, с целью содержательной характеризации такого объекта
25	Ненаправляемое обучение (Unsupervised Learning)	Совокупность методов машинного обучения, нацеленных на выявление скрытых взаимосвязей между объектами за счет анализа неохарактеризованных (unlabeled) данных
26	Визуализация (Visualization)	Методы создания изображений, диаграмм, анимации, используемых для коммуникаций, улучшения понимания самих данных и результатов их анализа

Hadoop представляет собой свободно распространяемый набор утилит, библиотек и программный каркас для разработки и выполнения распределенных программ, работающих на кластерах из сотен и тысяч узлов. Отличительные особенности Hadoop следующие:

1) доступность: Hadoop работает на больших кластерах общераспространенных (commodity) компьютеров или в облачных сервисах, таких как Elastic Compute Cloud (EC2) компании Amazon;

2) надежность: так как пакет Hadoop ориентирован на работу на обычных, общедоступных компьютерах, его архитектура предусматривает возможность частых сбоев и позволяет успешно справляться с большинством из них;

3) масштабируемость: Hadoop предусматривает возможность очень простого расширения вычислительных мощностей посредством добавления в кластер дополнительных узлов — общедоступных компьютеров, которых в одном кластере могут быть сотни;

4) простота: Hadoop предоставляет пользователям возможность быстрого написания эффективного программного кода.

Совокупность этих свойств делает Hadoop востребованной технологией в самых различных секторах и применениях. Доступность и простота позволяют быстро и дешево создавать кластеры Hadoop даже учащимся колледжей. Надежность и масштабируемость делают Hadoop привлекательным даже в самых крупных и технологически требовательных проектах и компаниях — таких как Yahoo и Facebook, например.

Ключевые элементы Hadoop — распределенная файловая система HDFS (Hadoop Distributed File System) и MapReduce — технология распределенных параллельных вычислений над очень большими наборами данных в компьютерных кластерах. Отметим лишь наиболее принципиальные особенности этих технологических элементов [36].

Распределенная файловая система HDFS функционирует таким образом, что если требуется обеспечить ра-

боту вычислительной системы с очень большим блоком данных, например несколько терабайт, то этот блок разбивается на более мелкие блоки (обычно 64 Мбайт), которые распределяются по многочисленным компьютерам, составляющим кластер Hadoop. Затем HDFS обеспечит параллельную работу вычислительной системы с каждым малым блоком. В результате кластер простых общедоступных компьютеров справляется с большой вычислительной задачей намного быстрее, чем одиночный сервер с самыми передовыми техническими характеристиками. К тому же затраты на создание кластера будут меньшими по сравнению с затратами на высокопроизводительные серверы.

Другая техническая особенность Hadoop заключена в самой философии работы с данными. Традиционно в работе с данными используется подход, предусматривающий неоднократную передачу данных между клиентами и сервером. Этот подход вполне оправдан, когда объемы данных сравнительно невелики. Однако в больших данных мы имеем дело с большими объемами данных по определению. Перемещение таких данных становится очень затратной затеей. Поэтому философия взаимодействия данных и вычислительных программ меняется на прямо противоположную: данные остаются там, где они изначально сохранены в рамках кластера Hadoop, а пересылаются от клиента к серверу вычислительные программы MapReduce, размеры которых обычно очень малы (килобайты). Как говорится, если гора не идет к Магомету, Магомет идет к горе.

Еще одна важная особенность Hadoop касается принципов построения баз данных. В большинстве сегодняшних приложений используются реляционные базы данных, в качестве ключевого элемента которых выступает технология SQL (Structured Query Language). Ключевое слово в этом наименовании — Structured (структурированный). Оно «выдает» тот факт, что реляционные базы данных и SQL ориентированы на работу с хорошо структурированными данными, подразумевающую тщательный заблаговременный выбор полей и таблиц, в которых данные будут храниться, установление четких взаимосвязей между ними и т. п.

Но задачи больших данных подразумевают работу не только с хорошо структурированными данными, но и с полу- и неструктурированными данными. Применение технологии реляционных баз данных и SQL в таких случаях было бы сопряжено со значительными затратами либо вообще невозможно. Нужны иные технологические подходы к работе с такими данными. В качестве такого подхода Hadoop в роли элементарной единицы данных использует пары «ключ — значение». Другими словами, исходные данные могу поступать в Hadoop в любой форме, однако в какой-то момент они преобразуются в пары «ключ — значение». В качестве технологии формирования и обработки запросов к данным, структурированными по принципу «ключ — значение», вместо SQL используются программы MapReduce, которые предъявляют принципиально иные (существенно менее строгие) требования к структурированности данных и обеспечивают принципиально иной уровень работы с ними.

Разумеется, здесь приведены лишь наиболее примечательные особенности применяемого в задачах больших данных аппаратного и программного обеспечения. Более детальное изложение данного вопроса выходит за рамки настоящей статьи. Всем, кто заинтересован более глубоким изучением технологии Hadoop, можно лишь порекомендовать обратиться к специальным источникам.

Гигантские объемы в сочетании с высокой скоростью, отличающие Big Data Analytics от других приложений, требуют соответствующих компьютеров, и сегодня практически все основные производители предлагают специализированные программно-аппаратные системы: SAP HANA, Oracle Big Data Appliance и Oracle Exalytics Business Intelligence Machine, Teradata Extreme Performance Appliance, NetApp E-Series Storage Technology, IBM Netezza Data Appliance, EMC Greenplum, Vertica Analytics Platform на базе HP Converged Infrastructure. Помимо этого, в игру вступило множество небольших и начинающих компаний: Cloudera, DataStax, Northscale, Splunk, Palantir, Factual, Kognitio, Datameer, TellApart, Paraccel, Hortonworks.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы характерные особенности мультимедиатехнологий?
2. Каковы основные компоненты мультимедиасреды?
3. Какие стандарты используются при создании мультимедиапродуктов?
4. Какие задачи решают геоинформационные технологии?
5. Какие типы геоинформационных систем существуют?
6. Какие классы данных используются в геоинформационных системах?
7. Какие модели данных используются для представления данных в геоинформационных технологиях?
8. Каковы принципы построения цифровой карты?
9. Какие виды обработки информации используют современные геоинформационные системы?
10. Какие существуют виды информационных угроз?
11. Какие существуют способы защиты информации от нарушений работоспособности компьютерных систем?
12. Какие существуют виды преднамеренных информационных угроз?
13. Каковы основные способы запрещения несанкционированного доступа к ресурсам вычислительных систем?
14. Что такое идентификация и аутентификация?
15. Какие существуют способы разграничения доступа к информационным ресурсам?
16. Что такое криптография и каковы ее основные задачи?
17. В чем отличие симметрических криптографических систем от ассиметрических?
18. Что понимается под остаточной информацией и каковы угрозы доступа к ней?
19. Какие существуют уровни защиты информации от компьютерных вирусов?
20. Каковы цели и способы защиты информации при сетевом обмене?
21. Что такое CASE-технология и какой подход к проектированию информационных систем она использует?

22. Какие компоненты включает в себя стандарт OMA для создания распределенных объектных систем?
23. Какие основные блоки содержит объектно-ориентированное CASE-средство?
24. Каковы основные критерии оценки и выбора CASE-средств?
25. Каковы разновидности архитектур компьютерных сетей?
26. Какие используются модели архитектуры «клиент-сервер»?
27. В чем отличие двухзвенной архитектуры «клиент-сервер» от трехзвенной?
28. Каковы особенности архитектуры «клиент-сервер», основанной на веб-технологии?
29. Каковы особенности интернет-технологии?
30. Каковы основные компоненты интернет-технологии?
31. Что такое браузер и какие его типы используются на практике?
32. Какие виды подключений используются для выхода в Интернет?
33. Какие протоколы используются для передачи данных в Интернет?
34. Каковы основные принципы и нормы работы Интернет?
35. Какие функции реализует интеллектуальная система?
36. Какова структура интеллектуальной системы?
37. Какие существуют разновидности интеллектуальных систем?
38. Каковы основные свойства информационно-поисковых систем?
39. Каковы основные свойства экспертных систем?
40. Каковы основные свойства расчетно-логических систем?
41. Каковы основные свойства гибридных экспертных систем?
42. Какие существуют типы моделей представления знаний в искусственном интеллекте?
43. В чем отличие фреймовых моделей от продукционных?

44. На какие типы предметных областей ориентированы экспертные системы?
45. Какие методы используются экспертными системами для решения задач?
46. В чем отличие поверхностных экспертных систем от глубинных?
47. По совокупности каких характеристик определяют особенности конкретной экспертной системы?
48. Что называют демонстрационным прототипом экспертной системы?
49. Какие инструментальные средства используются для построения экспертных систем?
50. Какие алгоритмы поиска решений используются в интеллектуальных системах расчетно-логического типа?
51. Каковы особенности гибридной экспертной системы?
52. Дайте характеристику развития методов и средств разработки ПО.
53. Перечислите основные этапы технологического процесса разработки ПО.
54. Раскройте содержание этапа спецификации ПО.
55. Раскройте содержание этапа проектирования ПО.
56. Раскройте содержание этапа кодирования ПО.
57. Раскройте содержание этапа отладки ПО.
58. Раскройте содержание этапа документирования ПО.
59. Раскройте содержание этапа тестирования ПО.
60. Раскройте содержание этапа сопровождения ПО.
61. Раскройте содержание этапа реинжиниринга ПО.
62. Дайте сравнительную характеристику существующих технологий создания ПО.
63. В чем сущность облачных вычислений?
64. Дайте сравнительную характеристику основных категорий облачных сервисов.
65. Какие существуют модели развертывания инфраструктуры облачных технологий?
66. Приведите примеры реализации облачных технологий.
67. Укажите преимущества и недостатки облачных технологий.

68. Дайте краткую характеристику проектов облачных технологий.
69. Что такое большие данные?
70. Назовите определяющие характеристики для больших данных.
71. Какие существуют подходы к большим данным?
72. В чем сущность методов анализа для больших данных?
73. Дайте сравнительную характеристику методов анализа, применяемых в больших данных.
74. Охарактеризуйте аппаратное и программное обеспечение больших данных.

ГЛАВА 4

ПРИКЛАДНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

В результате освоения данной темы студент должен:

1) знать:

- методологии корпоративного управления, возможности и области их применения;
- назначение и содержание концепций MRP и MRPII, структуру и состав задач, решаемых системой управления предприятием, построенной в соответствии со стандартом MRPII, цели внедрения систем типа MRP II/ERP;
- основные концепции модели жизненного цикла (PLM), функции PLM, содержание этапов жизненного цикла;
- состав и функции интегрированной информационной средой управления ЖЦИ;

2) уметь:

- применять методологию и модели корпоративного управления при проектировании информационных систем;
- использовать архитектурные и детализированные решения информационных технологий при проектировании корпоративных информационных систем;

3) владеть:

- методологией проектирования корпоративных информационных систем;
- моделями и средствами разработки корпоративных информационных систем.

4.1. ПРИКЛАДНОЙ ХАРАКТЕР ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Информационные технологии являются не только объектом исследований и разработки, но и средством создания информационных систем в различных предметных областях. Несмотря на специфику конкретных объектов, удалось разработать методологию, модели, методы и средства прикладных информационных технологий, что позволяет снизить затраты и сократить сроки информатизации. Спектр прикладных информационных технологий широк. Исходя из ограниченного объема учебника, рассмотрим общий подход к реализации прикладных информационных технологий.

Если на первых порах информатизации в каждой предметной области решались отдельные локальные задачи, то в настоящее время наметилась тенденция к комплексному характеру использования информационных технологий. Общим для любой предметной области является управление материальными, финансовыми, трудовыми, информационными ресурсами, а специфика проявляется в решении технологических задач.

Прикладные информационные технологии, основываясь на стандартных моделях, методах и средствах, допускают формулировку, постановку и реализацию поставленных задач в терминах предметной области пользователя. Совершенствование данного класса технологий направлено на обеспечение автоматизированного формирования модели предметной области и погружения ее в стандартную инструментальную среду.

Таким образом, корпоративное управление является общим для использования информационных технологий в промышленности и экономике, в образовании, медицине и других предметных областях.

Корпоративное управление и создание корпоративных информационных систем в настоящее время опираются на различные информационные технологии, так как, к сожалению, не существует универсальной. Можно выделить следующие три группы методов управления: ресурсами, процессами, корпоративными знаниями

(коммуникациями). Среди информационных технологий в качестве наиболее используемых можно выделить следующие: СУБД, Workflow, стандарты ассоциации Workflow Management Coalition, Intranet. На рисунке 4.1 показаны место и назначение каждой из информационных технологий.

На рисунке 4.2 интенсивность цвета соответствует степени поддержки информационными технологиями методов управления.

Рис. 4.1
Место и назначение каждой из информационных технологий в корпоративном управлении

	Ресурсы	Процессы	Коммуникации-знания
СУБД			
Workflow			
Интранет			

Рис. 4.2
Степень поддержки информационными технологиями методов управления

Задача управления ресурсами относится к числу классических методик управления и является первой, где широко стали использоваться информационные технологии. Это связано с наличием хорошо отработанных экономико-математических моделей, эффективно реализуемых средствами вычислительной техники. Рассмотрим эволюцию задач управления ресурсами.

Первоначально была разработана методология планирования материальных ресурсов предприятия MRP (Material Requirements Planning), которая использовалась с методологией объемно-календарного планирования MPS (Master Planning Shedule). Следующим шагом было создание методологии планирования производственных ресурсов (мощностей) — CRP (Capacitiy Requirements Planning). Эта методология была принципиально похожа на MRP, но была ориентирована на расчет производственных мощностей, а не материалов и компонентов. Эта задача требует больших вычислительных ресурсов даже на современном уровне.

Объединение указанных выше методологий привело к появлению задачи MRP «второго уровня» MRPII (Manufacturing Resource Planning) — интегрированной методологии планирования, включающей MRP/CRP и использующей MPS и FRP (Finance Resource/Requirements Planning) — планирование финансовых ресурсов. Далее была предложена концепция ERP (Economic Requirements Planning — интегрированное планирование всех бизнес-ресурсов предприятия.

Эти методологии были поддержаны соответствующими инструментальными средствами. В большей степени к поддержке данных методологий применимы СУБД.

Следующим шагом было создание концепции управления производственными ресурсами — CSPP (Customer Synchronized Resource Planning) — планирование ресурсов, синхронизированное с потреблением. Отличием данной концепции является учет вспомогательных ресурсов, связанных с маркетингом, продажей и послепродажным обслуживанием. На рисунке 4.3 показано соотношение между понятиями CSSP, ERP и стадиями жизненного цикла товара.

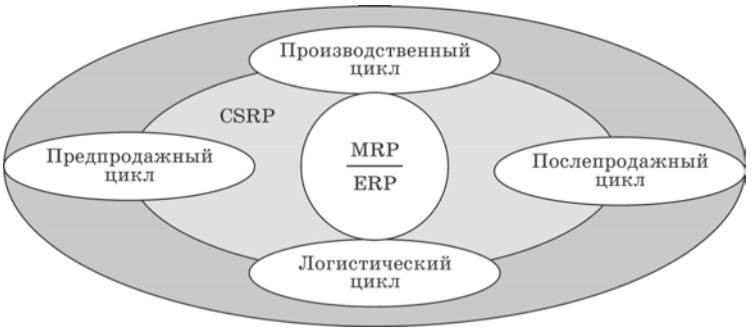


Рис. 4.3
Соотношение между понятиями CSSP, ERP и стадиями жизненного цикла товара

В связи с тем, что в современном производстве задействовано множество поставщиков и покупателей, появилась новая концепция логистических цепочек (Supply Chain). Суть этой концепции состоит в учете при анализе хозяйственной деятельности всей цепочки (сети) превращения товара из сырья в готовое изделие (рис. 4.4).

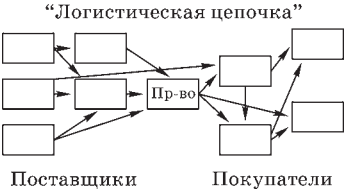


Рис. 4.4
Концепция логистических цепочек

При этом акцент сделан на следующие факторы:

- стоимость товара формируется на протяжении всей логистической цепочки, но определяющей является стадия продажи конечному потребителю;
- на стоимости товара критическим образом сказывается общая эффективность всех операций;
- наиболее управляемыми являются начальные стадии производства товара, а наиболее чувствительными — конечные (продажные).

Дальнейшим развитием концепции логистических цепочек является идея виртуального бизнеса (рис. 4.5), представляющего распределенную систему нескольких компаний и охватывающего полный жизненный цикл товара, или разделение одной компании на несколько «виртуальных бизнесов».

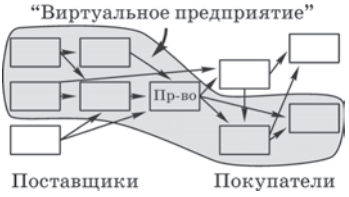


Рис. 4.5
Идея виртуального бизнеса

Рассмотренные выше методологии нашли проявление как в отдельных программных продуктах, так и в рамках Интранета как инструмента корпоративного управления.

Универсальный язык представления корпоративных знаний не зависит от конкретной предметной области и определяет грамматику и синтаксис. На данном этапе не существует единого языка описания и к этой категории может быть отнесен графический язык описания моделей данных, сетевых графиков, алгоритмов и др. Задачей универсального языка представления корпоративных знаний является: унификация представления знаний, однозначное толкование знаний, разбиение процессов обработки знаний на простые процедуры, допускающие автоматизацию.

Модели представления определяют специфику деятельности организации. Знания этого уровня являются метаданными, описывающими первичные данные.

Фактические знания отображают конкретные предметные области и являются первичными данными.

Интранет дает ощутимый экономический эффект в деятельности организации, что связано в первую очередь с резким улучшением качества потребления информации и ее прямым влиянием на производственный процесс. Для информационной системы организации ключевыми становятся понятия: «публикация информации», «потребители информации», «представление информации». Дальнейшее развитие корпоративного управления связано с моделями MRP/ERP и управления жизненным циклом изделия

4.2. МОДЕЛИ ПЛАНИРОВАНИЯ МАТЕРИАЛЬНЫХ И ФИНАНСОВЫХ РЕСУРСОВ (MRP/ERP)

Модели MRP/ERP с момента возникновения развивались эволюционно. С целью оптимального управления производством в середине 1960-х гг. APICS сформули-

ровало принципы управления материальными запасами предприятия. Эти принципы легли в основу концепции MRP (Material Requirement Planning — планирования материальных потребностей), основными положениями которой являются [8]:

- описание производственной деятельности как потока взаимосвязанных заказов;
- учет ограничения ресурсов при выполнении заказов;
- обеспечение минимизация производственных циклов и запасов;
- формирование заказов снабжения и производства на основе реализации и производственных графиков;
- привязка движения заказов к экономическим показателям;
- завершение выполнения заказа к тому моменту, когда он необходим.

Развитие вычислительных средств и наличие концепции привело к тому, что в 1970-х гг. стали появляться первые автоматизированные системы, реализующие MRP-концепцию.

Методика MRP декларирует, какие процессы учета и управления должны быть реализованы на предприятии, в какой последовательности они должны выполняться, и содержит рекомендации о том, как они должны выполняться.

Дальнейшее развитие концепции MRP шло по пути расширения функциональных возможностей предприятия в сторону более полного удовлетворения потребностей клиентов и снижения производственных издержек. В результате, в конце 1970-х гг. концепция MRP была дополнена положениями о формировании производственной программы в масштабах всего предприятия и контроля ее выполнения на уровне подразделений (Closed Loop MRP или, другими словами, воспроизведение замкнутого цикла в MRP-системах). Следующим шагом стало появление концепции MRPII (планирование производственных ресурсов — Manufacturing Resource Planning), основным содержанием которой стало прогнозирование, планирование и контроль производства по всему циклу, начиная

от закупки сырья и заканчивая отгрузкой товара потребителю.

MRPII (Manufacturing Resource Planning — планирование производственных ресурсов) представляет собой методологию, направленную на эффективное управление всеми ресурсами производственного предприятия. В общем случае она обеспечивает решение задач планирования деятельности предприятия в натуральных единицах, финансовое планирование в денежном выражении. Эта методология представляет собой набор проверенных на практике принципов, моделей и процедур управления и контроля, использование которых способствует улучшению показателей экономической деятельности предприятия.

Стандарт APICS на системы класса MRPII содержит описание 16 групп функций системы:

1) Sales and Operation Planning (Планирование продаж и производства);

2) Demand Management (Управление спросом);

3) Master Production Scheduling (Составление плана производства);

4) Material Requirement Planning (Планирование материальных потребностей);

5) Bill of Materials (Спецификации продуктов);

6) Inventory Transaction Subsystem (Управление складом);

7) Scheduled Receipts Subsystem (Плановые поставки);

8) Shop Flow Control (Управление на уровне производственного цеха);

9) Capacity Requirement Planning (Планирование потребностей в мощностях);

10) Input/output control (Контроль входа/выхода);

11) Purchasing (Материально-техническое снабжение);

12) Distribution Resource Planning (Планирование ресурсов распределения);

13) Tooling Planning and Control (Планирование и управление инструментальными средствами);

14) Financial Planning (Управление финансами);

15) Simulation (Моделирование);

16) Performance Measurement (Оценка результатов деятельности).

По мере накопления опыта моделирования производственных и непроизводственных операций содержание этих групп уточняется, постепенно охватывая все больше функций. При этом следует иметь в виду, что перечисленный функциональный состав относится только к управлению производственными ресурсами предприятия.

Стандарт MRPII делит сферы отдельных функций (процедур) на два уровня: необходимый и опциональный. Для отнесения к классу MRPII автоматизированные системы управления должны выполнять определенный объем необходимых (основных) функций (процедур).

Состав функциональных модулей и их взаимосвязи имеют глубокое обоснование с позиции теории управления. Они обеспечивают интеграцию функций планирования и согласование различных процессов управления во времени и пространстве. Представленный набор модулей является неизбыточным, что позволяет в основном сохранять его неизменным в системах следующих поколений. Более того, многие понятия, методы и алгоритмы, заложенные в функциональные модули MRPII, остаются неизменными в течение длительного времени и входят в качестве элементов в системы следующих поколений. По этой причине методологию MRPII можно считать базовой.

Для каждого уровня планирования MRPII характерны такие параметры, как степень детализации плана, горизонт планирования, вид условий и ограничений. Эти параметры для одного и того же уровня MRPII могут изменяться в широком диапазоне в зависимости от свойств производственного процесса на предприятии. С другой стороны, в зависимости от характера производственного процесса возможно применение на каждом отдельном предприятии определенного набора функциональных модулей MRPII. Таким образом, MRPII является гибкой и многофункциональной системой, применение которой возможно в широком спектре условий.

Принадлежность КИС к классу MRPII должна означать функциональную поддержку программным обеспечением выполнения следующего цикла: «планирование заказов → планирование потребности в сырье и материалах → планирование производственных ресурсов → контроль над исполнением производственной программы → обратная связь».

В общем виде система управления предприятием, построенная в соответствии со стандартом MRPII, имеет следующий вид (рис. 4.6). Дадим краткую характеристика вышеперечисленных функциональных блоков MRPII.

Бизнес-планирование. Процесс формирования плана предприятия наиболее высокого уровня. Планирование долгосрочное, план составляется в стоимостном выражении. Наименее формализованный процесс выработки решений.

Планирование спроса. Процесс прогнозирования (планирования) спроса на определенный период.

Планирование продаж и производства. Бизнес-план и план спроса преобразуются в планы продаж основных видов продукции (как правило, от 5 до 10). При этом производственные мощности могут не учитываться или учитываться укрупненно. План носит среднесрочный характер.

План продаж по видам продукции преобразуется в объемный или объемно-календарный план производства видов продукции. Под видом здесь понимаются семейства однородной продукции. В этом плане впервые в качестве планово-учетных единиц выступают изделия, но представления о них носят усредненный характер. Например, речь может идти о всех моделях телевизоров с одинаковым размером экрана, выпускаемых на заводе (без уточнения моделей). Часто этот модуль объединяется с предыдущим.

План-график выпуска продукции. План производства преобразуется в график выпуска продукции. Как правило, это среднесрочный объемно-календарный план, задающий количества конкретных изделий (или партий) со сроками их изготовления.

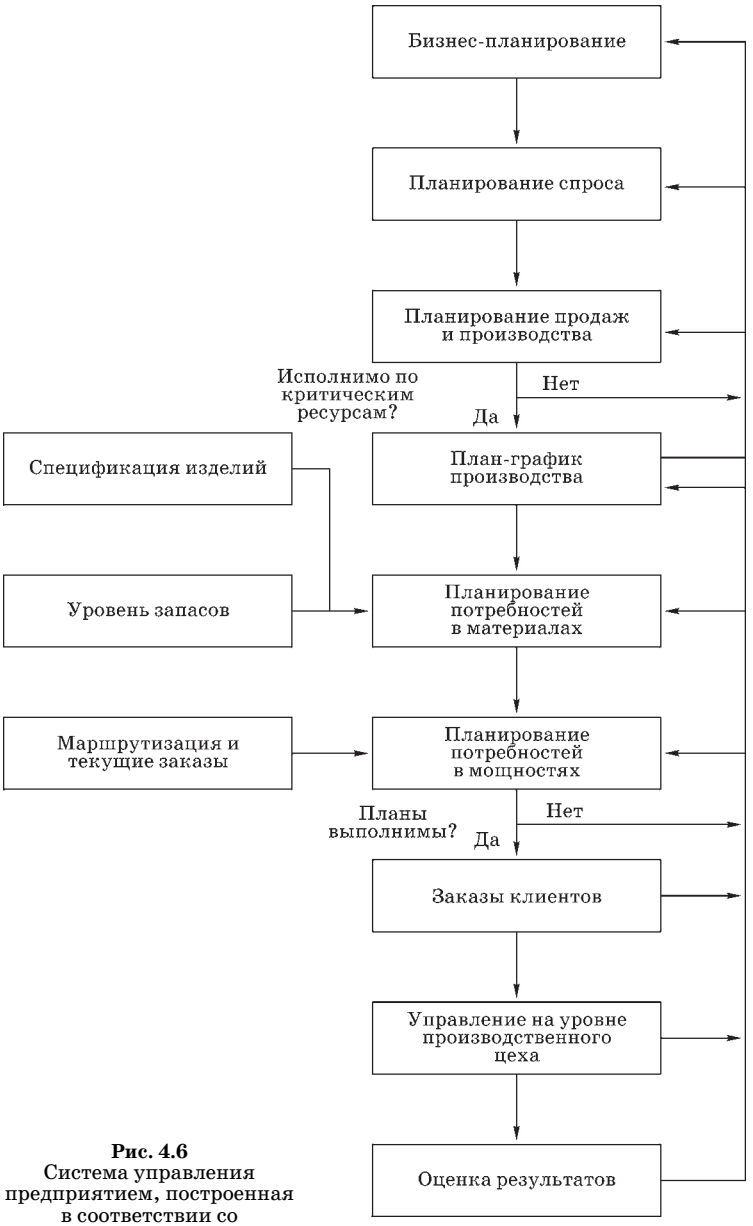


Рис. 4.6
Система управления предприятием, построенная в соответствии со стандартом MRPII

Планирование потребностей в материальных ресурсах. В ходе планирования на этом уровне определяются в количественном выражении и по срокам потребности в материальных ресурсах, необходимых для обеспечения графика выпуска продукции.

Входными данными для планирования потребностей в материалах являются спецификации изделий (состав и количественные характеристики комплектующих конкретного изделия) и размер текущих материальных запасов.

Планирование производственных мощностей. Как правило, в этом модуле выполняются расчеты по определению и сравнению располагаемых и потребных производственных мощностей. С наибольшими изменениями этот модуль может применяться не только для производственных мощностей, но и для других видов производственных ресурсов, способных повлиять на пропускную способность предприятия. Подобные расчеты, как правило, производятся после формирования планов практически всех предыдущих уровней с целью повышения надежности системы планирования. Иногда решение данной задачи включают в модуль соответствующего уровня. Входными данными при планировании производственных мощностей являются также маршрутизация выпускаемых изделий.

Управление заказами клиентов. Здесь реальные потребности клиентов сопоставляются с планами выпуска продукции.

Управление на уровне производственного цеха. Здесь формируются оперативные планы-графики. В качестве планово-учетных единиц могут выступать детали (партии), сборочные единицы глубокого уровня, детали (партии) операции и т. п. Длительность планирования невелика (от нескольких дней до месяца).

Оценка исполнения. По сути, в данном модуле оценивается реальное исполнение всех вышеперечисленных планов с тем, чтобы внести корректировки во все предыдущие циклы планирования.

Связь между уровнями в MRPII обеспечивается универсальной формулой, на которой строится система. Зада-

ча планирования на каждом уровне реализуется как ответ на четыре вопроса:

1) что необходимо выполнить;
2) что необходимо для этого;
3) что есть в наличии;
4) что необходимо иметь?

В роли ответа на первый вопрос всегда выступает план более высокого уровня. Этим и обеспечивается связь между уровнями. Структура ответов на последующие вопросы зависит от решаемой задачи.

Дальнейшее развитие систем MRPII связано с их перерастанием в системы нового класса — планирование ресурсов предприятия (Enterprise Resource Planning, ERP). Системы этого класса ориентированы на работу с финансовой информацией для решения задач управления большими корпорациями с разнесенными территориально ресурсами. Сюда включается все, что необходимо для получения ресурсов, изготовления продукции, ее транспортировки и расчетов по заказам клиентов. Помимо перечисленных функциональных требований, к системам ERP предъявляются и новые требования по применению графики, использованию реляционных баз данных, CASE-технологий для их развития, архитектуры вычислительных систем типа «клиент-сервер» и реализации их как открытых систем. Системы этого класса активно развиваются с конца 1980-х гг.

Следует отметить, что подход к решению задач планирования производства в системах ERP до недавнего времени оставался в основном неизменным, т. е. в том виде, в каком он утвердился в системах MRPII. Коротко его можно определить как подход, базирующийся на активном применении календарно-плановых нормативов на производственные циклы. Недостаток такого подхода состоит в том, что он вступает в противоречие с необходимостью оптимизации планирования. Элементы оптимизации планирования в традиционных MRPII/ERP системах встречаются только на нижнем уровне — при решении задач оперативного планирования с применением методов теории расписаний. С ростом мощностей вычислитель-

ных систем, внедрением MRPII/ERP, поиском новых более эффективных методов управления в условиях конкуренции с середины 1990-х гг. на базе систем MRPII/ERP появляются системы нового класса, которые получили название развитые системы планирования (Advanced Planning/Scheduling, APS). Для этих систем характерно применение экономико-математических методов для решения задач планирования с постепенным снижением роли календарно-плановых нормативов на производственные циклы.

Рост производительности и снижение незавершенного производства за счет внедрения таких систем объясняются тем, что при определении длительности производственного цикла в него не закладывается заранее усредненное время пребывания сырья в очередях. Данный подход особенно эффективен для сложного многономенклатурного производства. В то же время он требует существенного повышения профессионального уровня управленческого персонала.

Следующее направление в развитии автоматизации управления предприятиями состоит в интеграции систем MRPII/ERP с другими автоматизированными системами, имеющимися на предприятиях. В их числе — системы CAD/CAM, управления технологическими процессами и системами, системы финансовой отчетности и т. п. Системы такого класса получили название компьютерные интегрированные системы (Computer Integrated Manufacturing, CIM). Эти системы используются начиная с 1990-х гг.

Таким образом, система MRPII постоянно эволюционирует и совершенствуется. В каждый момент времени в концепциях MRPII/ERP можно выделить, условно, три слоя.

В первом слое находятся те методы и средства, которые проверены практикой и закреплены в виде стандартов.

Второй слой составляют достаточно устойчивые, часто применяемые методы и приемы, которые, однако, не носят обязательного характера. Эти методы и приемы можно обнаружить при более глубоком анализе функцио-

нальных структур. В качестве примеров можно привести методологию скользящего планирования в MPS/MRP, алгоритмы образования партий в MRP, правила приоритетов в SFC и многое другое.

Этот слой, жестко не регламентируемый, тем не менее, представляет собой довольно стройную систему взаимосвязанных идей и методов.

К третьему слою идей и методов MRPII/ERP следует отнести то новое, что вносят в свои базовые системы фирмы-производители программных продуктов. Реализованные на их основе новые информационные технологии представляют собой know-how фирм-разработчиков. Как правило, именно в этом слое можно обнаружить значительные отличия в продуктах различных фирм. Некоторые из новых технологий в состоянии оказывать серьезное влияние на эффективность построения крупных информационных систем.

Видное место среди идей и методов систем MRPII/ERP принадлежит специально разработанным методикам внедрения систем. Анализ литературы показывает, что на Западе сложилось устойчивое представление о том, в какой последовательности и какими методами следует внедрять системы типа MRPII/ERP. Тщательное планирование проектов по внедрению, организация деятельности коллективов, упор на переподготовку персонала всех уровней (особенно высшего уровня) — вот далеко не полный перечень условий достижения положительных результатов. Наличие мощной инфраструктуры и методологии построения систем способствовало достижению высокого уровня эффективности при внедрении систем управления типа MRPII/ERP на промышленных предприятиях. По некоторым оценкам, внедрение подобных систем способно привести к сокращению запасов на 8–30%, росту производительности труда на 8–27%, возрастанию количества заказов, выполненных в срок, на 7–20%.

На рисунке 4.7 представлены цели внедрения систем типа MRPII/ERP.

Дальнейшим развитием моделей MRPII/ERP является архитектура, ориентированная на сервисы (Service Ori-

Рис. 4.7
Цели внедрения систем типа MRP II/ERP

ental Architecture, SOA) и на сервисы бизнес-приложения (Service Oriental Business Application, SOBA). Развитие этих направлений связано с тем что существенным недостатком ERP является отраслевая специфика [20]. MRPII, служащая основой для многих ERP, — это методология планирования дискретных, большей частью машиностроительных производств. Ряд факторов привел к многочисленным неудачным попыткам внедрения подобных систем на других производствах и к проблеме адаптации программного продукта под конкретное производство. Ситуацию усугубило распространение таких методологий планирования, как заказное, гибкое производство (Agile Manufacturing) и экономичное производство (Lean Manufacturing). Таким образом, возникла проблема гибкости и универсальности интерфейса управления производством.

В основе идеологии SOA — интеграция множественных, одовременно работающих разных предприятий, территориально разнесенных бизнес-приложений. Это могут быть как вновь внедряемые, так и унаследованные, в том

числе традиционные ERP-системы. В определенном плане появление SOA — возврат к идеологии мэйнфреймов, но на другом качественном уровне. Если мейнфреймы объединяли вычислительные ресурсы, то SOA объединяют и позволяют использовать данные и приложения.

4.3. МОДЕЛИ УПРАВЛЕНИЯ ЖИЗНЕННЫМ ЦИКЛОМ ИЗДЕЛИЯ (PLM)

В настоящее время остро стоит проблема перехода от использования отдельных информационных технологий к созданию единых интегрированных комплексов. Одним из направлений является модель PLM (Product Lifecycle Management — управление жизненным циклом изделия).

Появление PLM связано с развитием электронного бизнеса, классификация которого, предложенная компанией Gartner Group, представлена на рисунке 4.8 [8].

Выделены четыре фазы развития: присутствие, взаимодействие, передача данных (транзакции) и трансформация бизнеса. Начиная с 2000 г. активно развиваются третья и четвертая фазы. Другая классификация основана на использовании данных, и развитие электронного

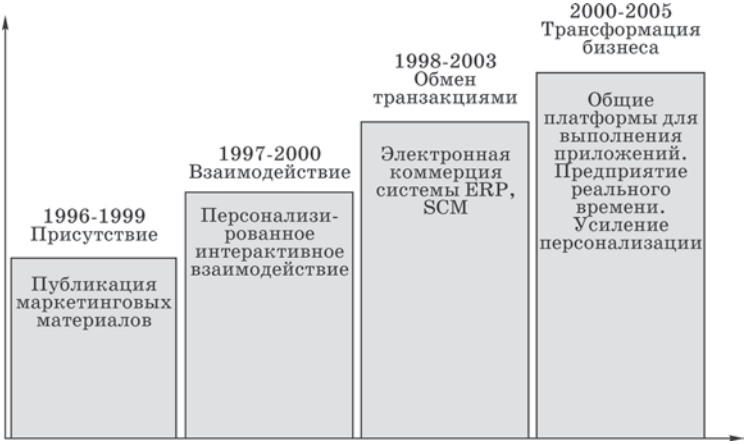


Рис. 4.8
Этапы развития электронного бизнеса

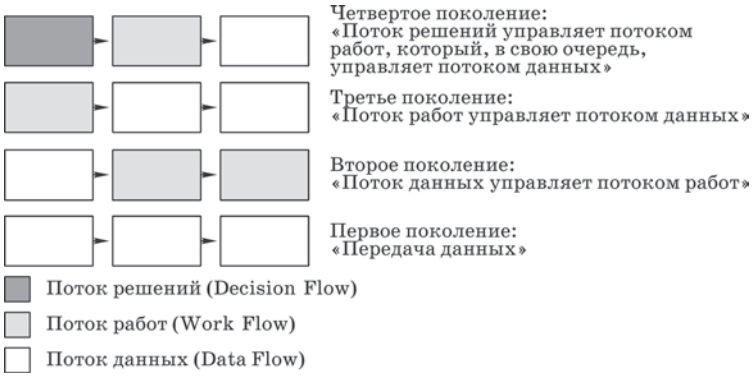


Рис. 4.9
Поколения электронного бизнеса

бизнеса рассматривается как восхождение от простого обмена данными к обеспечению средствами технологий потоков работ и потоков принятия решений (рис. 4.9).

Бурное развитие PLM прослеживается в следующем. Современный бизнес решает триединую задачу: во-первых, необходимо устанавливать более тесные и доверительные отношения с поставщиками и заказчиками, во-вторых, повышать уровень собственной операционной эффективности и, в-третьих, повышать конкурентоспособность выпускаемой продукции. Первая составляющая обеспечивается системами поддержки отношений, получающими все большее распространение — системами SCM (Supply Chain Management — система управления цепочками поставок) и CRM (Customer Relations Management — система управления отношениями с заказчиками), вторая — еще более популярными системами ERP, а вот третья пока не имеет достаточного комплексного информационного обеспечения. На то, чтобы занять это место, претендует подход, названный new PLM. Он заметно отличается от традиционного представления о том, что такое управление жизненным циклом изделий. Раньше под PLM (в узком смысле) чаще всего понимали то, что имело отношение к жизненному циклу материальных изделий, начиная от запуска в производство, регулирования объемов выпуска, определения

времени выпуска новых или обновленных изделий и, конечно же, сопровождение и сервис. Изменения в видении роли и места PLM произошли буквально в последние несколько лет. Теперь под этим термином понимают автоматизацию практически всех видов работ, которые составляет основу выпуска продукции — от проектирования до сбыта. Разные авторы расходятся в определениях; одни включают SCM, CRM и ERP в состав нового управления жизненным циклом изделий, другие считают эти системы взаимодополняющими (рис. 4.10).

В соответствии с триединой задачей «PLM по-новому» можно разделить на три взаимосвязанных составляющие управления жизненным циклом:

- жизненный цикл определения изделий (интеллектуальные активы предприятия);
- жизненный цикл производства (материальные активы предприятия);
- жизненный цикл операционной поддержки.

Первичным является жизненный цикл управления интеллектуальными активами; он начинается с оценки

Рис. 4.10
Взаимосвязь корпоративных приложений

пользовательских требований, выработки концепции продукта, а завершается, когда предприятие полностью отказывается от продукта, в том числе и от его сервисной поддержки.

PLM-решения — один из самых быстрорастущих и перспективных сегментов рынка ИТ-услуг, что в значительной степени связано со способностью PLM-приложений существенно сокращать расходы на проектирование и ускорять темпы выпуска продуктов на рынок. В числе факторов, способствующих принятию решений об использовании услуг PLM, — высокий уровень использования ИТ в производстве, сложность производственных процессов, а также разобщенность подразделений предприятия, ответственных за конструирование, производство, сбыт и обслуживание продуктов.

Одно из наиболее полных определений PLM состоит из четырех пунктов:

- стратегический подход к бизнесу, предлагающий непрерывный набор бизнес-решений, который поддерживает коллаборативный режим создания, управления, распределения и использования определения изделий (интеллектуальных активов предприятия);
- поддержка «расширенного представления о предприятии» (extended enterprise), в том числе поддержка процессоров проектирования, пользователей и партнеров;
- действие во времени от момента рождения концепции изделия до снятия его с производства и окончания сервисного периода;
- интеграция людей, процессов, систем и информации.

Важно подчеркнуть, что в этом определении PLM рассматривается не как часть или части технологий, а как бизнес-подход, цель которого состоит в поиске ответов на вопросы: «Как работает бизнес? Что создается?»

На основании этого определения выделяются три основные концепции PLM:

- возможность универсального, безопасного и управляемого способа доступа и использования информации, определяющей изделия;

- поддержание целостности информации, определяющей изделие, на протяжении всего его жизненного цикла изделия;
- управление и поддержка бизнес-процессов, используемых при создании, распределении и использовании подобной информации.

Концепция PLM охватывает все этапы жизненного цикла изделия (рис. 4.11) и подразумевает управление данными, получаемыми от следующих компонент:

- PDM — Product Data Management — управление данными об изделиях;
- CAM — Computer-Aided Manufacturing — система автоматизированной подготовки производства;
- CAE — Computer-Aided Engineering — система автоматизированного инженерного анализа;
- CAD — Computer-Aided Design — система автоматизированного проектирования;
- MPM — Manufacturing Process Management — планирование и моделирование производства с использованием обрабатывающих центров с ЧПУ, роботов и т. п.;
- BOM — Bill Of material Management — спецификация инженерная;
- CRM — Customer Relationship Management – управление отношениями с клиентами.

Одним из важнейших компонентов PLM является система управления данными об изделии (PDM), обеспечивающая обмен данными о составе изделия и вносимых в него изменениях и позволяющая создавать и поддерживать множество взаимосвязанных спецификаций изде-

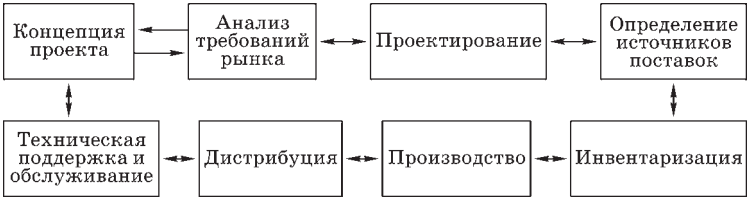


Рис. 4.11
Этапы жизненного цикла изделия

лия, благодаря чему пользователь получает согласованное представление о составе изделия по ходу работы над ним.

Система PDM должна реализовывать следующие функции:

1) функция управления составом изделия, которая может быть представлена совокупностью следующих возможностей: ведение спецификаций; многоуровневые спецификации, отображающие как дерево сборки изделия, так и полный набор конструкторских, технологических и прочих атрибутов; динамический просмотр иерархически организованной информации; отслеживание принадлежности каждой детали, сборки, узла, изделия модельному ряду; определение условий применимости и отображение ограничений применимости; ведение протоколов изменения версий вплоть до версий каждой детали; отслеживание действия внесенных изменений и модификаций;

2) функция отслеживания ссылок на документы электронного архива, соответствующих каждой детали, сборке, узлу, изделию. Получение данных непосредственно из электронного архива или из САПР сборок;

3) функция сравнения структур изделий, сопровождение и обслуживание информации об изделии с учетом специфики различных подразделений, включая предприятия-соисполнители (поставщики комплектующих, субподрядчики) и внешние торговые площадки;

4) дополнительные сервисные функции представления трехмерных данных (геометрические электронные модели изделия, детали, сборки). Возможности визуализации в системе PDM не должны зависеть от типов и форматов исходных данных, что особенно актуально для предприятий, использующих разнотипные САПР. Сама визуализация должна поддерживать рендеринг, анимацию, построение сечений и разрезов, ведение комментариев на изображении и т. д.

Рассмотрим содержание каждого из этапов жизненного цикла.

Концепция проекта. На основе анализа требований рынка формируется общая идея нового изделия или, что

случается значительно чаще, концепция усовершенствований в проекте уже существующего продукта. Система PLM предоставляет информацию, которая может использоваться для анализа жизнеспособности полученной концепции.

Анализ требований рынка. Производитель должен понять, насколько востребован рынком новый продукт, и оценить выполнимость этих требований. На этом этапе система PLM используется для извлечения данных из различных информационных систем, которые могут способствовать получению более точной картины.

Проектирование. Конструкторы создают проект нового изделия — соответствующие САПР- и PDM-решения являются интегральной частью PLM-решения. При проектировании используется вся необходимая дополнительная информация, поставщиком которой являются PLM-модули, включая факторы, связанные с послепродажным обслуживанием изделия, информацию о предпочтениях заказчика, данные о производственных возможностях и т. д.

Определение источников поставок (*PLM-sourcing*). Отдел закупок должен провести предварительную работу по поиску источников приобретения необходимых для производства изделия деталей, материалов, компонентов, оборудования и т. д. Задача систем PLM — предоставить достоверные данные о доступности тех или иных деталей/компонентов/материалов, их стоимости, потенциальных поставщиках и возможных альтернативных источниках.

Производство. В соответствии с определенными на этапе проектирования спецификациями и с использованием полученных на этапе поставок деталей и материалов производится продукт. Реализованные в PLM специальные методы контроля качества позволяют гарантировать соответствие производимого изделия заданным спецификациям.

Дистрибуция. Готовое изделие поставляется либо дистрибутору, который размещает его на своем складе до поступления соответствующего заказа, либо непосредственно заказчику. Полученные из системы PLM исторические данные о потребностях рынка помогают производителю

свести к минимуму число уровней инвентаризации готовой продукции.

Техническая поддержка и обслуживание. На этом этапе выполняются техническое сопровождение, обслуживание и ремонт — в течение гарантийного срока или как дополнительно оплачиваемый сервис. PLM позволяет учесть различную информацию об изделии, поступающую на этом этапе жизненного цикла, при разработке последующих проектов и тем самым способствует повышению привлекательности продукции для клиентов.

В PLM доступ к данным организован на ролевой основе. Система позволяет предоставлять пользователю информацию в форме, соответствующей выполняемым им функциям в жизненном цикле изделия: трехмерные модели, схематические диаграммы, инженерные спецификации (Bill of Materials, BOM), календарные планы или прогнозы на основе анализа требований рынка. Этим обеспечивается работа каждого пользователя в привычной ему среде. Конструкторы и технологи в среде САПР (CAD/CAM/CAE), а сотрудник маркетингового подразделения сможет получить из системы представление трехмерной сборки, пригодное для размещения в рекламной брошюре. С помощью информации, которую интегрирует система PLM, даже не обладая специальными техническими знаниями, сотрудники отдела закупок смогут заниматься поиском нужных деталей и выбором оптимальных каналов поставки непосредственно по данным, поступающим из конструкторских подразделений.

Важным преимуществом системы PLM, объединяющей все структурные подразделения предприятия, является возможность решения многих проблем на этапе проектирования. В результате появляется возможность увязать проект со сложностями производственного цикла и задачами закупок, что позволит сохранить оптимальную цену и высокое качества продукции.

Другой характерной чертой PLM является эффективная работа с поставщиками, внешними контрагентами и смежниками за счет безбумажных форм обмена информацией.

Знания проблем технического сопровождения готовой продукции, ее гарантийного или платного обслуживания оказывают влияние на последующие проекты. PLM предоставляет производителю возможность получения таких данных, их анализа и устранения выявленных проблем в следующих проектах. Это позволит удовлетворить запросы клиентов, повысить имидж и конкурентоспособность производителя.

В целом, преимущества, которые дает PLM-решение, можно сформулировать следующим образом:

- ускорение вывода новой продукции на рынок, благодаря привлечению к процессам проектирования в реальном времени всех заинтересованных участников, включая внешних поставщиков и заказчиков;
- совершенствование характеристик разрабатываемой продукции и повышение качества, обнаружение недостатков и ограничений проекта на самых ранних стадиях;
- увязка проектирования и производственных процессов: инженеры-технологи становятся интегральной частью команды проектировщиков, благодаря чему проект сразу создается с учетом специфики производственного процесса, включая тестирование, контроль качества и т. д.;
- учет и использование опыта других проектов;
- реализация новой бизнес-модели «виртуального предприятия» — к процессу проектирования и производства привлекаются поставщики либо работы определенного этапа жизненного цикла продукции передаются на аутсорсинг внешним компаниям.

4.4. ИНТЕГРИРОВАННАЯ ИНФОРМАЦИОННАЯ СРЕДА УПРАВЛЕНИЯ ЖЦИ

На рисунке 4.12 представлена диаграмма вариантов использования, описывающая состав и зависимости процессов ЖЦИ [37].

Общее количество объектов, используемых для описания ЖЦИ, составляет несколько тысяч. В соответствии

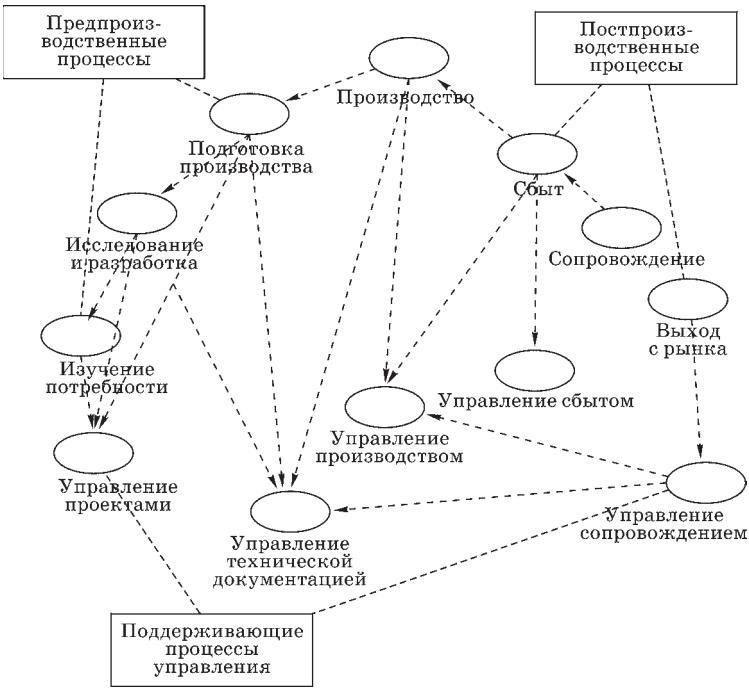


Рис. 4.12
Зависимости процессов управления ЖЦИ

с природой объектов условно первый уровень классификации можно представить в виде диаграммы классов, изображенной на рисунке 4.13.

Сущность «Производственный элемент» позволяет вести справочники предметов и средств труда, участвующих в ЖЦИ. Сущность «Процесс» позволяет специфицировать процессы ЖЦИ. Сущность «Экземпляр производственного элемента» позволяет специфицировать конкретные экземпляры производственных элементов в процессе производства и постпроизводственных стадиях ЖЦИ. Сущность «Хозяйственная операция» позволяет описывать базовые операции с экземплярами производственных элементов. Сущность «Накопитель» позволяет решать задачи управления остатками и запасами. Сущность «Субъект хозяйственной деятельности» позволяет

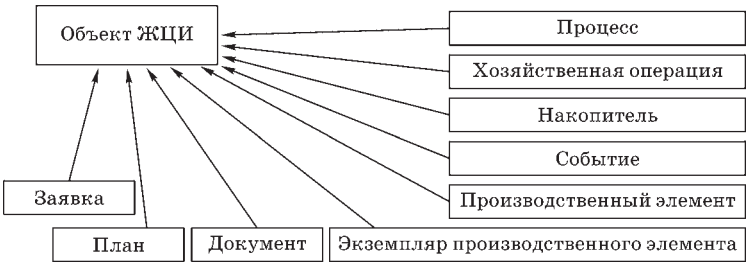


Рис. 4.13
Классификации объектов ЖЦИ

специфицировать все организационные звенья, участвующие в ЖЦИ. Сущность «Событие» специфицирует логику процессов ЖЦИ.

Сущность «Экономический элемент» позволяет специфицировать экономику ЖЦИ. Сущность «Документ» позволяет специфицировать документы, сопровождающие ЖЦИ. Сущность «Заявка» позволяет специфицировать взаимоотношения субъектов хозяйственной деятельности. Сущность «План» позволяет специфицировать планируемые результаты по управлению ЖЦИ.

Управление жизненным циклом сложных изделий является одним из важнейших процессов деятельности современных компаний. Качество данного процесса во многом связано с созданием и поддержанием интегрированной информационной среды. Введем несколько необходимых определений. Базовым структурным компонентом интегрированной информационной среды является изделие. Построение интегрированной информационной среды основывается на построении и сопровождении информационной модели производственной деятельности, которая включает в себя, в частности, информационную модель изделия.

ИМ — *информационная модель изделия* (product information model) содержит абстрактное описание фактов и инструкций об изделии [24].

ИИС — *интегрированная информационная среда* (integrated information environment) — это совокупность распределенных баз данных, содержащих сведения об изделиях, производственной среде, ресурсах и процессах

предприятия, обеспечивающая корректность, актуальность, сохранность и доступность данных тем субъектам производственно-хозяйственной деятельности (ПД), участвующим в осуществлении ЖЦИ (далее — субъекты ПД), кому это необходимо и разрешено. Все сведения (данные) хранятся в виде информационных объектов, которые составляют основу предметной области производственной деятельности информационной модели.

ИО — *информационный объект* (information object) — это совокупность данных и программного кода, обладающая свойствами (атрибутами) и методами, позволяющими определенным образом обрабатывать данные. ИО является самостоятельной единицей применения и хранения в ИИС. Описание информационных объектов основывается на понятии класса и экземпляра класса.

КИО — *класс информационных объектов* (class of information objects/entity) определяет свойства объектов до тех пор, пока им не присвоены конкретные значения. Класс может порождать экземпляры, наследующие его свойства и методы.

Экземпляр класса (instance) ИО, получающийся из КИО присвоением свойствам конкретных значений.

Интегрированная информационная среда должна обеспечить информационное взаимодействие между всеми участниками ПД.

Информационное взаимодействие (information interaction) — это совместное использование данных, находящихся в ИИС, и обмен данными, осуществляемые субъектами ПД, в соответствии с установленными правилами.

Главным компонентом ИИС является база интегрированных данных, включающая в себя следующие части: общая база данных об изделиях; общая база данных о предприятии.

ОБДИ — *общая база данных об изделиях* (common product database) — это часть ИИС — хранилище ИО, содержащих в произвольном формате информацию, требуемую для выпуска и поддержки технической документации, необходимой на всех стадиях ЖЦИ, для всех изделий, выпускаемых предприятием. Каждый ИО в ОБДИ

идентифицируется уникальным кодом и может быть извлечен из ОБДИ для выполнения действий с ним. ОБДИ обеспечивает информационное обслуживание и поддержку деятельности:

- заказчиков (владельцев) изделия;
- разработчиков (конструкторов), технологов, управленческого и производственного персонала предприятия-изготовителя;
- эксплуатационного и ремонтного персонала заказчика.

ОБДИ может состоять из нескольких разделов: нормативно-справочного; долговременного и актуального.

Нормативно-справочный раздел ОБДИ (standard andreference section) — это раздел ОБДИ, хранящий ИО, содержащие данные:

- о конструкционных материалах;
- о нормализованных деталях (нормалях);
- о нормативных документах на покупные комплектующие изделия;
- о нормативных документах на детали собственного изготовления;
- о нормативных документах на расчетные методы;
- о государственных, международных и внутренних НД;
- о прочих нормативных документах.

Долговременный раздел ОБДИ (permanent section) — это раздел ОБДИ, хранящий ИО, содержащие данные, аккумулирующие собственный опыт предприятия, в том числе данные:

- о ранее выполненных готовых проектах (архив);
- о типовых узлах и агрегатах собственного производства;
- о типовых деталях собственного производства;
- о типовых конструктивно-технологических элементах деталей;
- о типовых и групповых технологических процессах;
- о типовой технологической оснастке и инструменте;
- о готовых и типовых расчетных методиках и математических моделях изделий собственной разработки;
- о прочих готовых и типовых решениях.

Долговременный раздел ОБДИ обновляется по мере создания новых технических решений, признанных типовыми и пригодными для дальнейшего использования.

Актуальный раздел ОБДИ (actual section) — это раздел ОБДИ, хранящий ИО, содержащие данные об изделиях, находящихся на различных стадиях ЖЦИ:

- о конструкции и технологии изготовления изделий;
- о конкретных экземплярах изделий в производстве;
- о конкретных экземплярах изделий, находящихся на постпроизводственных стадиях ЖЦИ.

ОБДП — *общая база данных о предприятии* (common enterprise database) — это часть ИИС — хранилище ИО, содержащих в произвольном формате данные о финансово-экономическом состоянии предприятия, его внешних связях, производственно-технологической среде, действующей на предприятии системе качества и т. д.

База данных по экономике и финансам (economic andfinance database) — это раздел ОБДП, хранящий ИО, содержащие сведения:

- о конъюнктуре рынка изделий, включая цены и их динамику;
- о состоянии финансовых ресурсов предприятия;
- о ситуации на фондовом и финансовом рынках (курсы акций предприятия, биржевые индексы, процентные ставки, валютные курсы и т. д.);
- о реальном и прогнозируемом портфеле заказов;
- финансово-экономического и бухгалтерского характера.

База данных о внешних связях предприятия (enterprise external relationship database) — это раздел ОБДП, хранящий ИО, содержащие сведения о фактических и возможных поставщиках и заказчиках. Формируется и используется в процессе маркетинговых исследований.

База данных о производственно-технологической среде предприятия (enterprise manufacturing and technologi calenvironment database) — это раздел ОБДП, хранящий ИО, содержащие сведения:

- о производственной структуре предприятия;

- о технологическом и контрольно-измерительном оборудовании;
- о транспортно-складской системе предприятия;
- об энерговооруженности предприятия;
- о кадрах;
- прочие данные о предприятии.

База данных о системе качества (quality management system database) — это раздел ОБДП, хранящий ИО, содержащие сведения:

- о структуре действующей на предприятии системы качества;
- о действующих на предприятии стандартах по качеству;
- о международных и российских НД по качеству;
- о должностных инструкциях в области качества;
- прочая информация по системе качества.

Часть данных об изделии определяется как технические данные.

Данные технические (technical data) — это информация о свойствах и характеристиках технической продукции в электронной форме.

Особо следует выделить такой класс информационных объектов, как документ электронный (electronic document) — это информационный объект, состоящий из двух частей: реквизитной части, содержащей идентифицирующие атрибуты (имя, время и место создания, данные об авторе и т. д.) и электронную цифровую подпись; содержательной части, включающей текстовую, числовую и/или графическую информацию, которая обрабатывается в качестве единого целого.

Документ технический электронный (technical electronic document) — электронный документ, содержательная часть которого включает технические данные.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие выделяют группы методов управления?
2. Какая связь между MRP и MPS?
3. В чем отличие модели MRPII от MRP?

4. Каково соотношение между понятиями CSSP, ERP и стадиями жизненного цикла товара?
5. В чем идея виртуального бизнеса?
6. Раскройте содержание Интранета.
7. Каковы основные принципы концепции MRP?
8. В чем заключается основное содержание MRPII?
9. Перечислите основные группы функций системы MRPII.
10. Охарактеризуйте функциональные блоки MRPII.
11. Какие слои можно выделить в концепциях MRPII/ERP?
12. Какова взаимосвязь моделей MRPII/ERP и архитектуры, ориентированная на сервисы?
13. С каким фактором связано появление модели PLM?
14. Назовите составляющие управления жизненным циклом?
15. Дайте определение PLM.
16. Перечислите основные компоненты PLM.
17. Дайте краткую характеристику основных этапов жизненного цикла.
18. Укажите основные преимущества PLM.
19. Раскройте содержание в виде диаграммы классов процессов ЖЦИ.
20. Раскройте содержание информационной модели изделия.
21. Перечислите основные компоненты базы интегрированных данных.

ГЛАВА 5

ИНСТРУМЕНТАЛЬНАЯ СРЕДА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

В результате освоения данной темы студент должен

1) знать:

- основные и дополнительные функции операционных систем, типы архитектур ядер операционных систем;
- конструкцию языков программирования, виды (парадигмы) языков по областям применения, реализацию языков программирования, классификацию программных сред;
- архитектуру ЭВМ, структуру и состав различных типов ЭВМ;
- назначение методических средств информационных технологий, их состав, структуру, решаемые задачи;

2) уметь:

- использовать инструментальные средства для реализации информационных систем и технологий;
- применять инструментальные средства в процессе эксплуатации и сопровождения информационных систем;

3) владеть:

- методологией использования инструментальных средств в процессах проектирования, внедрения, сопровождения и модернизации информационных систем;
- навыками использования инструментальных средств для решения конкретных задач в процессе эксплуатации информационных систем.

5.1. ПРОГРАММНЫЕ СРЕДСТВА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Программные средства информационных технологий можно разделить на две большие группы: базовые и прикладные [25].

Базовые программные средства относятся к инструментальной страте информационных технологий и включают в себя:

- операционные системы (ОС);
- языки программирования;
- программные среды;
- системы управления базами данных (СУБД).

Прикладные программные средства предназначены для решения комплекса задач или отдельных задач в различных предметных областях.

Операционные системы

ОС предназначены для управления ресурсами ЭВМ и процессами, использующими эти ресурсы. В настоящее время существуют две основные линии развития ОС: Windows и Unix. Генеалогические линии данных ОС развивались следующим образом:

CP/M QDOS 86-DOS MS-DOS Windows

Multics UNIX Minix Linux

В свою очередь, каждый элемент линии имеет свое развитие, например Windows развивался в такой последовательности: Windows 95, 98, Me, NT, 2000 и далее. Соответственно Linux развивался следующим образом: версии 0.01, 0.96, 0.99, 1.0, 1.2, 2.0, 2.1, 2.1.10 и далее. Каждая версия может отличаться добавлением новых функциональных возможностей (сетевые средства, ориентация на разные процессоры, многопроцессорные конфигурации и др.).

Основные функции ОС:

- для выполнения по запросу программ достаточно тех элементарных (низкоуровневых) действий, которые являются общими для большинства программ и часто встречаются почти во всех программах (ввод и вывод

данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.);
- загрузка программ в оперативную память и их выполнение;
- стандартизованный доступ к периферийным устройствам;
- управление оперативной памятью (распределение между процессами, организация виртуальной памяти);
- управление доступом к данным на энергонезависимых носителях (таких как жесткий диск, оптические диски и др.), организованным в той или иной файловой системе;
- обеспечение пользовательского интерфейса;
- сетевые операции, поддержка стека сетевых протоколов.

Дополнительные функции ОС:
- параллельное или псевдопараллельное выполнение задач (многозадачность);
- эффективное распределение ресурсов вычислительной системы между процессами;
- разграничение доступа различных процессов к ресурсам;
- организация надежных вычислений (невозможность одного вычислительного процесса намеренно или по ошибке повлиять на вычисления в другом процессе) основана на разграничении доступа к ресурсам;
- взаимодействие между процессами: обмен данными, взаимная синхронизация;
- защита самой системы, а также пользовательских данных и программ от действий пользователей (злонамеренных или по незнанию) или приложений;
- многопользовательский режим работы и разграничение прав доступа (аутентификация, авторизация).

Таким образом, современные универсальные ОС можно охарактеризовать прежде всего как:
- использующие файловые системы (с универсальным механизмом доступа к данным);
- многопользовательские (с разделением полномочий);
- многозадачные (с разделением времени).

Многозадачность и распределение полномочий требуют определенной иерархии привилегий компонентов самой ОС. В составе ОС различают три группы компонентов:

- ядро, содержащее планировщик; драйверы устройств, непосредственно управляющие оборудованием; сетевая подсистема, файловая система;
- системные библиотеки;
- оболочка с утилитами.

Большинство программ, как системных (входящих в ОС), так и прикладных, исполняются в непривилегированном (пользовательском) режиме работы процессора и получают доступ к оборудованию (и при необходимости к другим ресурсам ядра, а также ресурсам иных программ) только посредством системных вызовов. Ядро исполняется в привилегированном режиме: именно в этом смысле говорят, что ОС (точнее, ее ядро) управляет оборудованием.

В определении состава ОС значение имеет критерий операциональной целостности (замкнутости): система должна позволять полноценно использовать (включая модификацию) свои компоненты. Поэтому в полный состав ОС включают и набор инструментальных средств (от текстовых редакторов до компиляторов, отладчиков и компоновщиков).

Ядро ОС — центральная часть операционной системы (ОС), обеспечивающая приложениям координированный доступ к ресурсам компьютера, таким как процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и вывода информации, переводя команды языка приложений на язык двоичных кодов, которые понимает компьютер. Также обычно ядро предоставляет сервисы файловой системы и сетевых протоколов.

Объекты ядра ОС:

- процессы;
- файлы;
- события;
- потоки;
- семафоры;

- мьютексы;
- каналы.

Рассмотрим типы архитектур ядер ОС [38].

Монолитное ядро предоставляет богатый набор абстракций оборудования. Все части монолитного ядра работают в одном адресном пространстве. Это такая схема операционной системы, при которой все компоненты ее ядра являются составными частями одной программы, используют общие структуры данных и взаимодействуют друг с другом путем непосредственного вызова процедур.

Достоинства: скорость работы, упрощенная разработка модулей.

Недостатки: нарушение работоспособности всей системы от сбоя в одном из компонентов, повышенные требования к объему оперативной памяти, небходимость полной перекомпиляции ядра при изменении состава аппаратного обеспечения компьютера.

Примеры ОС с монолитными ядрами: традиционные ядра UNIX (такие как BSD), Linux; ядро MS-DOS, ядро KolibriOS.

Альтернативой монолитным ядрам считаются архитектуры, основанные на микроядрах.

Модульное ядро ОС — современная, усовершенствованная модификация архитектуры монолитных ядер операционных систем компьютеров.

В отличие от «классических» монолитных ядер, считающихся ныне устаревшими, модульные ядра, как правило, не требуют полной перекомпиляции ядра при изменении состава аппаратного обеспечения компьютера. Вместо этого модульные ядра предоставляют тот или иной механизм подгрузки модулей ядра, поддерживающих то или иное аппаратное обеспечение (например, драйверов). Все модули ядра работают в адресном пространстве ядра и могут пользоваться всеми функциями, предоставляемыми ядром. Поэтому модульные ядра продолжают оставаться монолитными. Модульность ядра осуществляется на уровне бинарного образа, а не на архитектурном уровне ядра, так как динамически подгружаемые модули за-

гружаются в адресное пространство ядра и в дальнейшем работают как интегральная часть ядра.

Модульные ядра предоставляют особый программный интерфейс (API) для связывания модулей с ядром, для обеспечения динамической подгрузки и выгрузки модулей. В свою очередь, не любая программа может быть сделана модулем ядра: на модули ядра накладываются определенные ограничения в части используемых. Не все части ядра могут быть сделаны модулями. Некоторые части ядра всегда обязаны присутствовать в оперативной памяти и должны быть жестко «вшиты» в ядро. Также не все модули допускают динамическую подгрузку (без перезагрузки ОС). Примером может служить VFS — виртуальная файловая система, совместно используемая многими модулями файловых систем в ядре Linux.

Микроядро ОС — это минимальная реализация функций ядра операционной системы.

Классические микроядра предоставляют лишь очень небольшой набор низкоуровневых примитивов, или системных вызовов, реализующих базовые сервисы операционной системы.

К ним относятся:

- управление адресным пространством оперативной памяти;
- управление адресным пространством виртуальной памяти;
- управление процессами и потоками (нитями);
- средства межпроцессной коммуникации.

Классическим примером микроядерной системы является Symbian OS. Это пример распространенной и отработанной микроядерной (а начиная с версии Symbian OS v8.1, и наноядерной) операционной системы.

В отличие от Windows NT, создателям Symbian OS удалось совместить эффективность и концептуальную стройность, несмотря на то что современные версии этой системы предоставляют обширные возможности, в том числе средства для работы с потоковыми данными, стеками протоколов, критичными к латентности ядра, графикой и видео высокого разрешения.

Разработчики Symbian вынесли практически все прикладные (т. е. выходящие за пределы компетенции ядра) задачи в модули-серверы, функционирующие в пользовательском адресном пространстве.

Экзоядро ОС — ядро операционной системы компьютеров, предоставляющее лишь функции для взаимодействия между процессами и безопасного выделения и освобождения ресурсов.

В традиционных операционных системах ядро предоставляет не только минимальный набор сервисов, обеспечивающих выполнение программ, но и большое количество высокоуровневых абстракций для использования разнородных ресурсов компьютера: оперативной памяти, жестких дисков, сетевых подключений. В отличие от них, ОС на основе экзоядра предоставляет лишь набор сервисов для взаимодействия между приложениями, а также необходимый минимум функций, связанных с защитой: выделение и высвобождение ресурсов, контроль прав доступа, и т. д. Экзоядро не занимается предоставлением абстракций для физических ресурсов — эти функции выносятся в библиотеку пользовательского уровня (так называемую libOS).

Основная идея операционной системы на основе экзоядра состоит в том, что ядро должно выполнять лишь функции координатора для небольших процессов, связанных только одним ограничением — экзоядро должно иметь возможность гарантировать безопасное выделение и освобождение ресурсов оборудования. В отличие от ОС на основе микроядра, ОС, базирующиеся на экзоядре, обеспечивают гораздо большую эффективность за счет отсутствия необходимости в переключении между процессами при каждом обращении к оборудованию.

Архитектуры на основе экзоядер являются дальнейшим развитием и усовершенствованием микроядерных архитектур и одновременно ужесточают требования к минималистичности и простоте кода ядра.

LibOS может обеспечивать произвольный набор абстракций, совместимый с той или иной уже существующей операционной системой, например Linux или Windows.

Гибридное ядро ОС — модифицированные микроядра (минимальная реализация основных функций ядра операционной системы компьютера), позволяющие для ускорения работы запускать «несущественные» части в пространстве ядра.

Все рассмотренные подходы к построению операционных систем имеют свои достоинства и недостатки. В большинстве случаев современные операционные системы используют различные комбинации этих подходов. Так, например, сейчас, ядро Linux представляет собой монолитную систему с отдельными элементами модульного ядра.

Наиболее тесно элементы микроядерной архитектуры и элементы монолитного ядра переплетены в ядре Windows NT. Хотя Windows NT часто называют микроядерной операционной системой, это не совсем так. Микроядро NT слишком велико (более 1 Мбайт, кроме того, в ядре системы находится, например, еще и модуль графического интерфейса), чтобы носить приставку «микро». Компоненты ядра Windows NT располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений, как и положено в микроядерных операционных системах. В то же время все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром. Причина проста: чисто микроядерный дизайн коммерчески невыгоден, поскольку неэффективен.

Таким образом, Windows NT можно с полным правом назвать гибридной операционной системой.

В процессе эволюции сформировались и были реализованы основные идеи, определяющие функциональность ОС: пакетный режим, разделение времени и многозадачность, разделение полномочий, реальный масштаб времени, файловые структуры и файловые системы.

Языки программирования

Большинство алгоритмических языков программирования (Си, Паскаль) созданы на рубеже 1960–1970-х гг. (за исключением Java). За прошедший пери-

од времени периодически появлялись новые языки программирования, однако на практике они не получили широкого и длительного по времени распространения. Другим направлением в эволюции современных языков программирования были попытки создания универсальных языков программирования (Алгол, PL/1, Ада), объединявших в себе достоинства ранее разработанных языков.

Появление ПК и ОС с графическим интерфейсом (Mac OS, Windows) привело к смещению внимания разработчиков программного обеспечения в сферу визуального или объектно-ориентированного программирования, сетевых протоколов, баз данных. Это привело к тому, что в настоящее время в качестве инструментальной среды используется конкретная среда программирования (Delphi, Access и др.) и знания базового языка программирования не требуется. Поэтому можно считать, что круг используемых языков программирования стабилизировался.

Анализ синтаксиса и семантики языков программирования показывает, что их родственные конструкции различаются главным образом «внешним видом» (набором ключевых слов или порядком следования компонентов). Содержимое практически идентично, за исключением небольших различий, не имеющих существенного значения. Таким образом, конструкции современных языков имеют общее содержание (семантику), различный порядок следования компонент (синтаксис) и разные ключевые слова (лексику). Следовательно, различные языки предоставляют пользователю одинаковые возможности при различном внешнем виде программ.

Стандартизацию языков программирования в настоящее время осуществляют комитеты ISO/ANSI, однако их деятельность направлена в основном на неоправданное синтаксическое расширение языков. Для исключения существующих недостатков предложены способы задания семантического и синтаксического стандарта языков программирования.

Семантическое описание любой конструкции языка (оператора, типа данных, процедуры и т. д.) должно содержать не менее трех обязательных частей:

- список компонент (в «Тип указатель» это компоненты «Имя типа» и «Базовый тип»);
- описание каждой компоненты;
- описание конструкции в целом.

Для синтаксического описания обычно используется формальное описание конструкции, например в виде БНФ. Синтаксическое описание присутствует в любом языке, начиная с Алгола.

Со времени создания первых программируемых машин человечество придумало более 8500 языков программирования. Каждый год их число пополняется новыми. Ввиду их большого количества классификция затруднена. В общем плане можно выделить две группы: языки высокого уровня и языки низкого уровня.

Языки высокого уровня — максимально приближены к задаче. Наиболее выражено в предметно-ориентированных языках. Приоритет — что?

Языки низкого уровня — в центре внимания не задача, а технология ее реализации, связанная с языком/машиной. Привлекаются дополнительные понятия, не связанные с задачей. Приоритет — как?

Некоторыми языками умеет пользоваться небольшое количество их собственных разработчиков, другие становятся известны миллионам людей.

Профессиональные программисты иногда применяют в своей работе более десятка разнообразных языков программирования.

Среди большого количества языков программирования самую заметную роль в развитии программирования сыграли три пары: Алгол-60 и Фортран, Паскаль и СИ, Java и Си++. Эти языки не случайно объединены в пары, так как в противостоянии заложенных в них идей происходило прогрессивное развитие.

В таблице 5.1 приведены основные сведения о наиболее распространенных языках, а в таблице 5.2 — о языках специального назначения (экспериментальных и про-

мышленных) [39]. Виды (парадигмы) языков по областям применения:

A — процедурное программирование;
B — объектно-ориентированное программирование;
C — структурное программирование;
D — модульное (компонентное программирование);
E — логическое (реляционное) программирование;
F — функциональное программирование;
G — параллельное программирование;
H — гибрид (смесь парадигм B + C + D + G);
I — специализированный язык.

Таблица 5.1

Сведения о наиболее распространенных языках

Название	Год создания	Вид	Фирма	Стандарт
Фортран (FORTRAN)	1954	A	IBM	ISO 1539:1997
Лисп (LISP)	1958	F	MIT	—
Алгол-60 (Algol 60)	1960	A	IFIP	—
Кобол (COBOL)	1960	A	COLASYL Commitete	ISO 1989:1985
Симула (Sumula)	1962	B	—	—
Бейсик (BASIC)	1963	A	Darmouth College	ISO 10279:1991
ПЛ/1 (PL/1)	1964	A	IBM	ISO 6160:1979
Алгол-68 (Algol 68)	1968	A	IFIP	—
Паскаль (Pascal)	1970	C	ETH	ISO 7185:1990
Форт (FORTH)	1970	A*	Mohasco Industrie	ISO 15145:1997
Си (C)	1972	C*	AT&T Bell Labs	ISO 9899:1999
Smalltalk	1972	B	Xerox PARC	—
Пролог (Prolog)	1973	E	Univ. of Aix-Marseille	ISO 13211:1995
Ада (Ada)	1980	H*	Cil Honewell	ISO 8652:1995
Си++	1984	H*	AT&T Bell Labs	ISO 14882:1998
Java	1995	H	Sun Labs	—

Примечания. MIT — Massachusetts Institute of Technology; PARC — Palo Alto Research Center; ETH — Swiss Federal Institute of Technology; ISO — International Standard Organisation; * — поддержка системного программирования.

Таблица 5.2

Сведения о языках специального назначения

Наименование	Год создания	Вид	Фирма	Стандарт
АПЛ (APL)	1957	I	Harvard Univ.	ISO 8485:1989
Снобол (Shobol)	1962	I	AT&T Bell Labs	—
Сетл (SETL)	1969	I	IBM	—
Параллельный Паскаль Concurrent Pascal	1974	G	CIT	—
CLU	1974	D	MIT	—
Schema	1975	F	MIT	—
Mesa	1976	D*	Xerox PARC	—
Icon	1977	I	AT&T Bell Labs	—
Модула-2 (Modula-2)	1979	D*	ETH	ISO 10514:1996
Оккам (Occam)	1982	G*	Inmos	—
Cedar	1983	H*	Xerox PARC	—
Common Lisp	1984	F	MIT	—
Objective C	1984	H*	Productivity Products	—
Эйфель (Eiffel)	1986	D*	ISE	—
Оберон (Oberon)	1988	D*	ETH	—
Модула-3 (Modula-3)	1988	H*	DEC SRC	—
Оберон-2 (Oberon-2))	1991	D*	ETH	—
Limbo	1996	D*	Bell Labs (Lucent)	—
Component Pascal	1997	D*	Оберон Microsystems	—
C#	2000	H*		—

Примечания. CIT — California Institute of Technology; MIT — Massachusetts Institute of Technology; PARC — Palo Alto Research Center; ETH — Swiss Federal Institute of Technology; ISE — Interactive Software Engeneering; SRC — Systems Research Center; ISO — International Standard Organisation; * — поддержка системного программирования.

Важно различать язык программирования и его реализацию. Сам язык — это система записи, набор правил, определяющих синтаксис и семантику программы. Реализация языка — это программа, которая преобразует запись высокого уровня в последовательность машинных команд. Существуют два способа реализации языка: ком-

пиляция (рис. 5.1) и интерпретация (рис. 5.2).

Рис. 5.1
Схема компиляции

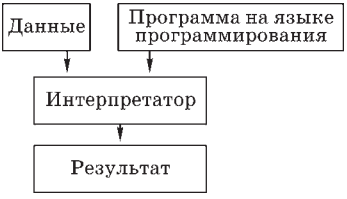


Рис. 5.2
Схема интерпретации

При компиляции специальная рабочая программа (компилятор) осуществляет перевод рабочей программы в эквивалентную на машинном коде и в дальнейшем ее выполнение совместно с данными. В методе интерпретации специальная программа (интерпретатор) устанавливает соответствие между языком и машинными кодами, применяя команды к данным. В принципе любой язык программирования может быть как интерпретируемым, так и компилируемым, но в большинстве случаев есть свой предпочтительный способ реализации. К сожалению, в настоящее время не существует «универсального» компилятора, который мог бы работать с любым существующим языком. Это объясняется отсутствием единой семантической базы. Хотя современные языки программирования похожи друг на друга, идентичность их далеко не полная.

Рис. 5.3
Области пересечения и объединения языков программирования

На рисунке 5.3 представлены области пересечения и объединения языков программирования. Таким образом, существует общая семантическая зона, в которую входят конструкции, принадлежащие всем языкам программирования (или боль-

шинству из них), и область объединения, содержащая конструкции специфические для данного языка. Поэтому создание «универсального» компилятора возможно двумя путями:

1) использование общих конструкций (область пересечения), отбрасывая специфические конструкции языков (область объединения). Это приведет к «обеднению» всех языков программирования;

2) использование всех имеющихся конструкций (область объединения + область пересечения). Такой подход приведет к значительному расширению семантической базы, использованию дополнительных ресурсов.

Многие годы идет спор о том, что такое программирование — наука, искусство или производственный процесс. Надо признать, что право на существование имеют все три версии. Однако в связи с появлением информационных технологий на первый план выходит промышленный характер программирования, который соответствует традиционным стадиям жизненного цикла программного продукта, изложенным в п. 3.7.

Однако наряду с этим направлением развивается так называемое исследовательское программирование. Например, предложенное Э. Раймондом самоорганизующееся, анархичное программирование, получившее название базар. Отличительными чертами его являются отсутствие четкого плана, минимальное управление проектом, большое число сторонних территориально удаленных разработчиков, свободный обмен идеями и кодами.

Программные среды реализуют отдельные задачи и операции информационных технологий. К их числу относятся:

1) текстовые процессоры: Microsoft Word, Лексикон, Lotus Word Perfect, Corel Word Pro, Sun Star Office Writer и др.;

2) электронные таблицы: Microsoft Excel, Corel Quattro Pro, Lotus 1-2-3, Sun Star Office Calc и др.;

3) личные информационные системы: Microsoft Outlook, Lotus Organizer, Lotus Notes, Sun Star Office Schedule и др.;

4) программы презентационной графики: Microsoft Power Point, Lotus Freelance Graphics, Corel Presentations, Sun Star Office Impress и др.;

5) браузеры: Microsoft Internet Explorer, Netscape Navigator, Opera и др.;

6) редакторы веб-страниц: Microsoft Front Page, Netscape Composer, Macromedia Free Hand и др.;

7) почтовые клиенты: Microsoft Outlook, Microsoft Outlook Express, Netscape Messenger, The Bat и др.;

8) редакторы растровой графики: Adobe Photoshop, Corel Photo-Paint и др.;

9) редакторы векторной графики: Corel Draw, Adobe Illustrator и др.;

10) настольные издательские системы: Adobe Page Maker, Quark Xpress, Corel Ventura, Microsoft Publisher и др.;

11) средства разработки: Borland Delphi, Microsoft Visual Basic, Borland C++ Builder, Microsoft Visual C++ и др.

5.2. ТЕХНИЧЕСКИЕ СРЕДСТВА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Основу технического обеспечения информационных технологий составляют компьютеры, являющиеся ядром любой информационной системы. Первоначально компьютеры были созданы для реализации большого объема вычислений, представляющих длинные цепочки итераций. Главным требованием при этом были высокая точность и минимальное время вычислений. Такие процессы характерны для числовой обработки.

По мере внедрения ЭВМ, их эволюционного развития, в частности создания персональных компьютеров, широко стали возникать другие области применения, отличные от вычислений, например обработка экономической информации, создание информационно-справочных систем, автоматизация учрежденческой деятельности и т. п. В данном случае не требовались высокая точность и большой объем вычислений, однако объем обрабаты-

ваемой информации мог достигать миллионов и миллиардов записей. При этом требовалось не только обработать информацию, а предварительно ее найти и организовать соответствующую процедуру вывода. Указанные процессы характерны для нечисловой обработки, требующей в большинстве случаев больших затрат машинного времени. Рассмотренные аспекты оказали решающее влияние на развитие архитектуры ЭВМ.

ЭВМ классической (фон-неймановской) архитектуры состоит из пяти основных функциональных блоков (рис. 5.4):

- запоминающего устройства (ЗУ);
- устройства управления;
- устройств управления и арифметически-логического, рассматриваемых вместе и называемых центральным процессором;
- устройства ввода;
- устройства вывода.

В фон-неймановской архитектуре для обработки огромного объема информации (миллиарды байтов) используется один процессор. Связь с данными осуществляется через канал обмена. Ограничения пропускной способ-

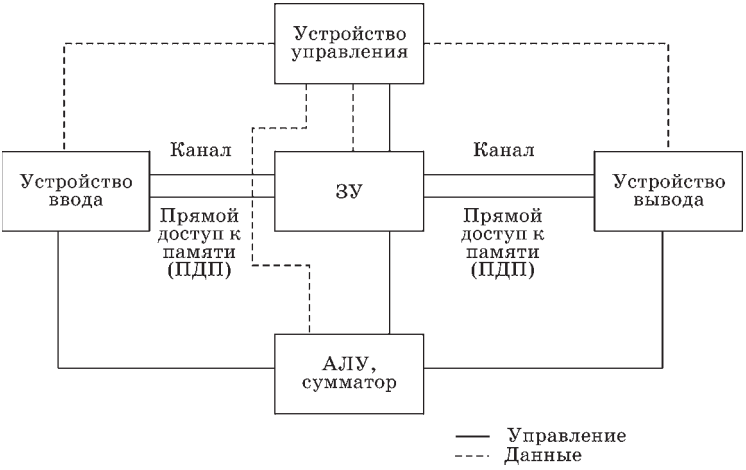


Рис. 5.4
Фон-неймановская архитектура ЭВМ

ности канала и возможностей обработки в центральном процессоре приводят к тупиковой ситуации при нечисловой обработке в случае увеличения объемов информации. Для выхода из тупика были предложены два основных изменения в архитектуру ЭВМ:

- использование параллельных процессоров и организация параллельной обработки;
- распределенная логика, приближающая процессор к данным и устраняющая их постоянную передачу.

Другой недостаток фон-неймановской архитектуры связан с организацией процесса обращения к ЗУ, осуществляемым путем указания адреса для выборки требуемого объекта из памяти. Это приемлемо для числовой обработки, но при нечисловой обработке обращение должно осуществляться по содержанию (ассоциативная адресация). Поскольку для нечисловой обработки в основном используется та же архитектура, необходимо было найти способ организации ассоциативного доступа. Он осуществляется путем создания специальных таблиц (справочников) для перевода ассоциативного запроса в соответствующий адрес. При такой организации обращения к ЗУ, называемом эмуляцией ассоциативной адресации, в случае работы с большими объемами информации резко падает производительность ЭВМ. Это связано с тем, что нечисловая обработка — это не только просмотр, но и обновление данных.

Для преодоления ограничений организации памяти были предложены ассоциативные запоминающие устройства.

Таким образом, ЭВМ для нечисловой обработки должна удовлетворять следующим требования: ассоциативность, параллелизм, обработка в памяти. Кроме этого, на более высоком уровне к архитектуре предъявляются следующие требования:

- перестраиваемость параллельных процессоров и запоминающих устройств;
- сложные топологии соединений между процессорами;
- мультипроцессорная организация, направленная на распределение функций.

Перечисленные выше ограничения и требования были реализованы в машинах баз данных (МБД).

Подытоживая вышесказанное, приведем классификацию архитектур ЭВМ:

- архитектура с одиночным потоком команд и одиночным потоком данных (SISD);
- архитектура с одиночным потоком команд и множественным потоком данных (SIMD);
- архитектура с множественным потоком команд и одиночным потоком данных (MISD);
- архитектура с множественным потоком команд и множественным потоком данных (MIMD).

К разряду SISD относятся современные фоннеймановские однопроцессорные системы. В этой архитектуре центральный процессор работает с парами «атрибут — значение». Атрибут (метка) используется для локализации соответствующего значения в памяти, а одиночная команда, обрабатывающая содержимое накопителя (регистра) и значение, выдает результат. В каждой итерации из входного потока данных используется только одно значение.

К классу SIMD относят большой класс архитектур, основная структура которых состоит из одного контроллера, управляющего комплексом одинаковых процессоров. В зависимости от возможностей контроллера и процессорных элементов, числа процессоров, организации поиска и характеристик маршрутных и выравнивающих сетей выделяют четыре типа SIMD

Матричные процессоры организованы так, что при выполнении заданных вычислений, инициированных контроллером, они работают параллельно. Предназначены для решения векторных и матричных задач, относящихся к числовой обработке.

Ассоциативные процессоры обеспечивают работу в режиме поиска по всему массиву за счет соединения каждого процессора непосредственно с его памятью, используются для решения нечисловых задач.

Процессорные ансамбли представляют совокупность процессоров, объединенных определенным образом для

решения заданного класса задач, ориентированных на числовую и нечисловую обработку.

Конвейерные процессоры (последовательные и векторные) осуществляют выполнение команд и обработку потоков данных по принципу, аналогичному транспортному конвейеру. В этом случае каждый запрос использует одни и те же ресурсы. Как только некоторый ресурс освобождается, он может быть использован следующим запросом, не ожидая окончания выполнения предыдущего. Если процессоры выполняют аналогичные, но не тождественные задания, то это последовательный конвейер, если все задания одинаковы — векторный конвейер.

К классу MISD может быть отнесена единственная архитектура — конвейер, но при условии, что каждый этап выполнения запроса является отдельной командой.

К разряду MIMD, хотя и не всегда однозначно, относят следующие конфигурации:

- мультипроцессорные системы;
- системы с мультиобработкой;
- вычислительные системы из многих машин;
- вычислительные сети.

Общим для данного класса является наличие ряда процессоров и мультиобработки. В отличие от параллельных матричных систем, число процессоров невелико, а термин «мультиобработка» понимается в широком смысле для обозначения функционально распределенной обработки (сортировки, слияния, ввода-вывода и др.)

Другим направлением развития вычислительной техники является нейрокомпьютеринг, основанный на нейронных сетях. Разработки проводятся в двух направлениях: аппаратном и программном. Нейрокомпьютеры обладают сверхвысокой производительностью, но благодаря сложным технологиям имеют очень высокую стоимость. Поэтому они используются узким кругом пользователей для решения суперзадач.

В последние годы ведутся работы по созданию биокомпьютера на основе молекулярных технологий. Идея молекулярного вычислителя состоит в представлении «машинного» слова в виде состояний молекул.

В настоящее время существуют следующие классы технических средств информационных технологий:

- персональные компьютеры;
- мобильные (носимые) ПК;
- нестандартные конструкции ПК;
- мейнфреймы;
- нейрокомпьютеры;
- системы для облачных вычислений;
- суперкомпьютеры;
- вычислительный кластер.

Персональные компьютеры

Подавляющее большинство людей используют именно ПК в качестве средства обработки и хранения информации. К ПК на данное время можно отнести стационарные, планшетные, карманные компьютеры, ноутбуки.

Несмотря на то что ЭВМ изначально были расчитаны на различные вычисления, сейчас большая часть людей используют их для доступа в информационные сети или игр. Вообще, сам термин подразумевает собой любую ЭВМ, используемую в личных целях, будь то суперкомпьютер или калькулятор.

Стационарные ПК. Первые персональные компьютеры (как и любые первые компьютеры вообще) не предназначались для переноски. То есть первые ПК были стационарными. Они состояли из отдельных конструктивно завершенных частей, например, системного блока, монитора и клавиатуры, соединенных интерфейсными кабелями с системным блоком. Это пример раздельной схемы построения ПК. Но в настоящее время также широкое распространение получили ПК-моноблоки, в которых системный блок, монитор и нередко другие устройства (клавиатура, звуковая подсистема, веб-камера, микрофон) конструктивно объединены в одно устройство.

Раздельная схема — в противоположность моноблочной — предполагает, что ПК состоит из системного блока и разнообразных внешних, т. е. конструктивно самостоятельных, подключаемых к системному блоку извне через стандартные интерфейсы (например, USB, D-Sub, DVI,

FireWire), устройств (в частности, мониторы, клавиатура, мышь, микрофоны, звуковые колонки, веб-камеры, принтеры, сканеры, различные внешние модемы, игровые устройства).

Исторически такая схема ПК была самой первой. Она же до сих пор остается самой распространенной схемой стационарных ПК. Например, профессиональные рабочие станции практически всегда строятся по такой схеме.

Главное достоинство раздельной схемы — сравнительно легкая масштабируемость. То есть в любой момент можно без особых затруднений заменить любой из компонентов ПК (например, монитор). Но обратная сторона медали — наименьшая транспортабельность и сравнительная громоздкость такого ПК. Естественно, раздельная схема применяется тогда, когда главное требование к ПК — легкость и простота масштабирования.

Функциональным ядром в раздельной схеме стационарного ПК естественно является системный блок.

Известны два вида конструктивной компоновки системного блока:

- desktop — горизонтальная конструктивная компоновка системного блока, с возможностью размещения монитора на таком системном блоке;
- tower — «башенный» — системный блок в вертикальной конструктивной компоновке.

Моноблок. Конструктивная схема стационарного ПК, в которой системный блок, монитор и микрофон, звуковые колонки, веб-камера конструктивно объединены в одно устройство — моноблок. Такой ПК эргономичнее (занимает минимум пространства) и более привлекателен с эстетической точки зрения. Также такой ПК и более транспортабелен, чем стационарный ПК, построенный по раздельной схеме. Обратной стороной этой медали является сравнительно трудная масштабируемость такого ПК и в том числе сравнительно трудная самостоятельная техническая модернизация.

Также сравнительно затруднено и техническое обслуживание. Например, если у моноблока сломается, напри-

мер, микрофон, то заменить его на исправный нередко возможно только в сервис-центре. Естественно, моноблоки широко применяются в тех случаях, когда на первом месте экономичность, а простота масштабирования или самостоятельного технического обслуживания, напротив, не являются решающими.

Мобильные (носимые) ПК

Ноутбуки. Компактные компьютеры, содержащие все необходимые компоненты (в том числе монитор) в одном небольшом корпусе, как правило, складывающемся в виде книжки (отсюда и название данного вида ПК). Приспособлены для работы в дороге, на небольшом свободном пространстве. Для достижения малых размеров в них применяются специальные технологии: специально разработанные специализированные микросхемы (ASIC), ОЗУ и жесткие диски уменьшенных габаритов, компактная клавиатура, не содержащая цифрового поля, внешние блоки питания, минимум интерфейсных гнезд для подключения внешних устройств.

Как правило, содержат развитые средства подключения к проводным и беспроводным сетям, встроенное мультимедийное оборудование (динамики, микрофон и веб-камеру). В последнее время вычислительная мощность и функциональность ноутбуков не сильно уступают стационарным ПК, а иногда и превосходят их.

Планшетные ПК. Планшетный компьютер iPad. Аналогичны ноутбукам, но содержат сенсорный, т. е. чувствительный к нажатию, экран и не содержат механической клавиатуры. Ввод текста и управление осуществляются через экранный интерфейс, часто доработанный специально для удобного управления пальцами. Некоторые модели могут распознавать рукописный текст, написанный на экране. Чаще всего корпус не раскрывается, как у ноутбуков, а экран расположен на внешней стороне верхней поверхности. Бывают и комбинированные модели, у которых корпус может тем или иным образом раскрываться (например, как слайдер), предоставляя доступ к расположенной внутри клавиатуре. По вычисли-

тельной мощи планшетные ПК уступают стационарным и ноутбукам, так как для длительной работы без внешнего источника питания приходится использовать энергосберегающие комплектующие, жертвуя их быстродействием.

Карманные ПК (*PDA*). Карманный компьютер HP rz1710. Управление ими, как правило, происходит с помощью небольшого по размерам и разрешению экрана, чувствительного к нажатию пальца или специальной палочки-указки — стилуса, а клавиатура и мышь отсутствуют. Некоторые модели, впрочем, содержат миниатюрную фиксированную или выдвигающуюся из корпуса клавиатуру. Разрешение экрана стремится приблизиться к мониторам обычных компьютеров, в среднем около 800×480 в современных моделях. В таких устройствах используются сверхэкономичные процессоры и флеш-накопители небольшого объема, поэтому их вычислительная мощь несопоставима с другими ПК (особенно стационарными). Тем не менее они содержат все признаки персонального компьютера: процессор, накопитель, оперативную память, монитор, операционную систему, прикладное ПО и даже игры и ориентированность на индивидуальное использование.

Все более популярными становятся КПК, содержащие также функции мобильного телефона (коммуникаторы). Встроенный коммуникационный модуль позволяет не только совершать звонки, но и подключаться к Интернету в любой точке, где есть сотовая связь совместимого стандарта (GSM/GPRS/3G, CDMA).

Нестандартные конструкции ПК

Barebone. Компьютер Shuttle XPC SB75G2. Barebone — компьютеры, строящиеся пользователем для выполнения определенных задач (обычно в качестве мультимедийной станции). В продажу поступают в виде так называемых скелетных баз в составе корпуса, материнской платы и системы охлаждения. Материнская плата, как правило, оснащена встроенными звуковым и видеоконтроллерами. Выбор конфигурации и соответственно комплектую-

щих в виде дисковых накопителей, памяти и периферии, а также других устройств (ТВ-тюнера, дополнительной видеокарты и т. п.) остаются на усмотрение пользователя. Как правило, баребоны имеют меньшую высоту корпуса и, как следствие, уменьшенный внутренний объем, а также усовершенствованную систему охлаждения, отличающуюся низкой шумностью.

Защищенные ПК. Ряд компаний производит компьютеры, обладающие устойчивостью к агрессивным средам: сильной вибрации, ударам, большой запыленности, влажности, вандализму — условиям, в которых обычные ПК быстро бы вышли из строя. Как правило, устойчивые ПК выпускаются в формате ноутбуков, более тяжелых и больших по размерам, чем обычные. Их стоимость также значительно выше. Одна из сфер применения таких ПК — военное дело (например, эксплуатация в полевых штабах).

Промышленные ПК. Предназначены для решения задач промышленной автоматизации. Отличаются стойкостью к различным внешним воздействиям, увеличенным жизненным циклом изделия, возможностью подключения к промышленным сетям (PROFINET, Profibus).

Тихий ПК. Бесшумный компьютер Zonbu. Для использования в жилых комнатах используются конструкции ПК, производящие минимум шума или работающие совершенно бесшумно. Такие модели можно оставлять включенными постоянно, что дает ряд преимуществ: отсутствует период загрузки, компьютер всегда готов к работе и может постоянно отслеживать новую почту или мгновенные сообщения для пользователя. В целом, постоянно включенный ПК может выполнять ряд особенных задач:

- быть мультимедийной станцией (воспроизводить видео-, аудиозаписи, интернет-радио);
- работать как видеомагнитофон: записывать передачи телевидения или радио для последующего просмотра или прослушивания в удобное время;
- служить P2P-клиентом (обмениваться файлами в автоматическом режиме с другими компьютерами);

- служить домашним или даже интернет-сервером;
- следить за температурой или присутствием с помощью соответствующих датчиков или фото-, видеокамеры (веб-камеры).

Чтобы сделать ПК тихим, используются специальные бесшумные технологии.

Компактные ПК. Технологии, уменьшающие габариты ПК:

- материнская плата уменьшенного формата (mini-ITX и др.);
- малогабаритный корпус;
- встроенные DVD- и CD-дисководы со щелевой загрузкой или отсутствие встроенного DVD- и CD-дисковода;
- меньшее количество отсеков для жестких дисков и DVD/CD-дисководов, зачастую всего один;
- меньше гнезд USB, аудио- и т. д.;
- внешние блоки питания;
- использование внешних (как, например, DVD- и CD-дисководы) устройств вместо встроенных.

Персональный сервер. Любой сервер, используемый неким человеком в качестве личного сервера и по этому признаку относимый к ПК. Но конструктивно такой сервер, как любой сервер, может быть каким угодно. В частности, такой сервер может быть и стоечным.

Персональная рабочая станция. Конструктивно любой компьютер, используемый в качестве персональной, т. е. однопользовательской, рабочей станции, который зачастую ПК можно признать лишь по этому признаку. То есть конструктивно это может быть даже суперкомпьютер, но он может считаться ПК, если используется в качестве персональной рабочей станции.

Персональный суперкомпьютер. Естественно, это такой же суперкомпьютер, только являющийся личным суперкомпьютером некоего человека. И хотя случаев владения персональными, т. е. личными, суперкомпьютерами еще не было, но в принципе возможно и такое. Ведь многие люди владеют, например, личными самолетами.

Мейнфреймы

Мейнфрейм (от *англ.* mainframe) — данный термин имеет три основных значения [40]:

- большая универсальная ЭВМ — высокопроизводительный компьютер со значительным объемом оперативной и внешней памяти, предназначенный для организации централизованных хранилищ данных большой емкости и выполнения интенсивных вычислительных работ;
- компьютер с архитектурой IBM System/360, 370, 390, zSeries;
- наиболее мощный компьютер, используемый в качестве главного или центрального компьютера (например, в качестве главного сервера).

Рассмотрим особенности и характеристики современных мейнфреймов.

Среднее время наработки на отказ. Время наработки на отказ современных мейнфреймов оценивается в 12–15 лет. Надежность мейнфреймов — это результат их почти 60-летнего совершенствования. Группа разработки операционной системы VM/ESA затратила 20 лет на удаление ошибок, и в результате была создана система, которую можно использовать в самых ответственных случаях.

Повышенная устойчивость систем. Мейнфреймы могут изолировать и исправлять большинство аппаратных и программных ошибок за счет использования следующих принципов.

Дублирование: два резервных процессора, резервные модули памяти, альтернативные пути доступа к периферийным устройствам.

Горячая замена всех элементов вплоть до каналов, плат памяти и центральных процессоров.

Целостность данных. В мейнфреймах используется память с коррекцией ошибок. Ошибки не приводят к разрушению данных в памяти или данных, ожидающих вывода на внешние устройства. Дисковые подсистемы, построенные на основе RAID-массивов с горячей заменой и встроенных средств резервного копирования, защищают от потерь данных.

Рабочая нагрузка. Рабочая нагрузка мейнфреймов может составлять 80–95% от их пиковой производительности. Операционная система мейнфрейма будет обрабатывать все сразу, причем все приложения будут тесно сотрудничать и использовать общие компоненты ПО.

Пропускная способность. Подсистемы ввода-вывода мейнфреймов разработаны так, чтобы работать в среде с высочайшей рабочей нагрузкой на ввод-вывод данных.

Масштабирование. Масштабирование мейнфреймов может быть как вертикальным, так и горизонтальным. Вертикальное масштабирование обеспечивается линейкой процессоров с производительностью от 5 до 200 MIPS и наращиванием до 12 центральных процессоров в одном компьютере. Горизонтальное масштабирование реализуется объединением ЭВМ в Sysplex (System Complex) — многомашинный кластер, выглядящий с точки зрения пользователя единым компьютером. Всего в Sysplex можно объединить до 32 машин. Географически распределенный Sysplex называют GDPS. В случае использования операционной системы VM для совместной работы можно объединить любое количество компьютеров. Программное масштабирование — на одном мейнфрейме может быть сконфигурировано фактически бесконечное число различных серверов. Причем все серверы могут быть изолированы друг от друга так, как будто они выполняются на отдельных выделенных компьютерах и в то же время совместно использовать аппаратные и программные ресурсы и данные.

Доступ к данным. Поскольку данные хранятся на одном сервере, прикладные программы не нуждаются в сборе исходной информации из множества источников, не требуется дополнительное дисковое пространство для их временного хранения, не возникают сомнения в их актуальности. Требуется небольшое количество физических серверов и значительно более простое программное обеспечение. Все это в совокупности ведет к повышению скорости и эффективности обработки.

Защита. Встроенные в аппаратуру возможности защиты, такие как криптографические устройства, и Logi-

cal Partition, и средства защиты операционных систем, дополненные программными продуктами RACF или VM:SECURE, обеспечивают надежную защиту.

Пользовательский интерфейс. Пользовательский интерфейс у мейнфреймов всегда оставался наиболее слабым местом. Сейчас же стало возможно для прикладных программ мейнфреймов в кратчайшие сроки и при минимальных затратах обеспечить современный веб-интерфейс.

Сохранение инвестиций — использование данных и существующих прикладных программ не влечет дополнительных расходов по приобретению нового программного обеспечения для другой платформы, переучиванию персонала, переноса данных и т. д.

Основным производителем мейнфреймов была и остается компания IBM. Более 500 крупнейших мировых корпораций имеют вычислительные комплексы, оснащенные мейнфреймами. Везде, где высоки требования к быстродействию, надежности, безопасности, используются именно они. Такие предприятия, как банки, страховые компании, транспортные корпорации, активно используют сейчас технологии электронной коммерции (электронный обмен информацией, электронные транзакции и др.), обслуживание пользователей с использованием современных информационных технологий.

В 2000–2005 гг. была разработана новая архитектура мейнфреймов, так называемая z-Архитектура, с 64-разрядной адресаций памяти: семейство мейнфреймов System z под управлением операционных систем z/OS. Новая операционная система большой вычислительной машины предоставила возможность параллельного использования и других операционных систем, например Linux. В настоящее время более тысячи Linux машин используют специальную операционную систему z/VM. Что следует особо отметить, z-платформа много дешевле, чем предыдущие архитектуры мейнфрейм.

Мейнфреймы вбирали в себя все новые технологии и модели обработки информации по мере их появления, начиная с модели централизованных вычислений и заканчивая веб-моделью. Они предоставляют пользовате-

лям широкий набор языков программирования — от языка COBOL до Java.

Мэйнфреймы могут выполнять два вида работ, которые соответственно представляют собой обслуживание двух абсолютно различных типов рабочих нагрузок:

Пакетная обработка заданий (Batch job), когда компьютер выполняет работу без участия человека. Используется в случае значительных объемов данных на входе.

Обработка заданий в режиме реального времени (online), например транзакционные системы, такие как система приобретения железнодорожных билетов, система оплаты по кредитной карте и т. п.

Разделение ресурсов — это еще одно ключевое различие между моделями. В случае централизованной обработки информации каждый ресурс (дисковое пространство, оперативная память, процессоры, каналы ввода-вывода и т. д.) используется совместно различными приложениями. Это позволяет добиться более эффективного использования ресурсов и снижает время простоя. В распределенных системах с выделенными ресурсами эффективность значительно меньше, а время простоя — больше.

В отличие от серверов архитектуры Intel или машин, работающих под управлением UNIX, для которых характерна загрузка от 5 до 15%, мейнфреймы, как правило, обеспечивают уровень загрузки от 80 до 100%. В мейнфрейме нагрузка может быть распределена между системами, находящимися на расстоянии до 100 км друг от друга, благодаря чему восстановление после катастрофы осуществляется быстро и удобно. А такие операции, как наращивание мощности или замена вышедших из строя узлов, не прерывают процесса работы. Многие мейнфреймы работают без перерывов на протяжении нескольких лет, а их адаптация к меняющимся задачам и приложениям осуществляется на ходу.

Нейрокомпьютеры

Нейрокомпьютер — устройство переработки информации на основе принципов работы естественных нейронных систем. Эти принципы были формализованы, что

позволило говорить о теории искусственных нейронных сетей. Проблематика же нейрокомпьютеров заключается в построении реальных физических устройств, что позволит не просто моделировать искусственные нейронные сети на обычном компьютере, но так изменить принципы работы компьютера, что станет возможным говорить о том, что они работают в соответствии с теорией искусственных нейронных сетей [41].

Его отличительные черты:

- ассоциативный характер обработки информации — в памяти хранится набор эталонов и обработка состоит в том, чтобы отнести поступающий сигнал к тому или иному эталону по заданной мере близости, которая может быть разной и задается архитектурой обрабатывающей нейронной сети, не всегда известной и выражаемой в аналитическом виде;
- способность к мэппингу — способность нейронной сети автоматически построить топологически корректное отображение одного пространства в другое, когда размерности этих пространств разные;
- обучаемость (первые две черты проявляются в результате одной универсальной процедуры обучения на примерах).

Согласно принятой в нейрокомпьютинге модели искусственный нейрон, как и настоящий, имеет несколько входов и один выход. Все входные сигналы, поступающие в нейрон, сначала умножаются на определенные коэффициенты, называемые весами, затем суммируются, сумма преобразуется с помощью несложной функции и передается на выход. Ключевой элемент такой модели — веса. Именно они придают системе гибкость и позволяют настроиться на решение определенной задачи. Сигнал, который умножается на большой вес, дает большой вклад в общую сумму, а сигнал с нулевым весом не учитывается вовсе.

В отличие от цифровых систем, представляющих собой комбинации процессорных и запоминающих блоков, нейропроцессоры содержат память, распределенную 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 коннекционизмом.

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 надежные нейросистемы могут создаваться и из ненадежных элементов, имеющих значительный разброс параметров.

Основные правила выделения функциональных компонентов идеального нейрокомпьютера (по Миркесу):

- относительная функциональная обособленность: каждый компонент имеет четкий набор функций; его взаимодействие с другими компонентами может быть описано в виде небольшого числа запросов;
- возможность взаимозамены различных реализаций любого компонента без изменения других компонентов.

Постепенно складывается рынок нейрокомпьютеров. В настоящее время широко распространены различные высокопараллельные нейроускорители (сопроцессоры) для различных задач. Моделей универсальных нейрокомпьютеров на рынке мало отчасти потому, что большинство из них реализованы для спецприменений. Примерами нейрокомпьютеров являются нейрокомпьютер Synapse (Siemens, Германия), процессор NeuroMatrix. Издается специализированный научно-технический журнал «Нейрокомпьютеры: разработка, применение». Проводятся ежегодные конференции по нейрокомпьютерам. С технической точки зрения сегодняшние нейрокомпьютеры — это вычислительные системы с параллельными потоками одинаковых команд и множественным потоком

данных (MSIMD-архитектура). Это одно из основных направлений развития вычислительных систем с массовым параллелизмом.

Основным требованием к аппаратному нейрокомпьютеру является каскадирование, т. е. возможность наращивания вычислительной мощности без особых технических требований.

Все фирмы, выпускающие специализированные нейронные вычислительные системы, положили в основу их построения принцип модульности, согласно которому вычислительная система строится как набор модулей, устанавливаемых на материнскую плату.

Применяя принцип модульности, можно в одной системе объединять модули различного назначения: вычислительные, периферийные, преобразования сигналов, ввода/вывода и т. д.

В нейрокомпьютинге постепенно созревает новое направление, основанное на соединении биологических нейронов с электронными элементами. По аналогии с Software (программное обеспечение — «мягкий продукт») и Hardware (электронное аппаратное обеспечение — «твердый продукт»), эти разработки получили наименование Wetware (*англ.*) — «влажный продукт».

В настоящее время уже существует технология соединения биологических нейронов со сверхминиатюрными полевыми транзисторами с помощью нановолокон (Nanowire (*англ.*)). В разработках используется современная нанотехнология, в том числе для создания соединений между нейронами и электронными устройствами используются углеродные нанотрубки.

Распространено также и другое определение термина «Wetware» — человеческий компонент в системах «человек — компьютер».

Применение:

1) управление в реальном времени, в том числе:

- самолетами и ракетами;
- технологическими процессами непрерывного производства (в энергетике, металлургии и др.);
- гибридным двигателем автомобиля;

- пневмоцилиндром;
- сварочным аппаратом;
- электропечью;
- турбогенератором;

2) распознавание образов:

- изображений, человеческих лиц, букв и иероглифов, отпечатков пальцев в криминалистике, речи, сигналов радара и сонара, элементарных частиц и происходящих с ними физических процессов (эксперименты на ускорителях или наблюдение за космическими лучами);
- заболеваний по симптомам (в медицине).

Системы для облачных вычислений

Внешне Cloud Computing выглядит как перенос компьютеров и систем хранения из предприятия в отдельную общую группу, или облако. Конечный пользователь выставляет определенные требования к ресурсам (например, ему требуются вычисления и доступ в глобальную сеть с определенной скоростью), а облако собирает из своих внутренних компонентов нужные мощности и предоставляет их [42].

Главные причины переноса в облако — это экономия средств и масштабируемость. Одна из целей, которую преследует Cloud Computing, — предоставить более дешевые ресурсы, чем те, которые есть у вас и которые вы сами обслуживаете. Помимо экономии, мы также получаем небывалую гибкость и масштабируемость, так как поставщик облачных ресурсов имеет возможность легко расширить виртуальную среду потребителя за счет своей виртуальной инфраструктуры, предоставив ему более высокую пропускную способность или более мощные вычислительные ресурсы.

Гипервизор — монитор виртуальных машин (Virtual Machine Manager, VMM).

Начнем с самого нижнего уровня, который отвечает за инфраструктуру (Infrastructure-as-a-Service, или IaaS, инфраструктура как сервис). IaaS представляет собой сервис по аренде инфраструктуры, т. е. вычислительных ре-

сурсов и систем хранения. К этим ресурсам относятся не только виртуальные серверы с гарантированной вычислительной мощностью, но и каналы связи требуемой пропускной способности для доступа к хранилищам данных и Интернету. Короче говоря, на этом уровне предоставляется возможность временного использования компьютеров или дата-центров при требуемом качестве обслуживания, с возможностью исполнения произвольной операционной системы и программ.

Следующий уровень сервиса при движении на диаграмме вверх — уровень платформы (Platform-as-a-Service, или PaaS, платформа как сервис). PaaS похож на уровень IaaS, но включает в себя операционные системы и сопутствующие службы, ориентированные на определенные приложения. Например, PaaS совместно с виртуальными серверами и системами хранения предоставляет определенную операционную систему и набор приложений (обычно в виде образа виртуальной машины, например файла формата .vmdk для VMWare), а также доступ к различным специализированным локальным сервисам (например, базе данных MySQL). Другими словами, PaaS — это IaaS вместе со стеком приложений, выполняющим конкретную задачу.

И наконец, на самом верху схемы располагается наиболее простой предоставляемый уровень — уровень приложений (Software-as-a-Service, или SaaS, программное обеспечение как сервис), который предполагает использование приложения из централизованной (и, возможно, удаленной — из облака) системы для работы на локальном компьютере. SaaS является измеряемой услугой и позволяет как бы арендовать приложение и оплачивать только время работы с ним.

Виртуализация предоставляет уникальные преимущества, хотя и не является обязательной для создания динамически масштабируемых архитектур. Помимо масштабируемости, виртуализация предоставляет возможность переносить виртуальные машины (Virtual Machine, VM) между физическими серверами в целях выравнивания нагрузки. Виртуализация обеспечивается программным

уровнем, который называется гипервизором (другое название — монитор виртуальных машин — Virtual Machine Monitor, VMM). Этот уровень обеспечивает возможность одновременной работы множества операционных систем (и их приложений) на одной физической машине. В гипервизоре имеется объект, называемый виртуальной машиной, который инкапсулирует в себе операционную систему, ее приложения и конфигурацию. Также гипервизор или виртуальная машина могут эмулировать работу различных устройств.

Базовые платформы облачных вычислений предлагают минимум — только виртуальное аппаратное обеспечение и, возможно, операционную систему. Они обычно более гибкие, поскольку у них меньше ограничений. Можно указать некоторые аппаратные требования, например тип процессора с определенной тактовой частотой, с определенным объемом памяти и т. д. Среди существующих базовых платформ можно отметить Amazon Elastic Compute Cloud, IBM Blue Cloud, Joyent Accelerator, Mosso.

Специализированные платформы облачных вычислений предоставляют какую-либо среду для разработки приложений и собственные сервисы в добавление к базовой платформе. Специализированные платформы, как правило, проще и зачастую предлагают ряд уникальных услуг. Можно выделить следующие специализированные платформы: Microsoft Azure, Google App Engine, Aptana Cloud, Heroku, Ning, Salesforce.

Суперкомпьютеры

Современный суперкомпьютер — это мощный компьютер с производительностью несколько миллиардов операций с плавающей точкой в секунду. Суперкомпьютер представляет собой многопроцессорный и/или многомашинный комплекс, работающий на общую память и общее поле внешних устройств [1].

Термин «суперкомпьютер» вошел в общеупотребительный лексикон благодаря распространенности компьютерных систем американца Сеймура Крея — Control Data 6600, Control Data 7600, Cray-1, Cray-2, Cray-3

и Cray-4. На сегодняшний день суперкомпьютеры являются уникальными системами, создаваемыми традиционными лидерами компьютерного рынка, такими как IBM, Hewlett-Packard, NEC и др., которые приобрели множество ранних компаний, вместе с их опытом и технологиями. Компания Cray Inc. по-прежнему занимает достойное место в ряду производителей суперкомпьютерной техники.

Большинство суперкомпьютеров 1970-х гг. оснащались векторными процессорами. К началу и середине 1980-х гг. небольшое число (от 4 до 16) параллельно работающих векторных процессоров практически стало стандартным суперкомпьютерным решением. Типичный векторный компьютер включает в себя скалярный процессор целочисленной арифметики, функциональные блоки сложения и умножения чисел с плавающей точкой, векторный процессор и общую память. Это компьютеры, построенные по технологии «разделяемая память — один поток управления — много потоков данных» («Shared Memory — Single Instruction — Multi Data»).

Конец 1980-х — начало 1990-х гг. охарактеризовались сменой магистрального направления развития суперкомпьютеров от векторно-конвейерной обработки данных к большому и сверхбольшому числу параллельно соединенных скалярных процессоров.

Массивно-параллельные системы стали объединять в себе сотни и даже тысячи отдельных процессорных элементов, причем ими могли служить не только специально разработанные, но и общеизвестные и доступные в свободной продаже процессоры. Большинство массивно-параллельных компьютеров создавалось на основе мощных процессоров с архитектурой RISC (Reduced Instruction Set Computer), наподобие Power PC или PA-RISC. Использование серийных микропроцессоров позволило не только гибко менять мощность установки в зависимости от потребностей и возможностей, но и значительно удешевить производство. Примерами суперкомпьютеров этого класса могут служить Intel Paragon, IBM SP, Cray T3D/T3E и ряд других.

Петафлопсный рубеж (тысяча триллионов операций с плавающей запятой в секунду) компания Cray Inc. преодолела в 2007 г.

Однако уникальные решения с рекордными характеристиками обычно недешевы, поэтому и стоимость подобных систем никак не могла быть сравнима со стоимостью систем, находящихся в массовом производстве и широко используемых в бизнесе. Прогресс в области сетевых технологий сделал свое дело: появились недорогие, но эффективные решения, основанные на коммуникационных технологиях. Это и предопределило появление кластерных вычислительных систем, фактически являющихся одним из направлений развития компьютеров с массовым параллелизмом вычислительного процесса (Massively Parallel Processing — MPP).

Вычислительный кластер

Вычислительный кластер — это совокупность компьютеров, объединенных в рамках некоторой сети для решения крупной вычислительной задачи. В качестве узлов обычно используются доступные однопроцессорные компьютеры, двух- или четырехпроцессорные SMP-серверы (Symmetric Multi Processor). Каждый узел работает под управлением своей копии операционной системы, в качестве которой чаще всего используются стандартные операционные системы: Linux, NT, Solaris и т. п. Рассматривая крайние точки зрения, кластером можно считать как пару персональных компьютеров, связанных локальной 10-мегабитной сетью Ethernet, так и обширную вычислительную систему, создаваемую в рамках крупного проекта. Такой проект объединяет тысячи рабочих станций на базе процессоров Alpha, связанных высокоскоростной сетью Myrinet, которая используется для поддержки параллельных приложений, а также сетями Gigabit Ethernet и Fast Ethernet для управляющих и служебных целей.

Состав и мощность узлов может меняться даже в рамках одного кластера, давая возможность создавать обширные гетерогенные (неоднородные) системы с задаваемой мощностью. Выбор конкретной коммуникационной

среды определяется многими факторами: особенностями класса решаемых задач, доступным финансированием, необходимостью последующего расширения кластера и т. п. Возможно включение в конфигурацию специализированных компьютеров, например файл-сервера, и, как правило, предоставлена возможность удаленного доступа на кластер через Интернет.

На современном рынке представлено не так много поставщиков готовых кластерных решений. Это связано прежде всего с доступностью комплектующих, легкостью построения самих систем, значительной ориентацией на свободно распространяемое программное обеспечение, а также с уникальностью задач, решаемых с помощью кластерных технологий. Среди наиболее известных поставщиков стоит отметить SGI, VALinux и Scali Computer.

Технологии суперкомпьютеров и кластеров первоначально «выросли» в основном из научных потребностей — для решения фундаментальных и прикладных задач физики, механики, астрономии, метеорологии, сопротивления материалов и т. д., где требовались огромные вычислительные мощности. В каких рыночных нишах будет востребована подобная производительность? Прежде всего это проектирование сложных управляемых систем (самолетов, ракет, космических станций), создание синтетических лекарств с заданными свойствами, генная инженерия, предсказание погоды и природных катаклизмов, повышение эффективности и надежности атомных электростанций, прогнозирование макроэкономических эффектов и многое другое.

Компьютеры следующего поколения

Размеры вычислительных устройств постоянно уменьшаются. Когда-то предполагалось, что более мощные машины будут требовать больше места для периферийных устройств, памяти и т. д. Это предположение оказалось неверным. В 1965 г. Гордон Мур сформулировал действующее и сейчас правило (названное законом Г. Мура), согласно которому производительность вычислительных систем удваивается каждые восемнадцать месяцев (Moore, G. E.

Cramming more components onto integrated circuits // Electronics. 1965. Vol. 38, № 8). Мур вывел свой эмпирический закон, построив зависимость числа транзисторов в интегральной микросхеме от времени. Как следствие из этого закона можно вывести темпы миниатюризации отдельного транзистора.

Ежегодное уменьшение на 10–30% размеров элементарных вычислительных модулей приведет в ближайшие 5–10 лет к практическому применению устройств с элементарными модулями размером примерно в 100–200 ангстрем (0,01–0,02 мк). Другими словами, быстрое развитие цифровых электронных технологий приводит к тому, что размер элементарного вычислительного устройства приближается к размеру молекулы или даже атома.

На таком уровне законы классической физики перестают работать и начинают действовать квантовые законы, которые для многих важных динамических задач еще не описаны теоретически. Для описания работы таких устройств неприменимы классические объекты и методы информатики. В частности, в силу квантового принципа неопределенности Гейзенберга, в таких микроскопических системах нет аналога понятию «bit».

Вместо двоичных цифр новые устройства будут оперировать с «волновыми функциями» («квантовыми битами»). В некотором смысле, информатика в своем развитии в недалеком будущем должна будет перейти от «арифметики» к «функциональному анализу». С одной стороны, это обусловливает переосмысление и замену основных классических (неквантовых) алгоритмов, а с другой — дает возможность вплотную подступиться к решению проблем искусственного интеллекта.

В научно-исследовательских лабораториях крупнейших университетов и транснациональных ИТ-компаний рассматриваются несколько возможных основных направлений создания элементной базы нового поколения вычислительных устройств [43]:

- на принципах ядерного магнитного или электронного парамагнитного резонанса;

- на атомных ионах, помещенных в ловушки Паули или Пеннинга;
- с использованием явления сверхпроводимости;
- на квантовых точках в полупроводниковых неорганических системах;
- на основе оптической симуляции квантовой логики или на металлобиологической гибридной основе.

Многие из указанных направлений имеют существенные недостатки, которые в некоторых случаях приводят к принципиальной невозможности создания конкурентоспособного вычислительного устройства. Характерным примером является проект корпорации IBM, которая в 1999 г. только на первый этап разработки молекулярной элементной базы нового поколения выделила 17 млрд долларов на 5 лет. В результате был создан макет, оперирующий с 5 или 7 квантовыми битами и весом около 7 тонн, способный решать только примитивные задачи типа разложения числа 15 на два множителя 5 и 3.

В настоящее время наиболее перспективным направлением разработки элементной базы компьютеров нового поколения представляется использование самоорганизующихся квантовых точек в твердотельных системах, которые могут выполнять функции квантовых битов и быть связанными в квантовый регистр на основе, например, электростатического или магнитного типа взаимодействия.

5.3. МЕТОДИЧЕСКИЕ СРЕДСТВА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Для большинства технологий характерной чертой их развития является стандартизация и унификация.

Стандартизация — нахождение решений для повторяющихся задач и достижение оптимальной степени упорядоченности.

Унификация — относительное сокращение разнообразия элементов по сравнению с разнообразием систем, в которых они используются.

Если в области традиционного «материального производства» уже давно сложилась система формирования и сопровождения стандартов, то в области информационных технологий многое предстоит сделать.

Главная задача стандартизации в рассматриваемой области — создание системы нормативно-справочной документации, определяющей требования к разработке, внедрению и использованию всех компонентов информационных технологий. На сегодняшний день в области информационных технологий наблюдается неоднородная картина уровня стандартизации. Для ряда технологических процессов характерен высокий уровень стандартизации, например для транспортирования информации, для других уровень стандартизации находится в зачаточном состоянии.

Многообразные стандарты и подобные им методические материалы упорядочим по следующим признакам:

1) по утверждающему органу:

- официальные международные стандарты;
- официальные национальные стандарты;
- национальные ведомственные стандарты;
- стандарты международных комитетов и объединений;
- стандарты фирм-разработчиков;
- стандарты «де-факто»;

2) по предметной области стандартизации:

- функциональные стандарты (стандарты на языки программирования, интерфейсы, протоколы, кодирование, шифрование и др.);
- стандарты на фазы развития (жизненного цикла) информационных систем (стандарты на проектирование, материализацию, эксплуатацию, сопровождение и др.).

В зависимости от методического источника в качестве стандартов могут выступать «метод», «модель», «методология», «подход». Следует отметить, что указанные материалы обладают разной степенью обязательности, конкретности, детализации, открытости, гибкости и адаптируемости.

Исходя из важности проблемы большое внимание уделяется ее решению на международном уровне на различных этапах [45].

Организационная структура, поддерживающая процесс стандартизации ИТ, включает три основных группы организаций:

1) международные организации, входящие в структуру ООН:

- ISO (International Organization for Standardization — Международная организация по стандартизации);
- IEC (International Electrotechnical Commision — Международная электротехническая комиссия);
- ITU-T (International Telecommunication Union-Telecommunications — Международный союз по телекоммуникации). До 1993 г. эта организация имела другое название — CCITT (International Telegraph and Telephone Consultative Committee — Международный консультативный комитет по телефонии и телеграфии или, сокращенно, МККТТ). Предметные области, для которых разрабатываются стандарты: X.200, X-400, X-500, X-600;

2) промышленные профессиональные или административные организации:

- IEEE (Institute of Electrical and Electronic Engineers — Институт инженеров по электротехнике и электронике). Предметные области, для которых разрабатываются стандарты: LAN IEEE802, POSIX;
- Internet и IAB (Internet Activities Board — Совет управления деятельностью Интернета). Предметные области, для которых разрабатываются стандарты: TCP/IP;
- Regional WOS (Workshops on Open Systems — Рабочие группы по открытым системам). Предметные области, для которых разрабатываются стандарты: OSE-profiles;

3) промышленные консорциумы:

- ECMA (European Computer Manufactureres Association — Европейская ассоциация производителей вычислительных машин). Предметные области, для ко-

торых разрабатываются стандарты: OSI, безопасность, управление, Office Document Architecture (ODE);
- OMG (Object Management Group — Группа управления объектами). Предметные области, для которых разрабатываются стандарты: RM: Common Object Request Broker Architecture (CORBA);
- X/Open (организована группой поставщиков компьютерной техники). Предметные области, для которых разрабатываются стандарты: X/Open Portability Guide (XPG4) Common Application Environment;
- NMF (Network Management Forum — Форум управления сетями);
- OSF (Open Software Foundation — Фонд открытого программного обеспечения). Имеет следующие предложения: OSF/1 (соответствует стандарту POSIX и XPG4), MOTIF — графический пользовательский интерфейс (Distributed Computer Environment), DCE — технология интеграции ECMA (European Computer Manufactures Assosiation — Европейская ассоциация производителей вычислительных машин).

В 1987 г. ISO и IEC объединили свою деятельность в области стандартизации ИТ, создав единый орган JTC1 (Joint Technical Committee 1 — Объединенный технический коммитет 1), предназначенный для формирования всеобъемлющей системы базовых стандартов в области ИТ и их расширений для конкретных сфер деятельности.

Работа над стандартами ИТ в JTC1 тематически распределена по подкомитетам (Subcommittees — SC).

В дополнение создана специальная группа по функциональным стандартам (Special Group on Functional Standards — SGFS) для обработки предложений по международным стандартизованным профилям (International Standardized Profiles — ISP), представляющим определения профилей ИТ.

Среди отечественных стандартов в области ИТ можно отметить следующие:
- ГОСТ 34.003-1990 «Информационная технология. Комплекс стандартов на автоматизированные

системы. Автоматизированные системы. Термины и определения»;
- ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы (АС)»;
- ГОСТ Р 34.10 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи»;
- РД 50-34.698-90 «Требования к содержанию документов» и др.;
- ГОСТ Р 51954 «Информационная технология. Профиль прикладной среды организации вычислений на суперЭВМ (PSE10-HIP)»;
- ГОСТ РВ 51987 «Информационная технология. Комплекс стандартов на автоматизированные системы. Типовые требования и показатели качества функционирования информационных систем. Общие положения»;
- Р 50.1.041 «Руководство по проектированию профилей среды открытой системы (СОС) организации-пользователя»;
- Р 50.1.027 «Автоматизированный обмен технической информацией. Основные положения и общие требования»;
- Р 50.1.028 «Методология функционального моделирования».

В качестве примера рассмотрим ряд стандартов различного уровня.

Международный стандарт ISO/OSI разработан Международной организацией по стандартизации (International Standards Organization — ISO), предназначен для использования в области сетевого информационного обмена, представляет эталонную семиуровневую модель, известную как модель OSI (Open System Interconnection — связь открытых систем). Первоначально усилия были направлены на разработку структуры (модели) протоколов связи цифровых устройств. Основная идея была связана с разбиением функций протокола на семь различных категорий (уровней), каждый из которых связан с одним более высоким и с одним более низким уровнем (за исключением

самого верхнего самого нижнего). Идея семиуровневого открытого соединения состоит не в попытке создания универсального множества протоколов связи, а в реализации «модели», в рамках которой могут быть использованы уже имеющиеся различные протоколы. В последнее время достигнут значительный прогресс в реализации различных типов протоколов, о чем говорит успешное функционирование многих сетей передачи данных, например Интернет. Более подробно данный стандарт изложен в п. 2.2.

Международный стандарт ISO/IEC 12207:1995-08-01 — базовый стандарт процессов жизненного цикла программного обеспечения, ориентированный на различные его виды, а также типы информационных систем, куда программное обеспечение входит как составная часть. Разработан в 1995 г. объединенным техническим комитетом ISO/IEC JTC1 «Информационные технологии, подкомитет SC7, проектирование программного обеспечения». Включает описание основных, вспомогательных и организационных процессов.

Основные процессы программного обеспечения:

- процесс приобретения, определяющий действия покупателя, приобретающего информационную систему, программный продукт или его сервис;
- процесс поставки, регламентирующий действия поставщика, снабжающего указанными выше компонентами;
- процесс разработки, определяющий действия разработчика принципов построения программного изделия;
- процесс функционирования, определяющий действия оператора, обслуживающего информационную систему в интересах пользователей и включающий, помимо требований инструкции по эксплуатации, консультирование пользователей и организацию обратной связи с ним;
- процесс сопровождения, регламентирующий действия персонала по модификации программного продукта, поддержки его текущего состояния и функциональной работоспособности.

Вспомогательные процессы регламентируют документирование, управление конфигурацией, обеспечение качества, верификацию, аттестацию, совместную оценку, аудит.

Степень обязательности для организации, принявшей решение о применении ISO/IEC 12207, обусловливает ответственность в условиях торговых отношений за указание минимального набора необходимых процессов и задач, требующих согласования с данным стандартом.

Стандарт содержит мало описаний, направленных на проектирование баз данных, что объясняется наличием отдельных стандартов по данной тематике.

ГОСТ 34 в качестве объекта стандартизации рассматривает автоматизированные системы различных видов и все виды их компонентов, в том числе программное обеспечение и базы данных. Стандарт в основном рассматривает проектные документы, что отличает его от стандарта ISO/IEC 12207. В структуре стандарта выделяют стадии и этапы разработки автоматизированных систем (АС).

Рассмотрим краткую характеристику стадий и этапов.

1. Формирование требований к АС:

- обследование объекта и обоснование необходимости создания АС;
- формирование требований пользователя к АС;
- оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания).

2. Разработка концепции АС:

- изучение объекта;
- проведение необходимых научно-исследовательских работ;
- разработка вариантов концепции АС, удовлетворяющей требованиям пользователя;
- оформление отчета о выполненной работе.

3. Техническое задание:

- разработка и утверждение технического задания.

4. Эскизный проект:

- разработка предварительных проектных решений по системе и ее частям;

- разработка документации на АС и ее части.

5. Технический проект:

- разработка проектных решений по системе и ее частям;
- разработка документации на АС и ее части;
- разработка и оформление документации на поставку изделий для комплектования АС и/или технических требований (технических заданий) на их разработку;
- разработка заданий на проектирование в смежных частях проекта объекта автоматизации.

6. Рабочая документация:

- разработка рабочей документации на систему и ее части;
- разработка или адаптация программ.

7. Ввод в действие:

- подготовка объекта автоматизации к вводу АС в действие;
- подготовка персонала;
- комплектация АС поставляемыми изделиями (программными, техническими и информационными средствами);
- строительно-монтажные работы;
- пусконаладочные работы;
- проведение предварительных испытаний;
- проведение опытной эксплуатации;
- проведение приемочных испытаний.

8. Сопровождение АС:

- выполнение работ в соответствии с гарантийными обязательствами;
- послегарантийное обслуживание.

ГОСТ 34 содержит обобщенную понятийную и терминологическую систему, общую схему разработки, общий набор документов. С точки зрения обязательности в настоящее время таковая отсутствует, что сделало ГОСТ 34 методической поддержкой.

Методика Oracle CDM (Custom Development Method) является развитием ранее разработанной версии Oracle CASE-Method, известной по использованию Designer/2000. Ориентирована на разработку приклад-

ных информационных систем под заказ. Структурно построена как иерархическая совокупность этапов, процессов и последовательностей задач, краткая характеристика которых дана ниже.

Этапы:

- стратегия (определение требований);
- анализ (формирование детальных требований);
- проектирование (преобразование требований в спецификации);
- реализация (разработка и тестирование приложений);
- внедрение (установка, отладка и ввод в эксплуатацию);
- эксплуатация (поддержка, сопровождение, расширение).

Процессы:

- RD — определение производственных требований;
- ES — исследование и анализ существующих систем;
- TA — определение технической архитектуры;
- DB — проектирование и построение базы данных;
- MD — проектирование и реализация модулей;
- CV — конвертирование данных;
- DO — документирование;
- TE — тестирование;
- TR — обучение;
- TS — переход к новой системе;
- PS — поддержка и сопровождение.

Процессы состоят из последовательностей задач, причем задачи разных процессов взаимосвязаны ссылками.

Методика не предусматривает включения новых задач, удаления старых задач, изменения последовательности выполнения задач. Методика необязательна, может считаться фирменным стандартом.

В связи с широким использованием в настоящее время объектной технологии большой интерес представляет CORBA (Common Object Request Broker Architecture) — стандарт в виде набора спецификаций для промежуточного программного обеспечения (middleware) объектного типа. Его автором является международный консорци-

ум OMG (Object Management Group), объединяющий более 800 компаний (IBM, Siements, Microsoft, Sun, Oracle и др.). OMG разработала семантический стандарт, включающий четыре основных типа:

- объекты, моделирующие мир (студент, преподаватель, экзамен);
- операции, относящиеся к объекту и характеризующие его свойства (дата рождения студента, пол и др.);
- типы, описывающие конкретные значения операций;
- подтипы, уточняющие типы.

На основе этих понятий OMG определила объектную модель, спецификацию для развития стандарта CORBA, постоянно развиваемую. В настоящее время CORBA состоит из четырех основных частей:

- Object Request Broker (посредник объектных запросов);
- Object Services (объектные сервисы);
- Common Facilities (общие средства);
- Application and Domain Interfaces (прикладные и отраслевые интерфейсы).

Параллельно с CORBA корпорацией Microsoft был разработан стандарт COM/DCOM (Component Object Model/Distributed COM), предназначенный для объединения мелких офисных программ. Основным недостатком данного стандарта была ориентация на Windows и Microsoft. Корпорация Microsoft долгое время не присоединялась к OMG и развивала собственный стандарт. Однако жизнь заставила приступить к мирным переговорам. OMG взаимодействует с другими центрами стандартизации: ISO, Open Group, WWW консорциум, IEEE и многими другими. CORBA стал неотъемлемой частью распределенных объектных компьютерных систем.

Приведенные примеры стандартов дают представление о подходах к решению проблем стандартизации.

Естественно затраты на стандартизацию могут сделать проектные работы по внедрению информационных технологий более дорогостоящими, однако эти затраты с лихвой окупаются в процессе эксплуатации и развития системы, например при замене оборудования или программной среды.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что входит в состав базовых программных средств?
2. Дайте определение операционной системы.
3. Охарактеризуйте направления развития операционных систем.
4. Укажите направления эволюции современных языков программирования.
5. Какие элементы используются для семантического и синтаксического описания любой конструкции языка программирования?
6. В чем различие языка программирования от его реализации?
7. Чем отличается компилятор от интерпретатора?
8. Перечислите стадии жизненного цикла программного продукта.
9. Какие функции реализуют программные среды?
10. Какие блоки входят в состав ЭВМ классической (фоннеймановской) архитектуры?
11. Укажите основные характеристики персональных компьютеров.
12. Укажите основные характеристики мобильных (носимых) ПК.
13. Укажите основные характеристики нестандартных конструкций ПК.
14. Укажите основные характеристики мейнфреймов.
15. Укажите основные характеристики нейрокомпьютеров.
16. Укажите основные характеристики систем для облачных вычислений.
17. Укажите основные характеристики суперкомпьютеров.
18. Укажите основные характеристики вычислительных кластеров.
19. Каковы перспективы развития ЭВМ?
20. В чем назначение унификации и стандартизации?
21. Перечислите основные типы стандартов.
22. Какие основные процессы программного обеспечения охватываются современными стандартами?

ГЛАВА 6

ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

В результате освоения данной темы студент должен

1) знать:

- методологию и стадии проектирования информационных систем;
- процесс создания программных компонент, архитектурные решения проектов и различные технологии реализации информационных систем;
- задачи оценки качества информационных систем, дефектологические свойства, количественные или качественные оценки качества информационных систем, роль и назначение стандартов, характеристики качества программного обеспечения;

2) уметь:

- использовать технологии и средства проектирования информационных систем;
- применять методы оценки качества в процессе проектирования информационных систем;

3) владеть:

- методологией и технологией проектирования информационных систем;
- навыками использования стандартов в процессе проектирования информационных систем.

6.1. МЕТОДОЛОГИЯ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Согласно определению [8] *система* есть сущность, которая в результате взаимодействия ее частей (компонентов) может поддерживать свое существование и функ-

ционировать как единое целое. В этом определении важно отметить, что поведение системы зависит не от природы свойств ее образующих частей, а от того, как эти части соединены между собой.

Из определения системы следует, что системы функционируют как целое, что порождает у них свойства, отличающиеся от свойств составляющих ее частей. Эти свойства известны как *эмерджентные* (*возникающие*). Разделив систему на компоненты, мы никогда не обнаружим ее существенных свойств, они проявляются только в результате действия целостной системы. Эмерджентные свойства являются важными в понимании сущности анализа и синтеза. Разбиение системы на составляющие части для познания принципов ее функционирования называется *анализом*. С помощью анализа мы получаем знания, однако теряем возможность понимания свойств. Дополнением анализа является *синтез* — создание целого из частей. С помощью синтеза мы обретаем понимание [8].

Принято выделять *простые* и *сложные системы*. Сложность может проявляться двумя различными путями: статическим и динамическим. Статическая сложность (детальная) определяется в процессе детализации как количество рассматриваемых элементов. Динамическая сложность зависит от отношений между элементами. Информационные системы относятся к классу динамически сложных. Таким образом, информационная система состоит из множества элементов или подсистем, которые находятся в разных состояниях и могут изменяться в зависимости от изменения других частей.

По виду элементов информационная система относится к системам типа «процесс» (элементами являются информационные процессы).

Отметим другие важные определения.

Структура — совокупность элементов и их связей.

Цель — состояние, к которому стремится система.

Среда — метасистема, в которую рассматриваемая система входит составной частью.

Функционирование системы — работа системы в рамках заданной структуры.

Развитие системы — работа системы в условиях острых противоречий, которые могут вызвать изменение структуры.

Управление — целенаправленный перевод системы из одного состояния в другое желаемое.

Если целью является познание уже существующей системы, то вполне пригодным оказывается дескриптивное определение системы, которая заключается в следующем: *система* — это совокупность объектов, свойства которой определяются отношением между этими объектами. Объекты называют подсистемами или элементами системы. Каждый объект при самостоятельном исследовании может рассматриваться как система. Функции объекта определяются внутренним устройством объекта. Таким образом, дескриптивное определение системы играет познавательную роль для объяснения функций, реализуемых системой. Функции системы проявляются в процессе взаимодействия ее с внешней средой. При этом важно определить границу между внешней средой и создаваемой системой. Это можно осуществить на основе конструктивного определения системы. Для технических систем особое значение имеет конструктивный подход.

Любая техническая система создается под заранее известную цель, которая обычно является субъективной, поскольку она предлагается разработчиком, но эта цель должна исходить из объективных потребностей общества. Таким образом, можно считать, что цель формируется в процессе взаимодействия между явлениями окружающей нас действительности. При этом возникает ситуация, которая заставляет строить новую систему. Ситуация может стать проблемной, если она не разрешается имеющимися средствами. Могут создаваться новые недостающие средства, и в этом смысле ярким примером является информационные технологии.

Проектирование — процесс, который дает начало изменениям в искусственной среде. Такое определение акцентирует внимание на последствиях внедрения. Проектировщик должен предвидеть конечный результат осуществления своего проекта и определить меры, необходи-

мые для достижения этого результата. Проектирование представляется как итерационный цикл (рис. 6.1). Каждая итерация отличается большей детализацией и меньшей общностью.

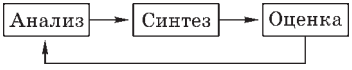


Рис. 6.1
Процесс проектирования

В процессе проектирования выделяют следующие фазы:

Дивергенция — расширение границ проектной ситуации с целью обеспечения более обширного пространства поиска решения.

Трансформация — стадия создания принципов и концепций (исследование структуры проблемы).

Конвергенция — охватывает традиционное проектирование (кодирование, отладка, проработка деталей).

Трудности проектирования:

- предположение о конечном результате проектирования приходится делать еще до того, как исследованы средства его достижения;
- часто случается, что в ходе исследования событий в обратном порядке (от конечного результата) обнаруживаются непредвиденные трудности или открываются новые, более благоприятные возможности.

Под проектированием ИС понимается процесс преобразования входной информации об объекте, методах и опыте проектирования объектов аналогичного назначения в соответствии с ГОСТ в проект ИС. С этой точки зрения проектирование ИС сводится к последовательной формализации проектных решений на различных стадиях жизненного цикла ИС: планирования и анализа требований, технического и рабочего проектирования, внедрения и эксплуатации ИС.

Основными особенностями исходных данных для проектирования ИС являются следующие:

- большое количество действий, подлежащих реализации (многофункциональность);
- значительный объем и сложность ограничений на взаимосвязи проектируемой системы с окружением и трудности их формального описания;

- распределенный и асинхронный режим обработки данных;
- многообразие используемых информационных объектов и их свойств;
- нечеткость требований, их субъективный характер;
- неполнота требований, их расширение в процессе проектирования, необходимость учета развития системы.

Отличительная черта проектирования ИС — коллективное проектирование.

Технология проектирования ИС

Технология проектирования ИС — это совокупность методологии и средств проектирования ИС, а также методов и средств его организации (управление процессом создания и модернизации проекта ИС).

Технологический процесс проектирования ИС в целом делится на совокупность последовательно-параллельных, связанных и соподчиненных цепочек действий, каждое из которых может иметь свой предмет. Таким образом, технология проектирования задается регламентированной последовательностью технологических операций, выполняемых на основе того или иного метода, в результате чего становится ясным не только что должно быть сделано для создания проекта, но и как, кем и в какой последовательности.

Методология проектирования предполагает наличие некоторой концепции, принципов проектирования, реализуемых набором методов, которые, в свою очередь, должны поддерживаться некоторыми средствами.

Назначение методологии создания ИС заключается в организации процесса проектирования и обеспечении управления этим процессом гарантированного выполнения требований как к самой ИС, так и к характеристикам процесса разработки. Основными задачами методологии являются следующие задачи:

- обеспечение создания ИС, отвечающих предъявляемым к ним требованиям;
- создание системы с заданным качеством в заданные сроки и в рамках бюджета;

- поддержка сопровождения, модификации и наращивания ИС;
- создание ИС, отвечающих требованиям открытости, переносимости и масштабируемости.

Методология должна обеспечивать снижение сложности процесса создания ИС за счет полного и точного описания этого процесса и применения современных методов и технологий создания ИС на всем жизненном цикле ИС — от замысла до реализации.

Организация проектирования предполагает определение методов взаимодействия проектировщиков между собой и с заказчиком в процессе создания проекта ИС, которые могут также поддерживаться набором специфических средств.

Классическое проектирование ИС берет свое начало в 1970-х гг. Одно из первых направлений получило название каскадной схемы проектирования (рис. 6.2). Она широко использовалась при проектировании ИС и включала следующие стадии проекта: разработка требований, проектирование, реализация, тестирование, ввод в действие (внедрение). Основной особенностью данной методики является последовательная организация работ при разбиении структуры ИС на заранее определенный ряд подсистем: организационное, методическое, информационное, программное и аппаратное обеспечение. В западной литературе такая схема организации работ получила название водопадной модели (waterfall model) и включала дополнительно итерационные процедуры уточнения требований к системе и рассмотрения

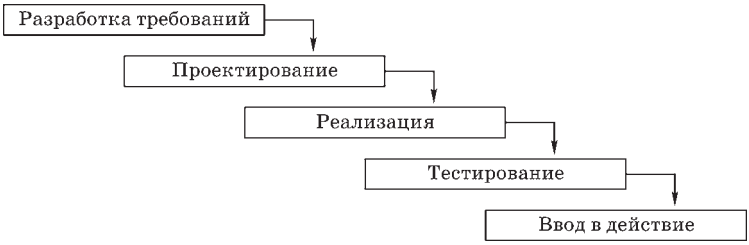


Рис. 6.2
Каскадная модель проектирования ИС

вариантов проектных решений. Каскадная модель предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе.

Основными недостатками каскадной схемы проектирования являются запаздывание получения конечных результатов и низкая эффективность.

Дальнейшим развитием каскадной модели явилась итерационная (поэтапная модель) с промежуточным контролем (рис. 6.3). Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.

В процессе совершенствования разработки ИС появилась схема непрерывной (спиральной) разработки (рис. 6.4), использовавшаяся при реализации крупных проектов фирмы IBM в 1970–1980-х гг.

Характерной особенностью данной методики стал непрерывный спиральный процесс разработки ИС с планируемыми точками передачи в эксплуатацию новых версий и новых функциональных подсистем. На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка. Особое внимание уделяется начальным этапам разработки — анализу и проектированию, где реализуемость

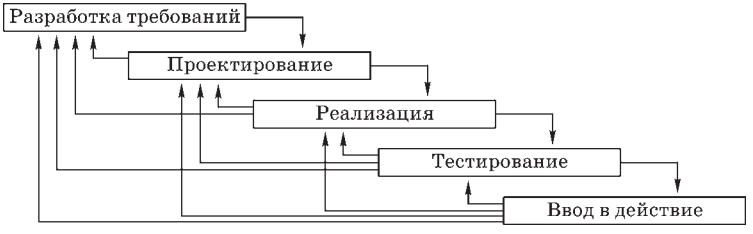


Рис. 6.3
Поэтапная модель проектирования ИС с промежуточным контролем

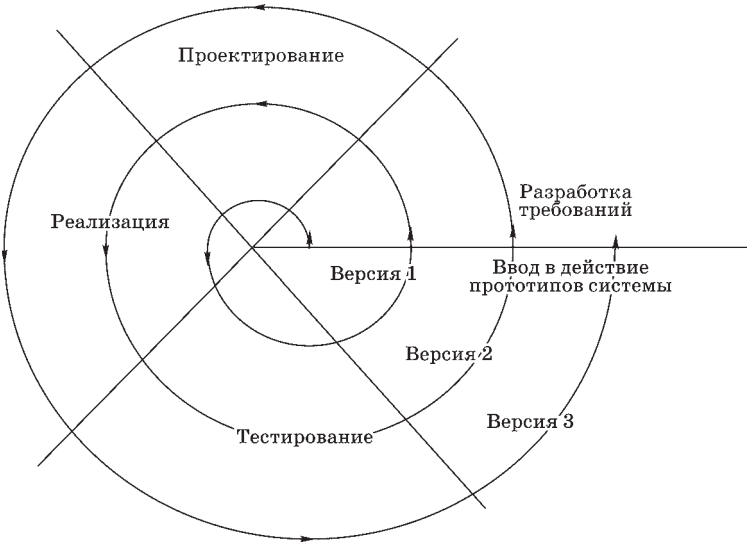


Рис. 6.4
Спиральная модель проектирования ИС

тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования).

Методы проектирования ИС

Их можно классифицировать по степени использования средств автоматизации, типовых проектных решений, адаптивности к предполагаемым изменениям.

Так, по степени автоматизации методы проектирования разделяются на:

- *ручное*, при котором проектирование компонентов ИС осуществляется без использования специальных инструментальных программных средств, а программирование — на алгоритмических языках;
- *компьютерное*, при котором производится генерация или конфигурирование (настройка) проектных решений на основе использования специальных инструментальных программных средств.

По степени использования типовых проектных решений различают следующие методы проектирования:

- оригинальное (индивидуальное), когда проектные решения разрабатываются с нуля в соответствии с требованиями к ИС. Характеризуется тем, что все виды проектных работ ориентированы на создание индивидуальных для каждого объекта проектов, которые в максимальной степени отражают все его особенности;
- типовое, предполагающее конфигурирование ИС из готовых типовых проектных решений (программных модулей). Выполняется на основе опыта, полученного при разработке индивидуальных проектов. Типовые проекты, как обобщение опыта для некоторых групп организационно-экономических систем или видов работ, в каждом конкретном случае связаны со множеством специфических особенностей и различаются по степени охвата функций управления, выполняемым работам и разрабатываемой проектной документации.

По степени адаптивности проектных решений выделяют методы:

- реконструкции, когда адаптация проектных решений выполняется путем переработки соответствующих компонентов (перепрограммирования программных модулей);
- параметризации, когда проектные решения настраиваются (генерируются) в соответствии с изменяемыми параметрами;
- реструктуризации модели, когда изменяется модель проблемной области, на основе которой автоматически заново генерируются проектные решения.

Сочетание различных признаков классификации методов обусловливает характер используемых технологий проектирования ИС, среди которых выделяют два основных класса: *каноническую* и *индустриальную* технологии. Индустриальная технология проектирования, в свою очередь, разбивается на два подкласса: *автоматизированное* (использование CASE-технологий) и *типовое* (параметрически-ориентированное или модельно-ориентированное) проектирование. Использование ин-

дустриальных технологий не исключает использования в отдельных случаях канонических.

В основе *канонического проектирования* лежит каскадная модель жизненного цикла ИС. Процесс каскадного проектирования в жизненном цикле ИС в соответствии с применяемым в нашей стране ГОСТ 34.601-90 «Автоматизированные системы. Стадии создания» делится на следующие семь стадий:

1) исследование и обоснование создания системы;
2) разработка технического задания;
3) создание эскизного проекта;
4) техническое проектирование;
5) рабочее проектирование;
6) ввод в действие;
7) функционирование, сопровождение, модернизация.

В зависимости от сложности объекта автоматизации и набора задач, требующих решения при создании конкретной ИС, стадии и этапы работ могут иметь различную трудоемкость. Допускается объединять последовательные этапы и даже исключать некоторые из них на любой стадии проекта. Допускается также начинать выполнение работ следующей стадии до окончания предыдущей.

Стадии и этапы создания ИС, выполняемые организациями-участниками, прописываются в договорах и технических заданиях на выполнение работ.

Стадия 1. Формирование требований к ИС. На начальной стадии проектирования выделяют следующие этапы работ:

- обследование объекта и обоснование необходимости создания ИС;
- формирование требований пользователей к ИС;
- оформление отчета о выполненной работе и тактико-технического задания на разработку.

Стадия 2. Разработка концепции ИС:

- изучение объекта автоматизации;
- проведение необходимых научно-исследовательских работ;
- разработка вариантов концепции ИС, удовлетворяющих требованиям пользователей;

- оформление отчета и утверждение концепции.

Стадия 3. Техническое задание:

- разработка и утверждение технического задания на создание ИС.

Стадия 4. Эскизный проект:

- разработка предварительных проектных решений по системе и ее частям;
- разработка эскизной документации на ИС и ее части.

Стадия 5. Технический проект:

- разработка проектных решений по системе и ее частям;
- разработка документации на ИС и ее части;
- разработка и оформление документации на поставку комплектующих изделий;
- разработка заданий на проектирование в смежных частях проекта.

Стадия 6. Рабочая документация:

- разработка рабочей документации на ИС и ее части;
- разработка и адаптация программ.

Стадия 7. Ввод в действие:

- подготовка объекта автоматизации;
- подготовка персонала;
- комплектация ИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями);
- строительно-монтажные работы;
- пусконаладочные работы;
- проведение предварительных испытаний;
- проведение опытной эксплуатации;
- проведение приемочных испытаний.

Стадия 8. Сопровождение ИС:

- выполнение работ в соответствии с гарантийными обязательствами;
- послегарантийное обслуживание.

Типовое проектирование ИС предполагает создание системы из готовых типовых элементов. Основополагающим требованием для применения методов типового проектирования является возможность декомпозиции про-

ектируемой ИС на множество составляющих компонентов (подсистем, комплексов задач, программных модулей и т. д.). Для реализации выделенных компонентов выбираются имеющиеся на рынке типовые проектные решения, которые настраиваются на особенности конкретного предприятия.

Типовое проектное решение (*ТПР*) — это тиражируемое (пригодное к многократному использованию) проектное решение. Принятая классификация ТПР основана на уровне декомпозиции системы. Выделяются следующие классы ТПР:

- *элементные ТПР* — типовые решения по задаче или по отдельному виду обеспечения задачи (информационному, программному, техническому, математическому, организационному);
- *подсистемные ТПР* — в качестве элементов типизации выступают отдельные подсистемы, разработанные с учетом функциональной полноты и минимизации внешних информационных связей;
- *объектные ТПР* — типовые отраслевые проекты, которые включают полный набор функциональных и обеспечивающих подсистем ИС.

Для реализации типового проектирования используются два подхода: параметрически-ориентированное и модельно-ориентированное проектирование.

Параметрически-ориентированное проектирование включает следующие этапы: определение критериев оценки пригодности пакетов прикладных программ (ППП) для решения поставленных задач, анализ и оценка доступных ППП по сформулированным критериям, выбор и закупка наиболее подходящего пакета, настройка параметров (доработка) закупленного ППП.

Критерии оценки ППП делятся на следующие группы:

- назначение и возможности пакета;
- отличительные признаки и свойства пакета;
- требования к техническим и программным средствам;
- документация пакета;
- факторы финансового порядка;

- особенности установки пакета;
- особенности эксплуатации пакета;
- помощь поставщика по внедрению и поддержанию пакета;
- оценка качества пакета и опыт его использования;
- перспективы развития пакета.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации.

Технология проектирования в этом случае должна обеспечивать единые средства для работы как с моделью типовой ИС, так и с моделью конкретного предприятия.

Типовая ИС в специальной базе метаинформации — репозитории — содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование ПО. Таким образом, модельно-ориентированное проектирование ИС предполагает прежде всего построение модели объекта автоматизации с использованием специального программного инструментария. Возможно также создание системы на базе типовой модели ИС из репозитория, который поставляется вместе с программным продуктом и расширяется по мере накопления опыта проектирования ИС для различных отраслей и типов производства.

Реализация типового проекта предусматривает выполнение следующих операций:

- установку глобальных параметров системы;
- задание структуры объекта автоматизации;
- определение структуры основных данных;
- задание перечня реализуемых функций и процессов;
- описание интерфейсов;
- описание отчетов;
- настройку авторизации доступа;
- настройку системы архивирования.

В настоящее время методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде жизненного цикла (ЖЦ) ИС, представляя его как некоторую последовательность стадий и выполняемых на них процессов. Для каждого этапа определяются состав и последовательность выпол-

няемых работ, получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т. д. Такое формальное описание ЖЦ ИС позволяет спланировать и организовать процесс коллективной разработки и обеспечить управление этим процессом.

Жизненный цикл ИС можно представить как ряд событий, происходящих с системой в процессе ее создания и использования. Модель жизненного цикла отражает различные состояния системы, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления. Модель жизненного цикла — структура, содержащая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение всей жизни системы, от определения требований до завершения ее использования.

Следствием недостатков классических методов проектирования явился переход к системному проектированию.

Системный подход оперирует с рядом категориальных понятий. Фундаментальным понятием системного подхода является понятие системы, определяя которую необходимо преследовать определенную цель. Если целью является познание уже существующей системы, то вполне пригодным оказывается дескриптивное определение системы, которое заключается в следующем: система — это совокупность объектов, свойства которой определяются отношением между этими объектами. Объекты называют подсистемами или элементами системы. Каждый объект при самостоятельном исследовании может рассматриваться как система. Функции объекта определяются внутренним устройством объекта. Таким образом, дескриптивное определение системы играет познавательную роль для объяснения функций, реализуемых системой. Функции системы проявляются в процессе взаимодействия ее с внешней средой. При этом важно определить границу между внешней средой и создаваемой системой. Это можно осуществить на основе конструктивного определения

системы. Для технических систем особое значение имеет конструктивный подход.

Любая техническая система создается под заранее известную цель. Цель такой системы обычно является субъективной, поскольку она предлагается разработчиком, но эта цель должна исходить из объективных потребностей общества. Таким образом, можно считать, что цель формируется в процессе взаимодействия между явлениями окружающей нас действительности. При этом возникает ситуация, которая заставляет строить новую систему. Ситуация может стать проблемной, если она не разрешается имеющимися средствами. Могут создаваться новые недостающие средства, и в этом смысле ярким примером является информационная технология.

Архитектурное описание самым тесным образом связано с процессом проектирования ИС, причем в ряде определений термина «архитектура» на этот факт указывается в явном виде. Обычно выделяют пять различных подходов в проектировании, которые называют также стилями проектирования и, по существу, определяют группы методологий разработки ПО: стиль, основанный на календарном планировании (Calendar-driven); стиль, основанный на управлении требованиями (Requirements-driven); стиль, в основу которого положен процесс разработки документации (Documentation-driven); стиль, основанный на управлении качеством (Quality-driven); архитектурный стиль (Architecture-driven).

Основой календарного стиля является график работ. Команда разработчиков выполняет работы поэтапно. Проектные решения принимаются из целей и задач конкретного этапа. Слабые места данного стиля — основные решения принимаются исходя из локальных целей, при этом мало внимания уделяется процессу разработки, разработке документации, созданию стабильных архитектур и процессу внесения изменений. Все это приводит к высокой суммарной стоимости владением в долгосрочном плане. Данный стиль считается морально устаревшим, однако многие организации продолжают его использовать.

Стиль, основанный на управлении требованиями или чертами (features), предполагает, что основное внимание уделяется функциональным характеристикам системы, при этом часто недостаточно внимания уделяется нефункциональным характеристикам, таким как масштабируемость, портабельность, поддерживаемость и др., определенным в ISO 9126. Проектные решения принимаются преимущественно исходя из локальных целей, связанных с реализацией тех или иных функций. Данный подход достаточно эффективен в случае, если требования определены и не изменяются в процессе проектирования. Основными недостатками данного подхода являются следующие: недостаточное внимание уделяется требованиям стандарта ISO 9126 (управление качеством); разрабатываемые архитектуры, как правило, не являются стабильными, так как каждая из реализуемых функций отображается на один или несколько функциональных компонентов. Это обстоятельство усложняет добавление к системе новых требований. Данный подход позволяет успешно отслеживать требования в плане реализации необходимой функциональности, однако использование его для долгосрочных проектов является неэффективным.

Стиль, в основу которого положен процесс разработки документации, до настоящего времени продолжает использоваться в правительственных организациях и крупных компаниях. Данный стиль может рассматриваться как вырожденный вариант стиля, основанного на управлении качеством, и ориентирован на разработку документации. В качестве основных недостатков данного подхода выделяются следующие: неоправданно много времени и сил затрачивается на разработку документации в ущерб качеству разрабатываемого кода. При этом создаваемая документация часто не используется ни пользователем, ни заказчиком.

Стиль, основанный на управлении качеством, предполагает самое широкое использование различных мер для отслеживания критичных с точки зрения функционирования параметров. Например, требуется получить время реакции системы на запрос менее одной секунды

или обеспечить среднее время между отказами не менее 10 лет. В этом случае данные параметры отслеживается на всех этапах проектирования систем и часто это делается в ущерб другим характеристикам, таким как масштабируемость, простота сопровождения и т. п. Типовыми проблемами, возникающими при использовании данного стиля, являются следующие: часто оптимизируются не те параметры, которые должны быть в действительности, когда появляются новые требования, бывает очень трудно изменить функциональность системы, созданные таким образом системы обычно не отличаются высоким качеством архитектурных решений. В целом данный стиль считается консервативным. Его использование может быть оправдано в случае, когда необходимо создать системы с экстремальными характеристиками.

Архитектурный стиль представляет собой наиболее зрелый подход к проектированию. В рамках данного стиля во главу угла ставится создание фреймворков (набора типовых решений, в том числе программных), которые могут быть легко адаптированы ко всем потенциальным требованиям всех потенциальных заказчиков. Отличительная особенность данного стиля состоит в том, что задача проектирования разбивается на две отдельные подзадачи: создание многократно используемого фреймворка и создание конкретного приложения (системы) на его основе. При этом эти две задачи могут решать разные специалисты. Основная цель создания данного стиля — это устранение недостатков стиля, основанного на управлении требованиями. Использование архитектурного стиля позволяет реализовать инкрементное и итеративное проектирование, т. е. оперативно изменять существующую и добавлять новую функциональность.

В качестве примера рассмотрим методологию создания информационных систем, основанную на использовании средств быстрой разработки приложений (Rapid Application Development, RAD), и методологию создания корпоративных информационных систем.

Методология RAD

Данная методология охватывает все этапы жизненного цикла современных информационных систем.

Методология RAD — это комплекс специальных инструментальных средств, позволяющих оперировать с определенным набором графических объектов, функционально отображающих отдельные информационные компоненты приложений [18].

Основные принципы методологии RAD можно свести к следующим:

- используется итерационная (спиральная) модель разработки;
- полное завершение работ на каждом из этапов жизненного цикла необязательно;
- в процессе разработки информационной системы обеспечивается тесное взаимодействие с заказчиком и будущими пользователями;
- CASE-средства и средства быстрой разработки приложений;
- применяются средства управления конфигурацией, облегчающие внесение изменений в проект и сопровождение готовой системы;
- используются прототипы, позволяющие полнее выяснить и реализовать потребности конечного пользователя;
- тестирование и развитие проекта осуществляются одновременно с разработкой;
- разработка ведется немногочисленной и хорошо управляемой командой профессионалов;
- обеспечиваются грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

Объектно-ориентированный подход. Средства RAD позволили реализовать совершенно иную по сравнению с традиционной технологию создания приложений: информационные объекты формируются как некие действующие модели (прототипы), чье функционирование согласуется с пользователем, а затем разработчик может переходить непосредственно к формированию законче-

ных приложений, не теряя из виду общей картины проектируемой системы.

Использование объектно-ориентированных принципов позволяет создать описание (модель) предметной области в виде совокупности объектов — сущностей, объединяющих данные и методы обработки этих данных (процедуры). Каждый объект обладает собственным поведением и моделирует некоторый объект реального мира. С этой точки зрения объект является вполне осязаемым и демонстрирует определенное поведение.

Применение принципов объектно-ориентированного программирования позволило создать средства проектирования приложений, называемые средствами *визуального программирования*. Визуальные инструменты RAD позволяют создавать сложные графические интерфейсы пользователя вообще без написания кода программы. При этом разработчик может на любом этапе наблюдать то, что закладывается в основу принимаемых решений. Визуальные средства разработки оперируют в первую очередь со стандартными интерфейсными объектами — окнами, списками, текстами, которые легко можно связать с данными из базы данных и отобразить на экране монитора. Другая группа объектов представляет собой стандартные элементы управления — кнопки, переключатели, флажки, меню и т. п., с помощью которых осуществляется управление отображаемыми данными.

Логика приложения, построенного средствами RAD, является событийно-ориентированной. Это означает, что каждый объект, входящий в состав приложения, может генерировать события и реагировать на события, генерируемые другими объектами. Примерами событий могут быть открытие и закрытие окон, щелчок на кнопке, нажатие клавиши клавиатуры, движение мыши, изменение данных в базе данных и т. п.

Фазы жизненного цикла в рамках методологии RAD. При использовании методологии быстрой разработки приложений жизненный цикл информационной системы состоит из четырех фаз:

- анализа и планирования требований;
- проектирования;
- построения;
- внедрения.

Рассмотрим каждую из них более подробно.

Фаза анализа и планирования требований содержит определение следующих характеристик:

- функции, которые должна выполнять разрабатываемая информационная система;
- наиболее приоритетные функции, требующие разработки в первую очередь;
- информационные потребности;
- масштаб проекта;
- временные рамки для каждой из последующих фаз;
- сама возможность реализации данного проекта в установленных рамках финансирования на имеющихся аппаратных и программных средствах.

Если реализация проекта принципиально возможна, то результатом фазы анализа и планирования требований будет список функций разрабатываемой информационной системы с указанием их приоритетов, а также предварительные функциональные и информационные модели системы.

На *фазе проектирования* необходимым инструментом являются CASE-средства, используемые для быстрого получения работающих прототипов приложений. Прототипы, созданные с помощью CASE-средств, анализируются пользователями, которые уточняют и дополняют те требования к системе, которые не были выявлены на предыдущей фазе. Таким образом, на данной фазе также необходимо участие будущих пользователей в техническом проектировании системы.

Далее на этой фазе проводится анализ и, если требуется, корректировка функциональной модели системы. Детально рассматривается каждый процесс системы. При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалоговое окно или отчет (это позволяет устранить неясности или неоднозначности). Затем определяются требования разграничения доступа к данным.

После детального рассмотрения процессов определяется количество функциональных элементов разрабатываемой системы. Это позволяет разделить информационную систему на ряд подсистем, каждая из которых реализуется одной командой разработчиков за приемлемое для RAD-проектов время (порядка полутора месяцев). С использованием CASE-средств проект распределяется между различными командами — делится функциональная модель.

На этой же фазе происходит определение набора необходимой документации. Результатами данной фазы являются:

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

На *фазе построения* выполняется собственно быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных ранее моделей, а также требований нефункционального характера. Разработка приложения ведется средствами визуального программирования. Формирование программного кода частично выполняется с помощью автоматических генераторов кода, входящих в состав CASE-средств. Код генерируется на основе разработанных моделей.

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

Завершается физическое проектирование системы, а именно:

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

Результатом реализации данной фазы является готовая информационная система, удовлетворяющая всем требованиям пользователей.

Фаза внедрения. Фаза внедрения в основном сводится к обучению пользователей разработанной информационной системы.

Ограничения методологии RAD. Ее применение наиболее эффективно при создании сравнительно небольших систем, разрабатываемых для конкретного заказчика.

При разработке же типовых систем, не являющихся законченным продуктом, а представляющих собой совокупность типовых элементов информационной системы, большое значение имеют такие показатели проекта, как управляемость и качество, которые могут войти в противоречие с простотой и скоростью разработки. Это связано с тем, что типовые системы обычно централизованно сопровождаются и могут адаптироваться к различным программно-аппаратным платформам, системам управления базами данных, коммуникационным средствам, а также интегрироваться с существующими разработками. Поэтому для такого рода проектов необходимы высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки.

Методология RAD не подходит для создания сложных расчетных программ, операционных систем и программ управления сложными инженерно-техническими объектами; для разработки приложений, в которых интерфейс пользователя является вторичным, т. е. отсутствует наглядное определение логики работы системы (приложе-

ния реального времени, драйверы или службы); для разработки систем, от которых зависит безопасность людей, например систем управления транспортом или атомными электростанциями.

Профили открытых информационных систем

Открытые информационные системы создаются в процессе информатизации всех основных сфер современного общества: органов государственного управления, финансово-кредитной сферы, информационного обслуживания предпринимательской деятельности, производственной сферы, науки, образования. Развитие и использование открытых информационных систем неразрывно связаны с применением стандартов на основе методологии функциональной стандартизации информационных технологий.

Профиль — это совокупность нескольких (или подмножество одного) базовых стандартов с четко определенными и гармонизированными подмножествами обязательных и факультативных возможностей, предназначенная для реализации заданной функции или группы функций.

Базовые стандарты и профили в зависимости от проблемно-ориентированной области применения информационных систем могут использоваться как непосредственные директивные, руководящие или рекомендательные документы, а также как нормативная база, необходимая при выборе или разработке средств автоматизации технологических этапов или процессов создания, сопровождения и развития информационных систем.

Обычно рассматривают две группы профилей, регламентирующих:

- архитектуру и структуру информационной системы;
- процессы проектирования, разработки, применения, сопровождения и развития системы.

В международной функциональной стандартизации информационных технологий принято довольно жесткое понятие профиля. Считается, что его основой могут быть только утвержденные международные и национальные

стандарты. Использование стандартов де-факто и нормативных фирменных документов не допускается.

Другой подход к разработке и применению профилей информационных систем состоит в использовании совокупности адаптированных и параметризованных базовых международных и национальных стандартов и открытых спецификаций, отвечающих стандартам де-факто и рекомендациям международных консорциумов.

Эталонная модель среды открытых систем определяет разделение любой информационной системы на две составляющие: *приложения* (прикладные программы и программные комплексы) и *среду*, в которой эти приложения функционируют.

Профили информационной системы с иерархической структурой могут включать в себя:

- стандартизованные описания функций, выполняемых данной системой;
- функции взаимодействия системы с внешней для нее средой;
- стандартизованные интерфейсы между приложениями и средой информационной системы;
- профили отдельных функциональных компонентов, входящих в систему.

Структура профилей информационных систем. Профили характеризуют каждую конкретную информационную систему на всех стадиях ее жизненного цикла, задавая согласованный набор базовых стандартов, которым должны соответствовать система и ее компоненты.

Стандарты, важные с точки зрения заказчика, должны задаваться в техническом задании на проектирование системы и составлять ее первичный профиль. То, что не задано в техническом задании, первоначально остается на усмотрение разработчика системы. Профиль конкретной системы не является статичным.

В профиль конкретной системы включаются спецификации компонентов, разработанных в составе данного проекта, и спецификации использованных готовых программных и аппаратных средств, если эти средства не специфицированы соответствующими стандартами. По-

сле завершения проектирования и испытаний системы, в ходе которых проверяется ее соответствие профилю, профиль применяется как основной инструмент сопровождения системы при эксплуатации, модернизации и развитии.

На стадиях жизненного цикла информационной системы выбираются и затем применяются следующие основные функциональные профили:

- прикладного программного обеспечения;
- среды информационной системы;
- защиты информации в информационной системе;
- инструментальных средств, встроенных в информационную систему.

Методология создания корпоративных ИС

Рассматриваемая методология создания корпоративных ИС состоит из двух основных взаимосвязанных частей: методологии анализа ИС, включающей описание деятельности организации и формирование требований к ИС на основе бизнес-процессов, и методологии проектирования от данных, предназначенной для проектирования и быстрой разработки программного и информационного обеспечения ИС.

Методология является развитием одного из классических направлений с включением новых разработок ИТ и содержит:

- итерационную спиральную модель жизненного цикла ИС;
- комплекс развивающихся систем согласованных моделей;
- технологию анализа ИС на основе бизнес-процессов;
- технологию проектирования от данных;
- комплекс согласованных инструментальных средств.

Итерационная спиральная модель жизненного цикла ИС. Процесс создания и сопровождения информационных систем в виде жизненного цикла (ЖЦ) ИС представляется в виде последовательности стадий, каждая из которых разбита на этапы, и выполняемых на них процессов. Для каждого этапа определяются последовательность выполняе-

мых работ, получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т. д. Такое формальное описание ЖЦ ИС позволяет спланировать и организовать процесс коллективной разработки и обеспечить управление этим процессом. Жизненный цикл ИС, определяемый методологией, приведен в таблице 6.1. Он включает стадии анализа, проектирования, разработки, тестирования и интеграции, внедрения, сопровождения и развития ИС. В таблице приведены также перечень основных этапов для каждой стадии ЖЦ и процессы, выполняемые на протяжении всего ЖЦ — процессы управления и интегральные процессы. Эти процессы в той или иной степени присутствуют на каждом из этапов.

Процесс создания ИС представляет собой процесс построения и последовательного преобразования согласованных моделей на всех этапах ЖЦ. Эти модели сохраняются и накапливаются в репозитории проекта. С помощью CASE-средств модели создаются, преобразуются и контролируются. Основными результатами на каждом этапе ЖЦ являются модели определяемых на данном этапе объектов (организации, требований к ИС, проекта ИС, требований к приложениям и т. д.). Характер выполняемых процессов и организация работ в представленной модели ЖЦ основаны на подходе информационного инжиниринга и отличаются от классической каскадной модели ЖЦ, несмотря на внешнюю схожесть. При традиционной

Жизненный

Процессы организации и управления			
Анализ	**Проектирование**	**Разработка**	
Обследование и создание моделей деятельности организации. Анализ (моделей) существующих ИС. Анализ моделей и формирование требований к ИС. Разработка плана создания ИС	Концептуальное проектирование. Разработка архитектуры ИС. Проектирование общей модели данных. Формирование требований к приложениям	Разработка, прототипирование и тестирование приложений. Разработка интеграционных тестов. Разработка пользовательской документации	
Интегральные процессы: управление конфигурацией,			

обработке данных разработка велась строго последовательно. Требования ТЗ утверждались в начале разработки, а их выполнение проверялось в конце. Переход от стадии к стадии, от этапа к этапу допускался только после полного выполнения всего перечня работ и получения всех запланированных результатов. ЖЦ ИС, предлагаемый в новой методологии, определяется следующими особенностями:

- современные средства CASE, 4GL, СУБД и другие предоставляют возможности быстрого проектирования, прототипирования, разработки и тестирования приложений и баз данных на основе построенных моделей;
- методология предполагает активное участие заказчиков на всех этапах создания ИС, поскольку модели, создаваемые на каждом этапе, понятны и разработчику и заказчику.

Предусмотрены возможности оперативного и быстрого пересмотра требований и разработанных решений на основе современных средств, возможности неравномерной, параллельной разработки различных частей проекта, возможности возврата на предыдущие этапы по отдельным частям проекта при необходимости внесения изменений, версионный характер изменения проекта или его частей при поддержке CASE-средств. Это определяет итерационный, спиральный характер предлагаемой модели жизненного цикла.

Таблица 6.1

цикл ИС

проектом: планирование, управление			
	Интеграция и тестирование	**Внедрение**	**Сопровождение**
	Интеграция и тестирование приложений в составе системы. Оптимизация приложений и баз данных. Подготовка эксплуатационной документации. Тестирование системы	Обучение пользователей. Развертывание системы на месте эксплуатации. Инсталляция баз данных. Эксплуатация. Проведение ПСИ	Регистрация, диагностика и локализация ошибок. Внесение изменений и тестирование. Управление режимами работы ИС
документирование, проверка, интеграция			

Комплекс развивающихся систем согласованных моделей. Методология предусматривает создание корпоративных информационных систем как процесс построения и последовательного развития систем согласованных моделей, начиная от системы моделей, описывающих деятельность организации, и заканчивая готовой информационной системой. Модели должны создаваться, преобразовываться и контролироваться с помощью соответствующих CASE-средств и сохраняться в репозитории. Исходными являются модели бизнес-процессов, на основе которых строится компьютерная модель организации, описанная в терминах бизнес-процессов и бизнес-функций. Из этой модели может быть получено большинство важнейших требований к информационной системе. Создается система моделей описания требований к ИС, которая затем преобразуется в систему моделей, описывающих проект ИС. Формируются модели архитектуры ИС, требований к программному обеспечению (ПО) и информационному обеспечению (ИО). Затем формируется архитектура ПО и ИО, выделяются корпоративные БД и отдельные приложения, формируются модели требований к приложениям и проводится их разработка, тестирование и интеграция.

Технология анализа ИС на основе бизнес-процессов. Основу деятельности любой организации составляют ее бизнес-процессы, которые определяются целями и задачами организации. Процессы обеспечивают реализацию всех видов деятельности организации, связанных с производством товаров и/или услуг, которые корпорация либо производит, либо продает и поставляет, либо делает все это в совокупности. Каждый бизнес-процесс характеризуется четко определенными во времени началом и концом, внешними интерфейсами, которые либо связывают его с другими бизнес-процессами внутри организации, либо описывают выход во внешнее окружение последовательностью выполняемых работ и правилами их выполнения (бизнес-правилами). Для каждой работы, входящей в бизнес-процесс, определены временные характеристики, определяющие ее место в общей последовательности работ, условия инициации и время выполнения. В отли-

чие от описания организации на основе иерархической функциональной структуры, которую невозможно объективно оценить, описание на основе процессов позволяет точно представить цели, характеристики (в том числе динамические) и конечный результат каждого вида деятельности организации. Исходя из того, что основные бизнес-процессы реализуют по своей природе цели и задачи организации, методология предлагает строить описание деятельности организации, как процесс создания и развития систем согласованных моделей, основанных на моделях бизнес-процессов. В процессе детализации моделей и их последующей интеграции должно обеспечиваться сохранение всех функциональных свойств, отражающих цели и задачи организации, и согласованности моделей. Такая согласованность обеспечивается методологией и поддерживающими ее современными CASE-средствами.

В процессе описания организации и ее деятельности формируются три основных системы моделей организации: *стратегическая*, *укрупненная* и *детальная*. Все эти системы моделей, описывая основные аспекты организации и ее деятельности, базируются на бизнес-процессах. В систему моделей описания организации добавлена также дополнительная система моделей, для того чтобы можно было учесть аспекты, не связанные с бизнес-процессами, но необходимые при создании ИС.

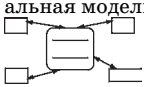
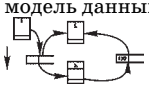
Назначение *стратегической системы моделей* заключается, во-первых, в определении основных целей и задач организации и, во-вторых, в формировании моделей бизнес-процессов, описывающих основные виды деятельности организации и реализующих ее стратегические цели и задачи.

В *укрупненной системе моделей* отображаются основные бизнес-процессы, которые описаны на стратегическом уровне исходя из основных целей и задач организации (без привязки к ее структуре) на реальную иерархически функциональную структуру организации, а также выделяются основные функции подразделений и уточняются состав и характеристики бизнес-процессов.

Все создаваемые далее модели организации строятся на базе построенных бизнес-процессов по результатам обследования деятельности организации, проводимого на уровне подразделений. Модели строятся с помощью CASE-средств и сохраняются в репозитории проекта. Построение этих моделей допускает распараллеливание работ при проведении обследования и построении моделей.

Главными целями создания детальной системы моделей является построение концептуальной модели данных и функциональной модели организации. Для достижения этих целей проводится детализация описания деятельности организации от уровня описания реализации общих

Схема преобразования

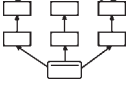
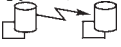
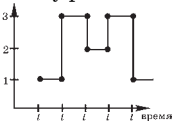
Уровень описания/ группы характеристик-моделей	Функции	Данные	Люди	
Укрупненная система моделей организации	Список бизнес-процессов и бизнес-функций организации 	Списки документов. Списки объектов 	Структурная модель организации 	
Детальная система моделей организации	Функциональные модели подразделений: диаграммы потоков данных 	Информационная модель организации (концептуальная модель) 	Структурные модели подразделений, роли персонала 	
Система моделей требований к ИУС	Требования к функциям; диаграмма потоков данных 	Требования к данным; структуры данных. Концептуальная модель данных 	Требования к регламенту и интерфейсу пользователей 	
Детализация проекта	↓	↓	↓	
Функционирующая система	Функции	Данные	Пользователи	

бизнес-процессов в организации и списковых моделей в подразделениях до уровня детальных моделей подразделений, позволяющих выделить все функции подразделений, обрабатываемые документы, основные данные, описать регламент работы персонала и создать в итоге функциональную модель организации и концептуальную модель данных.

Дальнейшее развитие систем согласованных моделей происходит на базе схемы преобразования моделей, представленной в таблице 6.2, которая описывает развитие согласованных моделей по основным аспектам, определяющим деятельность организации (данные, функции,

Таблица 6.2

моделей

	Сеть	**Время**	**Правила (мотивация)**
	Перечень структурных подразделений и внешних организаций 	Список работ во времени 	Список целей и задач организации. Критерии и правила выполнения бизнес-процессов 
	Логическая модель сетей подразделений организации и внешних связей 	Временная модель выполнения бизнес-процессов и бизнес-функций 	Критерии выполнения бизнес-функций; бизнес-правила 
	Требования к сетевой архитектуре системы 	Требования к временным характеристикам функции 	Требования к регламенту работы ИС 
	↓	↓	↓
	Коммуникации	Зависимость	Правила

люди, сеть, время, правила). Каждая строка таблицы соответствует системе согласованных моделей данного уровня и раскрывается в виде частных моделей по каждой группе характеристик. Переход от строки к строке показывает развитие системы моделей в процессе описания организации, а правила перехода определяются методологией и обеспечивают полноту и согласованность при построении моделей.

Технология проектирования от данных. Поскольку данные составляют основу деятельности любой организации и являются наиболее стабильной ее составляющей (функции и структура организации меняются гораздо чаще), то при построении корпоративной ИС наиболее адекватным решаемым задачам является подход к проектированию, основанный на данных. Такой подход обеспечивает наилучшее архитектурное решение при разбиении системы на приложения, а также простоту и согласованность при интеграции приложений.

Комплекс согласованных инструментальных средств. Предлагаемая методология создания ИС поддерживается комплексом согласованных между собой инструментальных средств, который обеспечивает непрерывный цикл автоматизации процессов, выполняемых на всех этапах ЖЦ ИС. Согласованность этих средств обеспечивается наличием интерфейсов для прямого взаимодействия и поддержкой общепринятых стандартов открытых систем. Методология и поддерживающий ее набор инструментальных средств обеспечивают полный контроль и гибкое управление ходом разработки, включая:

- поддержку коллективной разработки с возможностью параллельного и распределенного выполнения различных работ;
- возможность перехода к следующему этапу (шагу), не дожидаясь полного завершения предыдущего;
- применение методов контроля качества и постоянный контроль полученных результатов;
- поддержку итеративного характера разработки (возможность пересмотра полученных результатов и возврата на любой из предыдущих этапов;

- возможность быстрого внесения изменений в требования в процессе разработки;
- управление конфигурацией.

6.2. ТЕХНОЛОГИИ РЕАЛИЗАЦИИ ИНФОРМАЦИОННЫХ СИСТЕМ

Термин *компонент* в ИТ-отрасли используется для обозначения различных понятий. Во-первых, можно выделить аппаратные (hardware components) и программные компоненты (software components).

Термин *программный компонент* (ПрК) используется для обозначения двух связанных, но разных понятий. Если речь идет о программной архитектуре, то обычно под компонентом подразумевается программный модуль, реализующий некоторую функцию или набор функций и решает определенные подзадачи в рамках общих задач, решаемых системой. Это те компоненты, которые могут быть изображены на диаграммах компонентов (component diagram) в языке UML. Если речь идет о компонентных технологиях программирования или компонентно-ориентированной (component based) разработке программного обеспечения (ПО), то под ПрК понимают объекты со специальными свойствами. В дальнейшем термин «компонент» будет употребляться во втором смысле.

В этом случае понятие ПрК выступает в качестве ключевого для определения таких понятий, как компонентно-ориентированное программирование и компонентно-ориентированный подход к проектированию ИС.

Согласно [18] ПрК представляет собой структурную единицу программной системы, обладающую четко определенным интерфейсом, который полностью описывает ее зависимости от контекста.

ПК представляет собой откомпилированный автономный программный модуль, который можно объединять с другими модулями или кодом, организованным другим способом с целью создания приложений. ПрК могут быть простыми, такими как кнопка, или сложными, такими как компонент, реализующий управление сетью.

Чаще всего под ПрК понимают откомпилированный «двоичный компонент», который можно интегрировать в приложение на лету. При работе с компонентами исходный код обычно недоступен и поэтому компонент нельзя изменять. Под данное определение подпадают, в частности, динамические библиотеки.

Компонентно-ориентированное приложение обычно состоит из множества компонентов. Само приложение представляет собой некоторую среду, включающую средства «склеивания» или скелетный код, в который можно встраивать ПрК, которые могут взаимодействовать со средой и операционной системой и друг с другом. Изменение интерфейса ПрК приводит к изменению его кода, но изменение способа реализации не обязательно приводит к изменению интерфейса.

ПрК можно рассматривать и как пакет, ориентированный на повторное использование кода, который можно приобрести у независимого производителя.

Типовой ПрК обладает следующими свойствами:

- представляет собой фрагмент самодостаточного кода, т. е. для его функционирования не требуется наличия дополнительных библиотек;
- является самоустанавливаемым модулем, который может быть включен в состав системы, исключен из ее состава или заменен на другой модуль, например принадлежащий другому производителю, при минимальном участии пользователя;
- может повторно использоваться в различных контекстах;
- при работе с ПК используются механизмы динамического связывания;
- можно объединять с другими ПК с целью создания более крупных ПК;
- пользователи используют ПК преимущественно по принципу «черного ящика», т. е. пользователю известны только интерфейсы, но не внутренняя структура системы.

ПрК может быть независимо поставлен или не поставлен, добавлен в состав некоторой системы или удален из

нее, в том числе может включаться в состав систем других поставщиков.

Для функционирования ПрК, как правило, необходимо наличие соответствующей инфраструктуры, которая позволяет компонентам находить друг друга и взаимодействовать по определенным правилам. Набор правил, определяющих интерфейсы ПК и их реализаций, а также правил, по которым ПрК работают в системе и взаимодействуют друг с другом, называют *компонентной моделью* (component model) [18]. В компонентную модель входят также правила, регламентирующие жизненный цикл ПрК. Взаимодействовать друг с другом могут только ПК, построенные в рамках одной модели.

Для работы ПрК необходим некоторый набор *базовых служб* (basic services), которые обеспечивают, например, нахождение компонентов в распределенной среде, обеспечение обмена данными через сеть.

Набор таких базовых, необходимых для функционирования большинства компонентов служб, вместе с поддерживаемой с их помощью компонентной моделью называется *компонентной средой* (или *компонентным фрейворком*, component framework).

Компонентные технологии можно рассматривать как одну из фаз развития технологий разработки распределенных систем, при этом можно выделить следующие основные фазы (рис. 6.5):

- сокеты;
- вызов удаленных процедур;
- системы распределенных объектов;
- компонентные технологии;
- сервисно-ориентированные системы.

Перечисленные технологии появлялись именно в таком порядке. Переход к следующей фазе можно рассматривать как достижение некоторого уровня зрелости, поскольку технологии нижележащего уровня используются в качестве сервисов более низкого уровня. В частности, вызов удаленных процедур основывается на использовании сокетов, системы распределенных объектов, в свою очередь, базируются на вызове удаленных процедур, ком-

Рис. 6.5
Фазы развития технологий разработки распределенных систем

понентные технологии могут использовать такие механизмы, как RMI, а веб-сервисы могут быть реализованы как компоненты.

Известно достаточно много различных компонентных технологий. Некоторые из них остались на уровне теоретических исследований, однако ряд технологий активно используются на практике в течение уже многих лет. К последней группе можно отнести такие компонентно-ориентированные (component based) технологии, как JavaBeans, EJB, CORBA, ActiveX, VBA, COM, DCOM, .Net компоненты. Принципиально мультиагентные технологии также можно рассматривать как разновидность компонентных технологий.

Сокеты. Самым старым способом взаимодействия между элементами распределенных приложений являются сокеты, которые позволяют связывать приложения напрямую, записывая данные в сокет и читая данные из

сокета. API для сокетов является низкоуровневым и предоставляет максимальный контроль над передаваемыми данными.

Однако сокеты плохо подходят для работы со сложными структурами данных, особенно если это программные компоненты, поэтому разработка сложных приложений с использованием сокетов достаточно проблематична.

Работа с сокетами в самом общем виде выглядит следующим образом. Любой процесс может создать серверный сокет и привязать его к какому-нибудь порту. После этого сервер переходит в режим ожидания и ожидает запрос со стороны клиента.

Клиент также создает сокет, через который он может взаимодействовать с сервером. Взаимодействие может реализовываться как с установлением соединения, так и без установления соединения. В первом случае в качестве транспорта используется протокол TPC, а во втором — UDP.

Механизм работы с сокетами поддерживается практически во всех современных языках программирования.

Самую подробную информацию по программированию сокетов можно найти в [18].

Вызов удаленных процедур. Основная идея механизма сокетов состоит в использовании операций send и receive для обмена данными между процессами, которые выполняются на разных хостах. Основной недостаток сокетов состоит в сложности программирования, поскольку программирование осуществляется на низком уровне.

Идея вызова удаленных процедур состоит в обеспечении возможности программам вызывать процедуры, находящиеся на других машинах. Когда процесс, запущенный на хосте *A*, вызывает процедуру с хоста *B*, вызывающий процесс на хосте *A* приостанавливается, а выполнение вызванной процедуры происходит на хосте *B*. Информация может быть передана от вызывающего процесса к вызываемой процедуре через параметры и возвращена процессу в виде результата выполнения процедуры. Для программиста вызов удаленной процедуры ничем не

отличается от вызова локальной процедуры. Данный метод известен под названием удаленный вызов процедур (Remote Procedure Call, RPC).

Использование подпрограмм (процедур) в программе — традиционный способ структурировать задачу и сделать ее более ясной. Обычно подпрограммы собираются в библиотеки. Применительно к локальным вызовам данный подход используется давно и повсеместно.

Удаленный вызов процедуры существенным образом отличается от традиционного локального с точки зрения реализации, однако с точки зрения программиста такие отличия практически отсутствуют.

Среда распределенных вычислений (Distributed Computing Environment, DCE). Механизм удаленных вызовов процедур был тщательно адаптирован для использования в качестве основы систем промежуточного уровня и вообще распределенных систем и достаточно хорошо работал при использовании таких языков программирования, как C++. Дальнейшим развитием данного подхода можно считать *среду распределенных вычислений* (*Distributed Computing Environment*, *DCE*), разработанную организацией OSF (Open Software Foundation), которая позже была переименована в Open Group. DCE изначально она была разработана под UNIX, однако позже был были созданы версии для других ОС.

DCE включает ряд сервисов, таких как поддержка работы с нитями (Threads), вызов удаленных процедур и аутентификация, служба времени, служба каталогов, служба безопасности и файловый сервис. Архитектура DCE показана на рисунке 6.6.

Рис. 6.6
Архитектура DCE

Служба распределенных файлов (*distributed file service*) представляет собой всемирную файловую систему, предоставляющую прозрачные методы доступа к любому файлу системы одинаковым образом.

Служба каталогов (*directory service*) используется для отслеживания местонахождения любого из ресурсов системы. Служба каталогов позволяет процессу запрашивать ресурсы, не зная, где они находятся, если это не необходимо для процесса.

Служба защиты (*security sewice*) позволяет защищать ресурсы любого типа, кроме того, получение некоторых данных может быть открыто только тем, кому это разрешено.

Служба распределенного времени (*distributed time service*) предоставляет *механизмы* синхронизации часов различных хостов.

Модель программирования, лежащая в основе всей системы DCE, — это модель «клиент-сервер», в которой связь между клиентами и серверами осуществляется посредством использования RPC.

Более подробную информацию по DCE можно найти в [18].

В DCE распределенные объекты были добавлены достаточно искусственно в качестве расширения вызовов удаленных процедур. При этом клиент работает со ссылкой на удаленную процедуру для объекта. Фактически ссылки на объекты отсутствуют.

В качестве полноценной системы работы с распределенными объектами можно рассматривать Java RMI. В Java распределенные объекты интегрированы с языком.

Java поддерживает распределенные объекты в форме удаленных объектов, т. е. объектов, тело которых постоянно находится на одном и том же хосте, а интерфейсы доступны удаленным процессам. Интерфейсы реализованы обычным образом через заглушки, которые предоставляют интерфейсы, идентичные интерфейсам удаленных объектов, при этом заглушка для клиента представляется в виде локального объекта, находящегося в адресном пространстве клиента.

Объектная модель компонентов (COM). В самом общем виде идея COM состоит в том, что одна часть ПО должна получать доступ к сервисам, предоставляемым другой частью, причем используется доступ ко всем видам про-

граммных сервисов независимо от способа их реализации. COM используется стандартный механизм.

В COM любая часть программного обеспечения реализует свои сервисы как один или несколько объектов COM.

Каждый такой объект поддерживает один или несколько интерфейсов, состоящих из методов.

Каждый метод — это процедура или функция, которая выполняет требуемое действие и может быть вызвана программным обеспечением, использующим данный объект (клиентом объекта).

Клиенты получают доступ к сервисам объекта COM только через вызовы методов интерфейсов объекта — у них нет непосредственного доступа к данным объекта.

Объект COM может поддерживать более одного интерфейса, например объект COM, показанный на рисунке 6.7, имеет три интерфейса.

Каждый интерфейс включает множество методов. В число этих методов входят три стандартных метода, которые будут рассмотрены далее, и произвольное число методов, определяемых пользователем. Однажды определенный интерфейс нельзя изменять и дополнять. Если появляется необходимость изменить интерфейс, то создается новый интерфейс.

Рассмотрим пример. Допустим, разработчиками был создан объект COM для реализации проверки правописания, единственный интерфейс *ISpeller*, который кроме стандартных содержит три метода: *FindWord* (·); *AddWord* (·), *RemoveWord* (·), которые позволяют находить слово в словаре, добавлять слово в словарь и удалять слово из словаря соответственно.

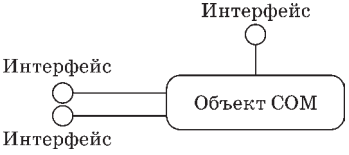


Рис. 6.7
Пример объекта COM

Если позднее появляется идея расширить функциональные возможности объекта COM за счет включения функции поиска синонимов, то, поскольку интерфейс нельзя изменять, можно создать новый интерфейс, ко-

торый можно назвать, например, *IThesaur*, который помимо трех стандартных методов содержит единственный метод *GetSynonym* (·).

COM с самого начала разрабатывалась как распределенная система, в которой компоненты можно создавать на другом хосте и их методы по сети (реально распределенная COM (Distributed COM — DCOM) появилась в 1996 г.). DCOM можно рассматривать как незначительное расширение оригинальной COM. По существу, DCOM добавляет к COM всего три основных элемента: способ создания удаленного объекта, протокол вызова методов удаленного компонента и механизмы обеспечения безопасного доступа к удаленному компоненту.

Технология COM+ появилась в рамках Windows 2000 и основывается, с одной стороны, на DCOM, а с другой стороны, на сервере транзакций (Microsoft Transaction Server). Основной целью создания COM+ следует считать разработку компонентной модели, которая могла эффективно использоваться в ИС уровня крупного предприятия.

Отличительными особенностями среды COM+ являются следующие:

- наличие эффективных механизмов работы с транзакциями;
- возможность реализации асинхронного взаимодействия с помощью очередей сообщений;
- наличие механизмов работы с событиями;
- улучшенные показатели безопасности;
- возможность работы с пулом объектов.

Все это позволяет создавать достаточно хорошо масштабируемые, стабильно работающие приложения уровня крупного предприятия.

Объекты COM+, являющиеся экземплярами компонент COM+, работают под управлением среды COM+. Набор связанных между собой компонент COM+, находящихся в одной динамической библиотеке, называется приложением COM+. Приложение COM+ включает множество компонент и ролей для доступа к ним. Информация о всех зарегистрированных приложениях находится в каталоге COM+.

.NET Framework. В 2002 г., т. е. всего через два года после появления COM+, была официально выпущена платформа Microsoft.NET, которая была объявлена Microsoft рекомендуемой основой для создания приложений и компонентов под Windows, в рамках которой используется собственная компонентная модель, радикально отличающаяся COM.

.NET Framework — программная платформа компании Microsoft, предназначенная как для создания обычных программ и веб-приложений. .NET является патентованной технологией.

В основу .NET Framework была положена амбициозная идея сделать платформонезависимую универсальную виртуальную машину, которая могла бы выполнять код, написанный на произвольном языке программирования в различных ОС без перекомпиляции кода. Однако со временем Microsoft ограничилась поддержкой только собственных ОС, предоставив независимым разработчикам заниматься поддержкой других платформ.

Основными составными частями .NET Framework являются инвариантная к языку программирования среда исполнения (common language runtime, CLR) и библиотека классов Framework (framework class library, FCL). CLR — это некоторая обертка для API ОС, которая служит средой для исполнения управляемых приложений (managed applications). FCL предоставляет объектно-ориентированный API, к которому обращаются управляемые приложения.

При работе с управляемыми приложениями программист теряет возможность работать с Windows API, MFC, ATL, COM и другими знакомыми инструментами и технологиями и должен работать с FCL.

Если программист не хочет отказываться от «старых» технологий, то ему предоставляется возможность создавать приложения, основанные на использовании неуправляемого кода, который исполняется без использования CLR. Совмещение в рамках одного приложения управляемого и неуправляемого кода возможно, но не приветствуется разработчиками платформы.

Инвариантная к языку программирования среда достигается за счет того, что среда разработки создает байткод, который интерпретируется виртуальной машиной. Встроенные (поставляются вместе с .NET Framework). В качестве основных языков, поддерживаемых платформой NET, выступают: C#, VB.NET, JScript .NET, C++/CLI, IronPython, IronRuby и F# (функциональный язык общего назначения).

В качестве входного языка виртуальной машины в .NET используется Common Intermediate Language (CIL) (в более ранних версиях он назывался Microsoft Intermediate Language (MSIL)).

Компонентная модель .NET. Компонент .NET представляет собой перекомпилированный MSIL, который может быть помещен в любой другой MSIL компонент или клиентскую программу без выполнения компиляции.

Компонент .NET представляет собой перекомпилированный самоописываемый MSIL-модуль, построенный на базе одного или более классов или модулей, расположенных в DLL-файле сборки (DLL assembly file) [18]. Сборки представляют собой базовые строительные блоки, которые обеспечивают развертывание и выполнение приложений .NET.

Исполняемый файл, библиотека или файл ресурсов, созданный на платформе .NET, представляет собой сборку. Кроме того, сборки обеспечивают реализацию механизма работы с версиями, определяют область действия типов и ресурсов, управляют правами доступа.

Технология CORBA. *Основы CORBA* (Component Object Request Broker Architecture) — это стандарт, определенный группой компаний OMG (Object Management Group), который позволяет различным программным компонентам, написанным на разных языках программирования и работающих на разных машинах, взаимодействовать друг с другом.

Спецификация CORBA лежит в основе всех стандартов, разработанных OMG, и определяет механизмы взаимодействия приложений в распределенной системе.

Ключевыми компонентами данного стандарта являются объектный брокер запросов и язык определения интерфейсов IDL. Следующими по важности элементами являются сервис динамического формирования запросов, репозиторий интерфейсов, сервис динамической обработки запросов и сервис, обеспечивающий взаимодействие различных брокеров запросов. Основная идея состоит в том, что приложение представляется в виде множества объектов. Каждый из объектов содержит информацию о возможностях кода и способах его вызова. Объект может быть вызван локально или через сеть из других приложений или объектов CORBA.

Обобщенная структура системы, построенной с использованием CORBA, показана на рисунке 6.8. Основу CORBA-системы составляет «системная шина», которая позволяет компонентам, реализованным на разных языках и работающих на разных платформах, находить друг друга и взаимодействовать. Эту шину называют брокером объектных запросов (Common Object Request Broker Architecture) или просто ORB.

Интерфейсы в CORBA определяются средствами языка IDL (Interface Definition Language — язык описания интерфейса). IDL описание используется для генерации кода заглушек для поддерживаемых языков программирования (Java, C++, C, COBOL, Lisp, PL/I, Smalltalk, Python). Сгенерированный код используется как основа для доступа к объекту в определенной программной среде.

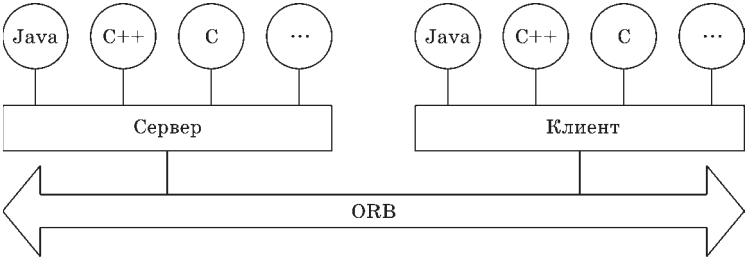


Рис. 6.8
Обобщенная структура системы, построенной с использованием CORBA

Наибольшая популярность технологии CORBA пришлась на последние годы прошлого века и первые годы текущего века. CORBA всегда рассматривалась как «тяжелая» технология, ориентированная на решение сложных задач, преимущественно в сфере крупных корпоративных систем. Однако CORBA далеко не всегда используется исключительно в крупных приложениях. Специализированные версии CORBA до сих пор работают в системах реального времени и малых встраиваемых системах.

С точки зрения реализации CORBA является спецификацией; она предоставляет описание для реализации конкретных продуктов. Существует множество компаний и сообществ, поставляющих продукты CORBA для различных языков программирования. Список поставщиков можно найти на сайте CORBA. Глобальная архитектура CORBA восходит к модели OMG, в рамках которой выделяются четыре группы архитектурных элементов (рис. 6.9):

- брокер объектных запросов;
- сервисы CORBA;
- средства CORBA;
- прикладные объекты и приложения.

Брокер объектных запросов (Object Request Broker — ORB) определяет объектную шину CORBA. Сервисы CORBA (CORBA Services) определяют системный уровень объектной структуры и могут рассматриваться как расширение объектной шины, общие средства CORBA (CORBA Facilities) определяют сервисы, которые непосредственно используются приложениями (бизнес-объектами), Application Objects представляют прикладные объекты и приложения, т. е. конечных потребителей инфраструктуры CORBA.

Каждый нижележащий уровень выступает как сервер для вышележащего.

Рис. 6.9
Группы архитектурных элементов CORBA

В основе CORBA лежит понятие *объектной модели*. Объектная модель описывает объектные концепции и терминологию, применимые к клиентскому коду и реализации объектов. Объектная модель CORBA являет собой пример классической объектной модели, в которой клиенты посылают сообщения объектам, а объекты интерпретируют сообщения и выполняют соответствующие им действия.

В основе объектной системы лежит понятие *объекта* (*object*). *Объект* — идентифицируемая, инкапсулированная сущность, предоставляющая один или более сервисов, которые могут быть запрошены клиентами.

Технология Enterprise Java Beans (EJB). Она представляет собой высокоуровневый подход к построению распределенных приложений масштаба предприятия. EJB — это модель серверных компонентов (server component model) для Java.

Основная идея технологии Enterprise JavaBeans состоит в создании такой инфраструктуры для компонент, чтобы они могли легко добавляться (plug in) и удаляться из серверов, тем самым расширяя или ограничивая функциональность сервера.

Краткая история. Фирма Sun Microsystems Inc. анонсировала технологию EJB в 1996 г., а в 1998 г. была представлена первая реализация (версия 1.0). На момент написания книги последней была версия EJB 3.0, которая появилась в 2006 г.

Версии. EJB 1.0. Это начальная версия, в рамках которой поддерживались stateful и stateless компонент, с которыми можно было работать через удаленный интерфейс. В качестве желательной опции рассматривались компоненты, имеющие состояния.

EJB 1.1. В данном релизе в качестве обязательных появились компоненты с состоянием и XML дескриптор (deployment descriptor).

EJB 2.0. В данном релизе в дополнение к удаленному интерфейсу появился локальный интерфейс, который могли использовать компоненты, работающие в том же контейнере. Появился новый тип компонентов — компоненты, управляемые сообщениями the message-driven

bean (MDB), которые могли принимать асинхронные сообщения. Для компонентов с сохранением была определена возможность сохранения состояния средствами контейнера. Появился язык Enterprise JavaBeans Query Language (EJB QL), похожий по структуре на SQL и предназначенный для поиска компонентов.

EJB 2.1. В рамках данного релиза появилась поддержка работы с веб-сервисами, расширились возможности EJB QL и в дескрипторе развертывания вместо DTD стала использоваться XML schema.

Можно заметить, что основные различия между версиями состоят в наращивании возможностей.

EJB 3. Данный релиз радикально отличается от предшествующих и определяет фактически новую компонентную модель. Основное отличие состоит в том, что пользователь не должен ничего знать о домашнем интерфейсе (Home Interface). В рамках данной спецификации используется идея POJO (Plain Old Java Objects). Идея состоит в том, что программист пишет только ту часть кода, которая реализует бизнес-логику. Для того чтобы определить, каким именно способом реализуется бизнес-код, используется механизм аннотаций. Например, если требуется определить интерфейс как локальный, то достаточно описать интерфейс обычным способом и вставить в строку аннотацию @Local. Механизм аннотаций позволяет очень существенно упростить процесс разработки.

Таким образом, речь идет о двух разных компонентных моделях и разных философиях разработки компонентом.

Использование EJB позволяет упростить процесс разработки КИС за счет того, что многие функции реализуются контейнерами. Кроме того, EJB можно использовать многократно, причем для повторного использования не требуется перекомпиляция всего проекта.

EJB компоненты хорошо сопрягаются с веб-сервисами и с CORBA.

Архитектурная модель EJB включает в себя следующие элементы (рис. 6.10):

- JEE контейнер;
- EJB контейнер;

Рис. 6.10
Архитектурная модель EJB

- веб-контейнер;
- клиент;
- база данных (БД).

Одним из побудительных мотивов создания EJB технологии были сложности CORBA. Однако эволюция технологии EJB привела к тому, что эта технология стала казаться разработчикам чрезмерно сложной, в частности:

- чрезмерно тяжеловесная модель программирования, требующая работы с несколькими интерфейсами;
- необходимость непосредственного взаимодействовать с Java Naming Directory Interface (JNDI);
- необходимость работать с непомерно сложным XML DD.

Эти недостатки устранены в EJB 3.0, где используется только один бизнес-интерфейс вместо home и remote интерфейсов, минимизируется использование DD за счет использования аннотаций. Кроме того, используется новый механизм для сохранения состояния для компонентов-сущностей (Entity Beans).

Одной из новых черт EJB 3 является использование Java Persistence API (JPA) для реализации сохранения состояния.

Сервис-ориентированная архитектура (service-oriented architecture, SOA). Это подход к созданию ИС, основанный использовании *сервисов* или *служб* (service). Ниже термины «сервис» и «служба» рассматриваются как синонимы. СОА — это в первую очередь интеграционная архитектура, использование которой позволяет обеспечить гибкую интеграцию ИС. При использовании СОА приложения взаимодействуют, вызывая сервисы, входящие в состав других приложений. Сервисы объединяются

в более крупные последовательности, реализуя бизнес-процессы, которые могут быть доступны как сервисы.

СОА можно рассматривать также как подход к построению слабосвязанных (loosely coupled) систем, реализующих механизмы асинхронного взаимодействия. К слабосвязанным системам обычно относят такие системы, как электронная почта и системы очередей сообщений.

Переход на СОА архитектуры позволяет решать следующие задачи:

- сократить сроки освоения и внедрения новых ИТ-систем, быстро создавать новые ИТ-системы на базе уже существующих;
- уменьшить суммарную стоимость владения ИТ-продуктом и стоимость их интеграции;
- увеличить срок жизни ИТ-систем за счет возможности их оперативной модернизации;
- использовать гибкие модели ценообразования путем передачи разработки детализированных бизнес-модулей сторонним производителям (аутсорсинг);
- уменьшить стоимость работ по интеграции, необходимой при слиянии и поглощении компаний;
- осуществить реализацию бизнес-процессов на уровне, не зависящем от приложений и платформ, используемых для поддержки процессов.

СОА — это интеграционная архитектура, основанная на концепции сервисов (служб).

Бизнес-функции и инфраструктурные функции, которые необходимы для построения распределенных систем, реализуются как сервисы, которые в сочетании или по отдельности обеспечивают прикладную функциональность либо другим сервисам, либо приложениям, взаимодействующим с конечным пользователем.

Концепция СОА предполагает использование единого механизма взаимодействия служб. Этот механизм строится на основе концепции свободных связей и должен поддерживать использование формальных интерфейсов.

СОА привносит преимущества, которые дают слабосвязанность и инкапсуляция, в интеграцию на уровне предприятия.

Сервисы можно рассматривать как строительные блоки, которые могут использоваться как для построения сервисов более высокого уровня, так и для построения законченных распределенных ИТ-систем.

Сервисы могут вызываться независимо внешними или внутренними потребителями для выполнения элементарных функций либо могут объединяться в цепочки для формирования более сложных функций и для быстрого создания новых функций.

Используя СОА, организации могут создавать гибкие КИС, которые позволяют оперативно реализовывать быстро изменяющиеся бизнес-процессы и многократно использовать одни и те же компоненты в рамках одной ИТ-системы, в рамках семейств продуктов и в независимых ИТ-системах.

6.3. ОЦЕНКА КАЧЕСТВА ИНФОРМАЦИОННЫХ СИСТЕМ

Понятие качества ИС соответствует понятию о том, что система успешно справляется со всеми возлагаемыми на нее задачами, имеет хорошие показатели надежности, приемлемую стоимость, удобна в эксплуатации и обслуживании, легко сопрягается с другими системами и в случае необходимости может быть модифицирована.

Разные группы пользователей имеют различные точки зрения на характеристики качества ИС. Например, если задать вопрос о том, какой должна быть хорошая ИС, то от пользователя можно скорее всего получить следующие варианты ответов:

- система имеет хорошую производительность;
- система имеет широкие функциональные возможности;
- система удобна в эксплуатации;
- система надежна.

Менеджер даст скорее всего другие варианты ответов:

- система не должна быть очень дорогой;
- система не должна быть очень дорогой в эксплуатации;
- система не должна морально устаревать в течение возможно более длительного промежутка времени, а

в случае необходимости может быть легко модифицирована.

Для системного администратора наиболее важными могут оказаться такие характеристики системы, как:

- надежность и стабильность работы;
- простота администрирования;
- наличие хорошей эксплуатационной документации;
- хорошая поддержка изготовителем.

Другие заинтересованные лица могут иметь свои точки зрения на то, какой должна быть качественная система.

Качество ИС связано с дефектами, заложенными на этапе проектирования и проявляющимися в процессе эксплуатации. Любые свойства ИС, в том числе и дефектологические, могут проявляться лишь во взаимодействии с внешней средой, включающей технические средства, персонал, информационное и программное окружение.

В зависимости от целей исследования и этапов жизненного цикла ИС дефектологические свойства разделяют на дефектогенность, дефектабельность и дефектоскопичность.

Дефектогенность определяется влиянием следующих факторов:

- количество разработчиков ИС, их профессиональные и психофизиологические характеристики;
- условия и организация процесса разработки ИС;
- характеристики инструментальных средств и компонент ИС;
- сложность задач, решаемых ИС;
- степень агрессивности внешней среды (потенциальная возможность внешней среды вносить преднамеренные дефекты, например воздействие вирусов).

Дефектабельность характеризует наличие дефектов ИС и определяется их количеством, местонахождением и другими факторами, влияющими на дефектабельность:

- структурно-конструктивные особенности ИС;
- интенсивность и характеристики ошибок, приводящих к дефектам.

Дефектоскопичность характеризует возможность проявления дефектов в виде отказов и сбоев в процессе от-

ладки, испытаний или эксплуатации. На дефектоскопичность влияют:

- количество, типы и характер распределения дефектов в ИС;
- устойчивость ИС к проявлению дефектов;
- характеристики средств контроля и диагностики дефектов;
- квалификация обслуживающего персонала.

Оценка качества ИС является крайне сложной задачей ввиду многообразия интересов пользователей. Поэтому невозможно предложить одну универсальную меру качества и приходится использовать ряд характеристик, охватывающих весь спектр предъявляемых требований. Наиболее близки к задачам оценки качества ИС модели качества программного обеспечения, являющегося одной из важных составных частей ИС. В настоящее время используется несколько абстрактных моделей качества программного обеспечения, основанных на определениях характеристики качества, показателя качества, критерия и метрики.

Критерий может быть определен как независимый атрибут ИС или процесса ее создания. С помощью такого критерия может быть измерена характеристика качества ИС на основе той или иной метрики. Совокупность нескольких критериев определяет показатель качества, формируемый исходя из требований, предъявляемым к ИС. В настоящее время наибольшее распространение получила иерархическая модель взаимосвязи компонент качества ИС. Вначале определяются характеристики качества, в числе которых могут быть, например, общая полезность, исходная полезность, удобство эксплуатации. Далее формируются показатели, к числу которых могут быть отнесены: практичность, целостность, корректность, удобство обслуживания, оцениваемость, гибкость, адаптируемость, мобильность, возможность взаимодействия. Каждому показателю качества ставится в соотвествие группа критериев. Для указанных выше показателей ниже приведены возможные критерии. Надо отметить, что один и тот же критерий может характеризовать несколько показателей.

С помощью метрик можно дать количественную или качественную оценку качества ИС. Различают следующие виды метрик и шкал для измерения критериев.

Первый тип — метрики, которые используют интервальную шкалу, характеризуемую относительными величинами или реально измеряемыми физическими показателями, например временем наработки на отказ, вероятностью ошибки, объемом информации и др.

Второй тип — метрики, которым соответствует порядковая шкала, позволяющая ранжировать характеристики путем сравнения с опорными значениями.

Третий тип — метрики, которым соответствуют номинальная или категорированная шкала, определяющая наличие рассматриваемого свойства или признака у рассматриваемого объекта без учета градаций по этому признаку. Так, например, интерфейс может быть «простым для понимания», «умеренно простым», «сложным для понимания».

Развитием иерархического подхода является представленная на рисунке 6.11 модель классификации критериев качества информационных систем. С помощью функциональных критериев оценивается степень выполнения ИС основных целей или задач. Конструктивные критерии предназначены для оценки компонент ИС, не зависящих от целевого назначения.

Поскольку в современных ИС ключевой компонентой является программная компонента, а пользователи, работающие с системой, в подавляющем большинстве случаев взаимодействуют непосредственно с программной компонентой, поэтому показатели качества информационных и программных систем в значительной степени совпадают.

Качество программного обеспечения определяется в стандарте ISO 9126 как вся совокупность его характеристик, относящихся к возможности удовлетворять высказанные или подразумеваемые потребности всех заинтересованных лиц.

Различаются понятия внутреннего качества, связанного с характеристиками программного обеспечения (ПО) самого по себе, без учета его поведения; внешнего качества, характеризующего ПО с точки зрения его поведения; и ка-

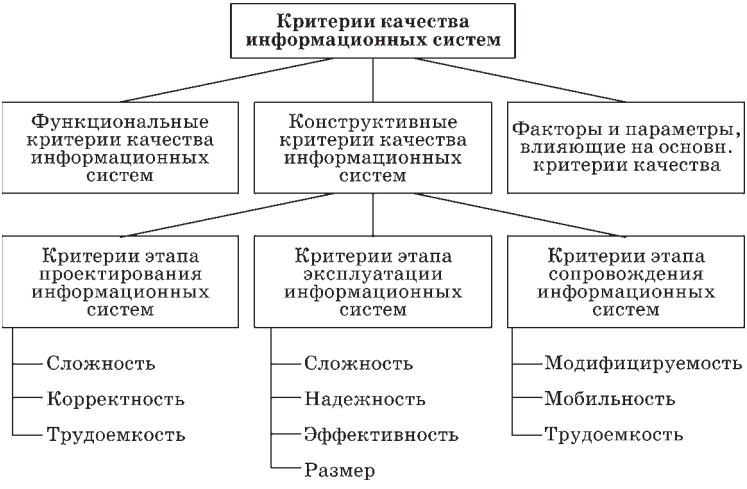


Рис. 6.11
Модель классификации критериев качества информационных систем

чества ПО при использовании в различных контекстах — того качества, которое ощущается пользователями при конкретных сценариях работы ПО. Для всех этих аспектов качества введены метрики, позволяющие оценить их. Кроме того, для создания добротного ПО существенно качество технологических процессов его разработки.

ISO 9126 — это международный стандарт, определяющий оценочные характеристики качества программного обеспечения. Российский аналог стандарта ГОСТ 28195. Стандарт делится на четыре части, описывающие следующие вопросы: модель качества; внешние метрики качества; внутренние метрики качества; метрики качества в использовании.

Вторая и третья части стандарта ISO 9126-2, 3 посвящены формализации соответственно внешних и внутренних метрик характеристик качества сложных программных систем. В них изложены содержание и общие рекомендации по использованию соответствующих метрик и взаимосвязей между типами метрик.

Четвертая часть стандарта ISO 9126-4 предназначена для покупателей, поставщиков, разработчиков, сопро-

вождающих, пользователей и менеджеров качества ПС. В ней повторена концепция трех типов метрик, а также аннотированы рекомендуемые виды измерений характеристик. Модель качества, установленная в первой части стандарта ISO 9126-1, классифицирует качество ПО в шести структурных наборах характеристик:

- функциональность;
- надежность;
- эффективность (производительность);
- удобство использования;
- удобство сопровождения;
- переносимость.

Перечисленные характеристики, в свою очередь, детализированы подхарактеристиками (субхарактеристиками) (рис. 6.12). Ниже приведены определения этих характеристик и атрибутов по стандарту ISO 9126:2001.

Функциональность (functionality) определяется как способность ПО в определенных условиях решать задачи, нужные пользователям.

Для данной характеристики выделяются следующие субхарактеристики:

- функциональная пригодность;
- точность;
- способность к взаимодействию;
- защищенность;
- соответствие стандартам и правилам.

Функциональная пригодность (*suitability*) — определяется как способность решать нужный набор задач.

Точность (*accuracy*) — определяется как способность выдавать нужные результаты.

Способность к взаимодействию (*interoperability*) — это способность взаимодействовать с нужным набором других систем.

Соответствие стандартам и правилам (*compliance*) — соответствие ПО имеющимся индустриальным стандартам, нормативным и законодательным актам, другим регулирующим нормам.

Защищенность (*security*) — определяется как способность предотвращать неавторизированный, т. е. без ука-

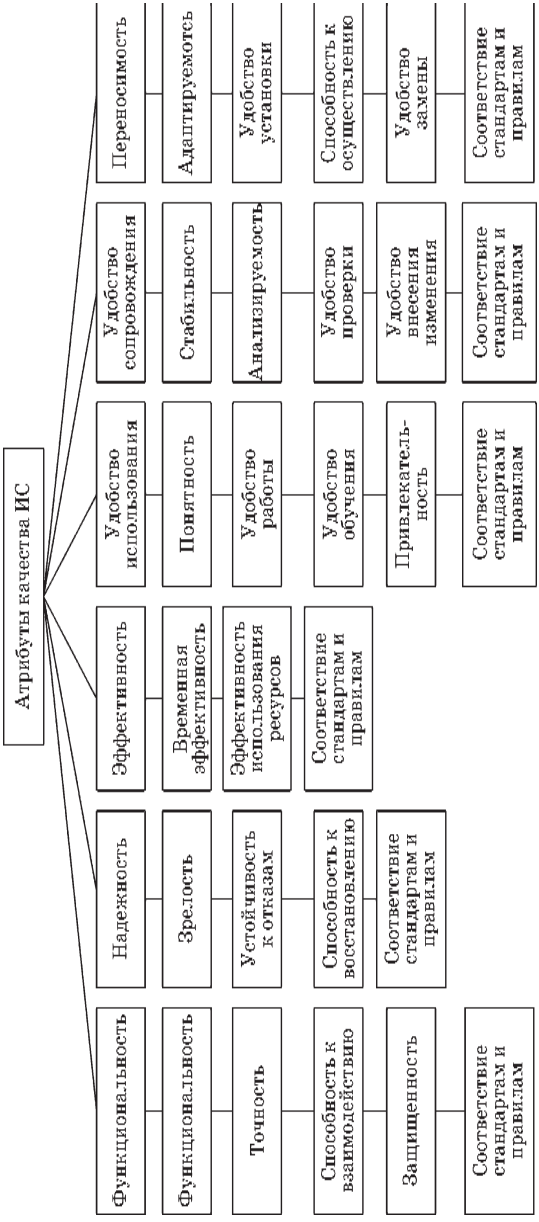


Рис. 6.12
Характеристики качества ИС

зания лица, пытающегося его осуществить, и неразрешенный доступ к данным и программам.

Надежность (reliability) — это способность ПО поддерживать определенную работоспособность в заданных условиях.

Для данной характеристики выделяются следующие субхарактеристики:

- зрелость;
- устойчивость к отказам;
- способность к восстановлению;
- соответствие стандартам.

Зрелость, завершенность (*maturity*) — величина, обратная частоте отказов ПО. Обычно измеряется средним временем работы без сбоев и величиной, обратной вероятности возникновения отказа за данный период времени.

Устойчивость к отказам (*fault tolerance*) — способность поддерживать заданный уровень работоспособности при отказах и нарушениях правил взаимодействия с окружением.

Способность к восстановлению (*recoverability*) — определяется как способность восстанавливать определенный уровень работоспособности и целостность данных после отказа, включает необходимые для этого время и ресурсы.

Соответствие стандартам надежности (*reliability compliance*).

Удобство использования (usability) или практичность определяется как способность ПО быть удобным в обучении и использовании, а также привлекательным для пользователей.

Для данной характеристики выделяются следующие субхарактеристики:

- понятность;
- удобство обучения;
- удобство работы;
- привлекательность;
- соответствие стандартам.

Понятность (*understandability*) — это показатель, обратный усилиям, которые затрачиваются пользовате-

лями на восприятие основных понятий ПО и осознание их применимости для решения своих задач.

Удобство обучения (*learnability*) — показатель, обратный усилиям, затрачиваемым пользователями на обучение работе с ПО.

Удобство работы (*operability*) — это показатель, обратный усилиям, предпринимаемым пользователями для решения своих задач с помощью ПО.

Привлекательность (*attractiveness*) — это способность ПО быть привлекательным для пользователей. Этот атрибут добавлен в 2001 г.

Соответствие стандартам определяется как удобство использования (*usability compliance*).

Производительность (efficiency) или эффективность — это способность ПО при заданных условиях обеспечивать необходимую работоспособность по отношению к выделяемым для этого ресурсам. Можно определить ее и как отношение получаемых с помощью ПО результатов к затрачиваемым на это ресурсам всех типов.

Для данной характеристики выделяются следующие субхарактеристики:

- временная эффективность;
- эффективность использования ресурсов;
- соответствие стандартам.

Временная эффективность (*time behaviour*) — способность ПО выдавать ожидаемые результаты, а также обеспечивать передачу необходимого объема данных за отведенное время.

Эффективность использования ресурсов (*resource utilisation*) — способность решать нужные задачи с использованием определенных объемов ресурсов определенных видов. Имеются в виду такие ресурсы, как оперативная и долговременная память, сетевые соединения, устройства ввода и вывода и пр.

Соответствие стандартам производительности (*efficiency compliance*).

Удобство сопровождения (maintainability) определяется как удобство проведения всех видов деятельности, связанных с сопровождением программ.

Для данной характеристики выделяются следующие субхарактеристики:

- анализируемость;
- удобство внесения изменений;
- стабильность;
- удобство проверки;
- соответствие стандартам.

Анализируемость (*analyzability*) или удобство проведения анализа — определяется как удобство проведения анализа ошибок, дефектов и недостатков, а также удобство анализа необходимости изменений и их возможных последствий.

Удобство внесения изменений (*changeability*) — показатель, обратный трудозатратам на выполнение необходимых изменений.

Стабильность (*stability*) — это показатель, обратный риску возникновения неожиданных эффектов при внесении необходимых изменений.

Удобство проверки (*testability*) — это показатель, обратный трудозатратам на проведение *тестирования* и других видов проверки того, что внесенные изменения привели к нужным результатам.

Соответствие стандартам удобства сопровождения (*maintainability compliance*).

Переносимость (portability) — определяется как способность ПО сохранять работоспособность при переносе из одного окружения в другое, включая организационные, аппаратные и программные аспекты окружения.

Иногда эта характеристика называется в русскоязычной литературе мобильностью. Однако термин «мобильность» стоит зарезервировать для перевода «mobility» — способности ПО и компьютерной системы в целом сохранять работоспособность при ее физическом перемещении в пространстве.

Для данной характеристики выделяются следующие субхарактеристики:

- адаптируемость;
- удобство установки;
- способность к сосуществованию;

- удобство замены;
- соответствие стандартам.

Адаптируемость (*adaptability*) — способность ПО приспосабливаться к различным окружениям без проведения для этого действий, помимо заранее предусмотренных.

Удобство установки (*installability*) — это способность ПО быть установленным или развернутым в определенном окружении.

Способность к сосуществованию (*coexistence*) — это способность ПО сосуществовать с другими программами в общем окружении, деля с ними ресурсы.

Удобство замены (*replaceability*) другого ПО данным — определяется как возможность применения данного ПО вместо других программных систем для решения тех же задач в определенном окружении.

Соответствие стандартам *переносимости* (*portability compliance*).

Перечисленные атрибуты относятся к внутреннему и внешнему *качеству ПО* согласно ISO 9126. Для описания *качества ПО* при использовании стандарт ISO 9126-4 предлагает другой, более узкий набор характеристик:

- эффективность;
- продуктивность;
- безопасность;
- удовлетворение пользователей;
- эффективность (effectiveness).

Способность ПО предоставлять пользователям возможность решать их задачи с необходимой точностью при использовании в заданном контексте.

Продуктивность (*productivity*) — это способность ПО предоставлять пользователям определенные результаты в рамках ожидаемых затрат ресурсов.

Безопасность (*safety*) — это способность ПО обеспечивать необходимо низкий уровень риска нанесения ущерба жизни и здоровью людей, бизнесу, собственности или окружающей среде.

Удовлетворение пользователей (*satisfaction*) — это способность ПО приносить удовлетворение пользователям при использовании в заданном контексте.

Одним из путей обеспечения качества ИС является сертификация. В США Радиотехническая комиссия по аэронавтике в своем руководящем документе определяет процесс сертификации следующим образом: «Сертификация — процесс официального утверждения государственным полномочным органом выполняемой функции системы путем удостоверения, что функция удовлетворяет всем требованиям заказчика, а также государственным нормативным документам». К сожалению, в настоящее время не существует стандартов, полностью удовлетворяющих оценке качества ИС. В западноевропейских странах имеется ряд стандартов, определяющих основы сертификации программных систем. Стандарт Великобритании (BS 750) описывает структурные построения программных систем, гарантом которых может быть документ — соблюдение качества на государственном уровне. Имеется международный аналог указанного стандарта (ISO 9000) и аналог для стран-членов НАТО (AQAP1). Существующая в нашей стране система нормативно-технических документов относит программное обеспечение к «продукции производственно-технического назначения», которая рассматривается как материальный объект. Однако программное обеспечение является скорее абстрактной нематериальной сферой. Существующие ГОСТы, например ГОСТ 28195-89 «Оценка качества программных средств. Общие положения», явно устарели и являются неполными.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение системы.
2. Укажите основные фазы проектирования.
3. В чем суть каскадной схемы проектирования информационных систем?
4. Раскройте содержание итерационной (поэтапной) модели.
5. Укажите основные преимущества схемы непрерывной разработки.
6. Приведите классификацию методов проектирования информационных систем.

7. Раскройте содержание стадий проектирования информационных систем.
8. Сформулируйте основные понятия системного подхода.
9. В чем назначение типовых проектных решений?
10. Раскройте содержание стилей проектирования информационных систем.
11. Укажите особенности методологии RAD.
12. Каково назначение объектно-ориентированного подхода?
13. Раскройте структуру профилей информационных систем.
14. Перечислите основные фазы жизненного цикла информационной системы.
15. Дайте краткую характеристику технологии анализа информационных систем на основе бизнес-процессов.
16. Что такое программный компонент?
17. Каковы особенности типового программного компонента?
18. Перечислите основные фазы компонентных технологий и дайте их характеристику.
19. Каковы особенности объектной модели компонентов?
20. В чем преимущества программной платформы .NET Framework?
21. Раскройте содержание технологии CORBA.
22. Что представляет собой технология Enterprise Java Beans?
23. Каковы особенности сервис-ориентированной архитектуры?
24. Дайте определение дефектогенности, дефектабельности и дефектоскопичности.
25. Укажите основные критерии качества информационных систем.
26. Перечислите характеристики качества по стандарту ISO 9126:2001.

СПИСОК ЛИТЕРАТУРЫ

1. *Граничин*, *О.* Информационные технологии в управлении / О. Граничин, В. Кияев [Электронный ресурс]. — Электрон. дан. — Режим доступа: http:// intuit.ru/studies/courses/1055/271/info.
2. *Дмитриев*, *В. И.* Прикладная теория информации. — М. : Высш. шк., 1989. — 320 с.
3. *Казиев*, *В. М.* Информация: понятия, виды, получение, измерение и проблема обучения // Информатика и образование. — 2000. — № 4. — С. 12–22.
4. *Макарова*, *Н. В.* Информатика. — М. : Финансы и статистика, 2000. — 702 с.
5. *Дрот*, *В. Л.* Толковый словарь современной компьютерной лексики / В. Л. Дрот, Ф. А. Новиков. — СПб. : БХВ-Санкт-Петербург, 2004. — 608 с.
6. *Борзенко*, *А.* Новые технологии в центрах обработки данных [Электронный ресурс]. — Электрон. дан. — Режим доступа: http:// bytemag.ru/arnicles/detail.
7. *Грей*, *Дж.* Управление данными: прошлое, настоящее и будущее // Системы управления базами данных. — 1998. — № 38. — С. 71–80.
8. *Советов*, *Б. Я.* Теория информационных процессов и систем / Б. Я. Советов, В. А. Дубенецкий, В. В. Цехановский [и др.]. — М. : ИЦ «Академия», 2010. — 432 с.
9. *Головин*, *Ю. А.* Информационные сети / Ю. А. Головин, А. А. Суконщиков, С. А. Яковлев. — М. : ИЦ «Академия», 2013. — 384 с.

10. *Назаров, С. В.* Компьютерные технологии обработки информации. — М. : Финансы и статистика, 1999. — 249 с.
11. *Петровский, А. Б.* Теория принятия решений. — М. : ИЦ «Академия», 2009. — 400 с.
12. *Советов, Б. Я.* Интеллектуальные системы и технологии / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — М. : ИЦ «Академия», 2013. — 320 с.
13. *Советов, Б. Я.* Базы данных: теория и практика / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — М. : Юрайт, 2013. — 464 с.
14. *Падерно, П. И.* Введение в проектирование интеллектуальных интерфейсов / П. И. Падерно, С. Ф. Сергеев, Н. А. Назаренко. — СПб. : СПбГУ ИТМО, 2011. — 108 с.
15. *Кошкарев, А. В.* Геоинформатика / А. В. Кошкарев, В. С. Тикунов. — М. : Картоцентр-Геоиздат, 1998. — 312 с.
16. *Ведров, А. М.* CASE-технологии. Современные методы и средства проектирования информационных систем [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://citforum.ru/database/case/index.shtml.
17. *Новоженов, Ю. В.* Объектно-ориентированные CASE-средства / Ю. В. Новоженов, М. З. Звонкин, Н. Н. Тимонин // Системы управления базами данных. — 1996. — № 5–6. — С. 119–125.
18. *Советов, Б. Я.* Архитектура информационных систем / Б. Я. Советов, А. И. Водяхо, В. А. Дубенецкий [и др.]. — М. : ИЦ «Академия», 2010. — 288 с.
19. *Поспелов, Г. С.* Искусственный интеллект — основа новой информационной технологии. — М. : Высш. шк., 1988. — 288 с.
20. *Советов, Б. Я.* Представление знаний в информационных системах / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — М. : ИЦ «Академия», 2012. — 144 с.
21. *Коварцев, А. Н.* Современные технологии разработки и тестирования программного обеспечения (ПО). —

Самара : Самарский гос. аэрокосмический ун-т, 2000. — Ч. 1 : Разработка ПО средствами технологии графосимволического программирования : метод. указ. — 41 с.
22. *Орлик, С.* Основы программной инженерии [Электронный ресурс]. — Электрон. дан. — Режим доступа: http//swebok.sorlik.ru.
23. *Соловьев, С. В.* Технология разработки прикладного программного обеспечения / С. В. Соловьев, Р. И. Цой, Л. С. Гринкруг. — М. : Изд-во «Академия Естествознания», 2011. — 407 с.
24. ГОСТ Р ИСО/МЭК 15910-2002. Информационная технология. Процесс создания документации пользователя программного средства : [Текст]. — Введ. 2002-06-25. — М. : Изд-во стандартов, 2002.
25. ГОСТ 19781-90. Единая система программной документации. Обеспечение систем обработки информации программное : [Текст]. — Введ. 1992-01-01. — М. : Стандартинформ, 2005.
26. *Морозов, А. В.* Тестирование в современном образовательном процессе. проблемы проектирования тестов / А. В. Морозов, И. А. Кочанов. — [Электронный ресурс]. — Режим доступа: http://expo.smolensk.ru/InfoKom09/dokald_09/morozov.doc.
27. *Терехов, А. Н.* Введение в технологию программирования [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://intuit.ru/goods_store/ebooks/8300.
28. *Креймер, М.* Облачные технологии: реальность нашей жизни [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://relax.dviger.com/gallery/work/c_22335.html.
29. *Сидоров, В.* Computing уходит в небо, или Что такое «облачные вычисления»? [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://netler.ru/pc/cloud.htm.
30. Об облачных вычислениях — преимущества облачных технологий [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://www.parallels.com/ru/spp/understandingclouds/.

31. *Уваров, С.* Облачные технологии [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://www.ixbt.com/cm/cloud-computing.shtml.
32. Облачные технологии как будущее виртуального мира // Актуальная правда: конструктивная информация [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://www.apravda.com/node/531.
33. *Черняк, Л.* Большие данные — новая теория и практика // Открытые системы. — 2011. — № 10. — С. 18–26.
34. *Дубова, Н.* Большие данные крупным планом // Открытые системы. — 2011. — № 10. — С. 30–34.
35. *Черняк, Л.* Большие данные с четырех сторон // Открытые системы. — 2011. — № 10. — С. 34–40.
36. Большие данные глазами уверенного пользователя [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://innocrowd.ru/big-data-essentials.
37. *Дубенецкий, В. А.* Проектирование корпоративных информационных систем / В. А. Дубенецкий, Б. Я. Советов, В. В. Цехановский. — СПб. : Изд-во СПбГЭТУ «ЛЭТИ», 2013. — 204 с.
38. *Лав, Р.* Разработка ядра Linux. — М. : Вильямс, 2006. — 448 с.
39. *Мюллер, С.* Модернизация и ремонт ПК. — М. : Вильямс, 2007. — 1504 с.
40. Attunity Connect for Mainframe // eAI Journal [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://www.eaijournal.com/attunity-connect-for-mainframe/.
41. *Королев, Л. Н.* Нейрокомпьютинг, нейросети и нейрокомпьютеры [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://podelise.ru/docs/index-614137html.
42. Cloud Computing и Linux [Электронный ресурс]. — Электрон. дан. — Режим доступа: https://www.ibm.com/developerworks/ru/library/l-cloud-computing.
43. *Граничин, О.* Информационные технологии в управлении / О. Граничин, В. Кияев [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://intuit.ru/studies/courses/1055/271/info.

44. *Граничин, О. Н.* Создание гибридных сверхбыстрых компьютеров и системное программирование / О. Н. Граничин, С. Л. Молодцов. — СПб. : Изд-во СПбГУ, 2006. — 108 с.
45. *Сухомлин, В.* Итология — наука об информационных технологиях [Электронный ресурс]. — Электрон. дан. — Режим доступа: http://citforum.ru/programming/prg96/sukhomlin.shtml.

ОГЛАВЛЕНИЕ

Борис Яковлевич СОВЕТОВ,
Владислав Владимирович ЦЕХАНОВСКИЙ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ: ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

Учебное пособие

Зав. редакцией физико-математической
литературы *Н. Р. Крамор*
Ответственный редактор *С. В. Макаров*
Технический редактор *О. О. Николаева*
Подготовка иллюстраций *А. П. Маркова*
Верстка *Л. Е. Голод*
Выпускающие *Е. П. Королькова, Н. А. Крылова*

ЛР № 065466 от 21.10.97
Гигиенический сертификат 78.01.07.953.П.007216.04.10
от 21.04.2010 г., выдан ЦГСЭН в СПб

Издательство «ЛАНЬ»
lan@lanbook.ru; www.lanbook.com;
196105, Санкт-Петербург,пр. Юрия Гагарина, д. 1, лит. А.
Тел./факс: (812) 336-25-09, 412-92-72.
Бесплатный звонок по России: 8-800-700-40-71

Подписано в печать 31.08.2015.
Бумага офсетная. Гарнитура Школьная. Формат 84×108 $^{1}/_{32}$.
Печать офсетная. Усл. п. л. 23,52. Тираж 300 экз.
Заказ № .

Отпечатано в полном соответствии
с качеством предоставленного оригинал-макета
в ПАО «Т8 Издательские технологии».
109316, г. Москва, Волгоградский пр., д. 42, к. 5.