МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

УТВЕРЖДАЮ
Зав. кафедрой АСУ, профессор

А.М. Кориков

СОВРЕМЕННЫЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

Учебное пособие
теория, самостоятельная и индивидуальная работа студента

Учебное пособие
для студентов уровня основной образовательной программы магистратура направления подготовки 01.04.02 «Прикладная математика и информатика» профиля Математическое и программное обеспечение вычислительных комплексов и компьютерных сетей

Разработчик
доцент кафедры АСУ

В.Г. Резник

2016

Резник В.Г.

Современные компьютерные технологии. Теория, самостоятельная и индивидуальная работа студента: Учебное пособие. – Томск, ТУСУР, 2016. – 100 с.

Учебное пособие предназначено для изучения магистрами второго года обучения теоретического материала в формате лекций, а также самостоятельной и индивидуальной работы по дисциплине «Современные компьютерные технологии» на уровне основной образовательной программы магистратура направления подготовки 01.04.02 «Прикладная математика и информатика» профиля «Математическое и программное обеспечение вычислительных комплексов и компьютерных сетей».

Оглавление

Введение

Рассматриваемое учебное пособие содержит теоретический материал, необходимый для изучения дисциплины «Современные компьютерные технологии (СКТ)». Весь теоретический материал опирается на учебный план основной образовательной программы магистратура направления подготовки 010400.68 «Прикладная математика и информатика» профиля «Математическое и программное обеспечение вычислительных комплексов и компьютерных сетей».

Согласно этому плану, студент-магистрант проходит лекционный курс обучения в объеме 18 часов, проводит лабораторные работы в объеме 36 часов, получает прикладные навыки на практических занятиях в объеме 18 часов и завершает обучение по данной дисциплине, выполняя курсовую работу (проект).
Необходимость написания данного учебного пособия обусловлена требованиями Федерального Государственного образовательного стандарта высшего профессионального образования (ФГОС ВПО) третьего поколения, утвержденного Приказом Министерства образования и науки Российской Федерации от 20 мая 2010 г. N 545. Эти требования, при ограниченных временных ресурсах на обучение, ставят необходимым систематизацию большого коли-чества теоретического материала по современным технологиям, который, в основном, содержится в журнальных публикациях или на специализированных сайтах сети Интернет. Если еще учесть требование, что не менее 40% лекционного времени должно отводится на интерактивное общение студентов и преподавателя, то повышенные требования к темпам самостоятельной подготовки студента обосновывают создание учебного пособия, способного обеспечить такую работу.

Поскольку ФГОС ВПО предъявляет высокие требования к практическим навыкам обучающихся, то все учебные работы по дисциплине, включающие лабораторные работы, практические занятия и курсовое проектирование, проводятся на базе «Учебного программного комплекса кафедры АСУ» (УПК АСУ), созданного на основе операционной системы (ОС) Linux дистрибутива Xubuntu. УПК АСУ входит в состав вычислительного комплекса кафедры АСУ, включающего в себя распределенный доступ к общим файловым ресурсам кафедры и вычислительным средствам компьютерного кластера кафедры.

Последовательность и изложение данного учебного материала предполагает, что студент:

- успешно изучает теоретическую часть данного курса;
- успешно осваивает практические навыки работы с вычислительными технологиями УПК кафедры АСУ;
- проводит систематическую предварительную самостоятельную работу по разделам дисциплины;
- завершает отдельные разделы обучения подготовкой письменных отчетов

и защитой курсового проекта на уровне дифференцированного зачета.

Данное учебно пособие содержит девять разделов, включающих рекомендуемую литературу для самостоятельной подготовки, а также вопросы для самостоятельного контроля знаний. Вся методическая последовательность изложения теоретического материала, начиная со второго по девятый разделы, отражают концептуальные взгляды автора на историческую последовательность зарождения и развития современных компьютерных технологий. Не претендуя на строгую научность, подобный субъективный подход позволяет провести осмысленную укрупненную систематизацию самих технологий по мере того, как эти технологии стали активно проявляться в современном социуме.

Первая часть учебного пособия является введением в теоретическую часть данной дисциплины. В нем отражены цели и задачи дисциплины, а также выделены основные наиболее значимые идейные парадигмы, которые реализуются современными компьютерными технологиями.

Второй раздел посвящен вычислительным технологиям, которые являются идейными представлениями «классических технологий» применения средств вычислительной техники.

Третий раздел пособия описывает технологии, которые существенно расширили горизонты возможного применения ЭВМ: хранение информации, как самостоятельного продукта ЭВМ, с возможностью их последующего широкого использования в обществе.

Четвертый раздел дает описание объектно-ориентированных технологий, которые окончательно перевели технологии обработки информации в статус производственных. Информационные объекты стали продуктом, который можно производить, продавать и потреблять.

Пятый раздел дает дает представление об офисных технологиях как базового элемента применения ЭВМ на новом уровне индивидуального и коллективного использования вычислительной техники. Появляется основа для интеграции различных технологий и массовое их применение.

Шестой и седьмой разделы можно изучать в любом порядке. Сложно отделить автоматизацию производства от технологий применения сетей ЭВМ. Выбранная последовательность обоснована тем, что современное производство интегрирует в себе все достижения предыдущих технологий и, одновременно, является заказчиком этих технологий. С другой стороны, технологии взаимодействия открытых систем, из второстепенных идейных и технологических средств по объединению компьютеров, превратились в общую самостоятельную парадигму применения других информационных технологий.

Восьмой раздел содержит описание отдельных наиболее значимых аспектов современных достижений компьютерных технологий и тенденций их развития. На данном уровне объектами производства выступают уже не отдельные информационные объекты, целые информационные технологии.

Наконец, девятый раздел посвящен интеллектуальным системам и технологиям. Хотя работы в этом направлении ведутся достаточно давно, следует признать, что многие достижения здесь ждут своего расцвета, когда появятся сопутствующие технологии, адекватные поставленным задачам.

1 СОСТОЯНИЕ И ТЕНДЕНЦИИ РАЗВИТИЯ СКТ

Целью дисциплины является изложение накопленных знаний по современным компьютерным технологиям, отражающим *достижения науки и техники в области развития вычислительной техники и программного обеспечения*.

Совокупность этих знаний отражена в большом количестве публикаций распределенных по различным журнальным статьям или электронным ресурсам специализированных сайтов Интернет. Тем не менее, у обучающихся должны быть сформированы в данной области знаний как общекультурные, так и профессиональные компетенции. И хотя сами компетенции выражены достаточно абстрактно, их следует знать максимально конкретно.

Общекультурные компетенции (ОК):

- способностью использовать углубленные теоретические и практические знания в области прикладной математики и информатики (ОК-3);
- способностью самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения, в том числе, в новых областях знаний, непосредственно не связанных со сферой деятельности, расширять и углублять свое научное мировоззрение (ОК-4);
- способностью порождать новые идеи и демонстрировать навыки самостоятельной научно-исследовательской работы и работы в научном коллективе (ОК-5);
- способностью совершенствовать и развивать свой интеллектуальный и общекультурный уровень, добиваться нравственного и физического совершенствования своей личности (ОК-6);

Профессиональные компетенции (ПК):

- способностью разрабатывать концептуальные и теоретические модели решаемых научных проблем и задач (ПК-2);
- способностью углубленного анализа проблем, постановки и обоснования задач научной и проектно-технологической деятельности (ПК-3);
- способностью разрабатывать аналитические обзоры состояния области прикладной математики и информационных технологий по профильной направленности ООП магистратуры (ПК-10).

Обобщая основную цель обучения данного курса, можно четко сформулировать, что в результате изучения дисциплины обучающийся должен:

- *Знать основные парадигмы* обработки информации, формирующие современные компьютерные технологии; историческое развитие концепций обработки информации; примеры конкретных систем, демонстрирующих последние достижения в области компьютерных технологий.
- *Уметь использовать* современные интегрированные офисные технологии и системы разработки программного обеспечения.
- *Владеть инструментальными средствами*, предоставляемыми современными компьютерными системами и комплексами.

Исходя из этих целей, можно определить, что задачей дисциплины является *формирование у обучающихся теоретических представлений* о парадигмах и технологиях использования вычислительной техники в современном обществе, а также приобретение ими навыков использования современных технологий на практике.
Решение этой задачи достигается комплексным применением следующих подходов:

- *изучением* теоретической части дисциплины в объеме **18 часов**;
- *выполнением* лабораторных работ в объеме **36 часов**;
- *проведением* практических занятий в объеме **18 часов**;
- *индивидуальной разработкой* курсового проекта (работы) в пределах **8 часов** аудиторных занятий.

Теоретическая часть данного раздела излагает следующие вопросы:

- *Идейные парадигмы* компьютерных технологий в их органической связи с технологическими достижениями в плане развития аппаратных средств ЭВМ.
- *Идеи многоуровневой организации* вычислительных систем в проекции задач автоматизации и автоматического управления.
- *Идеи построения распределенных систем* обработки данных совмещенных с идеями «виртуализации».

1.1 Идейные парадигмы развития аппаратных средств компьютеров

Большое количество компьютерных технологий, которые нам демонстрируют реклама и Интернет, скрывают и искажают как научную, так и практическую значимость современных технологических достижений в области применения средств вычислительной (цифровой) техники.
Следует различать:

- *идейную часть технологий*, которая является основой для формирования парадигм развития технологий на определенный исторический период времени;
- *современные технологии*, которые определяют последние достижения промышленности или являются уже неотъемлемой частью современного социума.

Первая, идейная часть, создается как отражение кризисного состояния текущего момента развития технологий? с целью создания *перспективного вектора движения* к новым достижениям.
Вторая, реализованная часть технологий, служит мерой технологических достижений социума и базой для его дальнейшего развития.

Чтобы адекватно ориентироваться во всем многообразии современных компьютерных технологий, сначала *ограничимся рассмотрением только цифровых вычислительных систем*, а затем выберем критерии классификации, по которым будем сравнивать и упорядочивать отдельные технологии. Количество таких критериев должно быть небольшим и они должны быть наиболее значимыми в текущий момент времени.
Чтобы не впадать в крайности и не вызывать справедливый огонь критики, в качестве критериев выберем некоторые аспекты тенденций, присущих текущему развитию технологий. Эти аспекты следующие:

1. *Исторический аспект развития* компьютерных технологий (КТ), отражающий их временную компоненту развития;
2. *Модульный аспект создания* КТ, отражающий общие закономерности создания сложных систем;
3. *Централизующий аспект тенденций модификации* КТ, отражающий *диалектическую противоположность модульному аспекту*, в плане более длительного временного цикла развития.

Исторический аспект развития КТ отражает изменение во времени идейных парадигм этих технологий.
Очевидно, что новые парадигмы возникали в моменты кризисных ситуаций, требующих существенного изменения теоретических представлений на подходы к созданию технических и программных средств цифровой вычислительной техники.
С другой стороны, известные технологические революции в создании аппаратных средств компьютеров, известные как переход от релейной и ламповой технологий к полупроводниковой элементной базе, а затем к микросхемам, сами по себе обеспечили появление и реализацию на практике новых парадигм. Такое положение дел привело к тому, что новые технологии стали порождать другие технологии. В результате компьютерные технологии стали менять качество, становиться популярными и проникать во все сферы социума.

Рассматривая развитие КТ в историческом аспекте, можно легко выделить последовательность следующих технологических достижений [1 - 9]:

- *Вычислительные технологии*, — когда компьютер рассматривался как мощный калькулятор, способный обеспечить решение многих расчетных задач.
- *Технологии хранения информации*, — когда потребности работы, в первую очередь со сложными экономическими моделями, потребовали технологии проектирования предметной области данных для вычислений, а затем и создание технологий управления этими данными (СУБД).
- *Объектно-ориентированные технологии* возникли как закономерное изменение концептуальной основы программирования при создании все более сложных программных систем.
- *Технологии автоматизированного управления* стали естественным интег-

ратором предшествующих технологий, которые обеспечили промышленное производство управленческим аспектом КТ и обеспечили себе экономическую основу развития.

- *Технологии взаимодействия открытых систем*, после ряда существенных модификаций, заняли самостоятельное место как базовая основа современных технологических решений вычислительных проектов.
- *Офисные технологии* интересны тем, что они обеспечили массовое внедрение персональных КТ.
- *Сервисные технологии* отражают новое качество, демонстрирующее собой массовые общественные КТ, а также новые качественные тенденции применения этих КТ, изменяющие первоначальную тенденцию *разделения вычислительных систем на их объединение* на коммерческой основе.
- *Интеллектуальные системы и технологии* отражают собой современное и будущее компьютеризованное сообщество социума.

Перечисленная выше последовательность технологических достижений достаточно точно отражает временной характер практической актуализации соответствующих идей и принята за основу при изложении теоретического материала данного учебного пособия.
Единственное исключение сделано для *офисных технологий*, описание которых перенесено с шестой позиции на четвертую. Это сделано из методических соображений для более полного согласования теоретического материала и практических работ по данному курсу.

Модульный аспект создания компьютерных технологий отражает *общие тенденции* изменений, присущие развитию сложных систем.
Поскольку развитие любой системы, связанное с увеличением числа ее элементов, порождает как минимум **n!** *бинарных отношений*, то увеличение числа этих отношений создает проблемы, которые разрешаются двумя основными способами:

1. *потребности управления* одних элементов другими, приводит к *вертикальной декомпозиции системы на уровни подчинения*;
2. *потребности специализации внутри уровня*, приводят к *выделению подсистем по модальностям некоторого общего признака*.

Таким образом, данный аспект является крайне важным при анализе всех КТ.

Централизующий аспект тенденций модификации КТ является важнейшим фактором современности. Его значение обусловлено двумя причинами:

- *общественная значимость* и *сложность сопровождения* технологических систем, которая не может обслуживаться малыми организационными структурами;
- *стремление корпоративных и государственных организаций* монополизировать ведущие технологии с целью максимального извлечения экономической или политической выгоды.

Замечание

Подобные тенденции обязательно должны учитываться при анализе и последующем использовании соответствующих КТ.

1.2 Многоуровневые модели управления

С тех пор, как *академик В.М. Глушков* показал, что в любом устройстве обработки информации можно выделить *операционный* и *управляющий автоматы*, соединенные каналами обратной связи, идеи модульного построения ЭВМ стали активно внедряться в практические реализации.

В соответствии с принципом модульности, ЭВМ стала представляться *иерархической многоуровневой системой*, состоящей из множества *виртуальных машин* со своим языком программирования.
Классическая архитектура такой ЭВМ, *из 6 уровней*, показана на рис. 1.1.

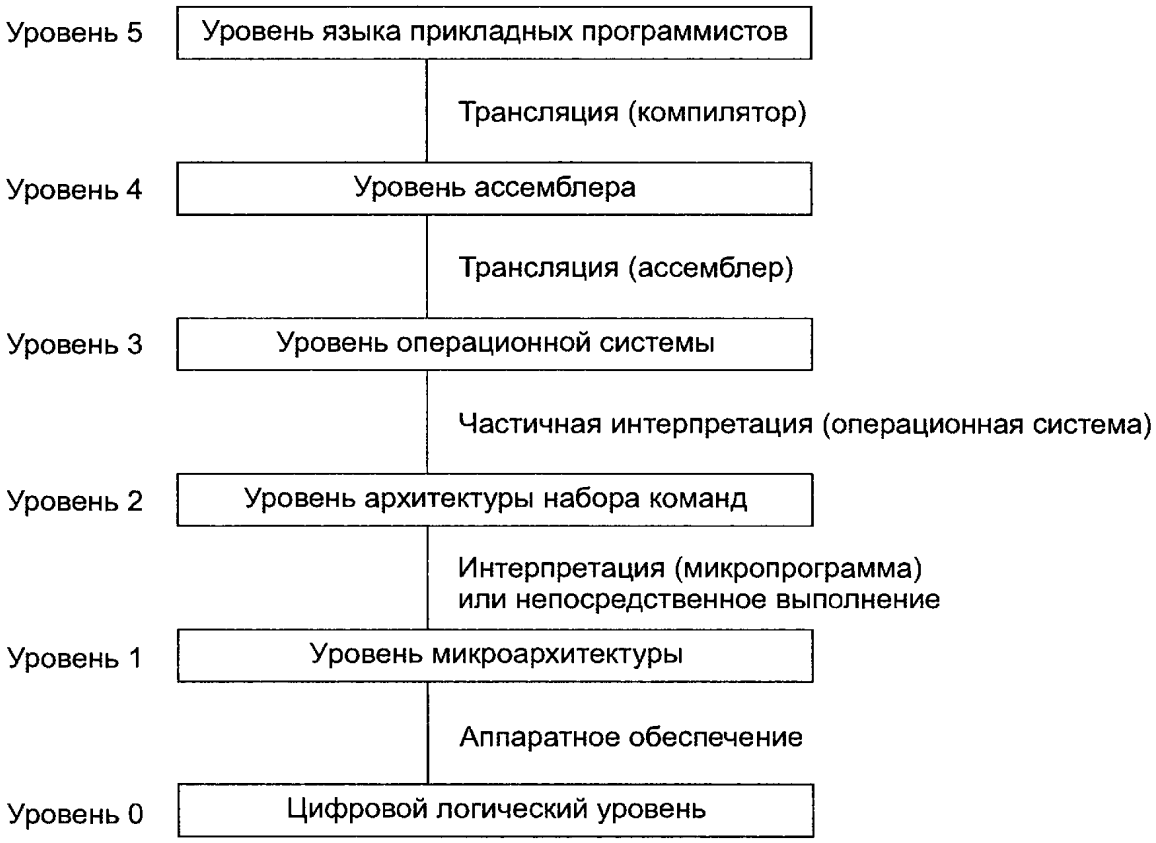

Рисунок 1.1 - Классическая многоуровневая архитектура ЭВМ

Здесь, *уровень 0* – это аппаратное обеспечение машины.
Это – *цифровой логический уровень*, объектами которого являются ***вентили***.
Сами вентили состоят из аналоговых компонентов, таких как транзисторы, но они могут быть точно смоделированы как цифровые устройства.
Каждый такой вентиль:

- *имеет один или несколько цифровых входов* (сигналов), представляющих 0 или 1;
- *вычисляет простые функции* таких сигналов как «И» или «ИЛИ».

Несколько вентилей формируют ***1 бит памяти***, который может содержать 0 или 1.
Биты памяти, объединенные в группы, например, 8, 16, 32 или 64, формируют *регистры*.
Каждый регистр может содержать одно двоичное число до определенного предела.
Из вентилей также может состоять сам компьютер.

В дальнейшем, многоуровневая организация стала широко применяться и в других технологиях. Например, хорошо известна *7-уровневая модель взаимодействия открытых систем (ВОС)*, ставшая международным стандартом для архитектуры сетевых технологий.
Другой пример, - иерархическая модель АСУ из 3-х уровней, содержащая:

- *верхний уровень* — АСУП — АСУ предприятия;
- *средний уровень* — АСУПП — АСУ производственными процессами;
- *нижний уровень* — АСУТП — АСУ технологическими процессами.

1.3 Распределенные системы и идеи «виртуализации»

Первые технологии, *ориентированные на парадигму вычислений*, потребовали создание теоретических моделей, которые бы адекватно отражали существующие представления о возможностях применения ЭВМ. Такой моделью стала архитектура систем обработки данных (СОД), показанная на рис. 1.2.

Характерной чертой модели СОД является разделение компьютерных архитектур на:

- *сосредоточенные системы*, когда связь между ЭВМ рассматривается как часть самой компьютерной архитектуры;
- *распределенные системы*, в которых отдельные ЭВМ или их части связаны средствами телекоммуникаций или сети.

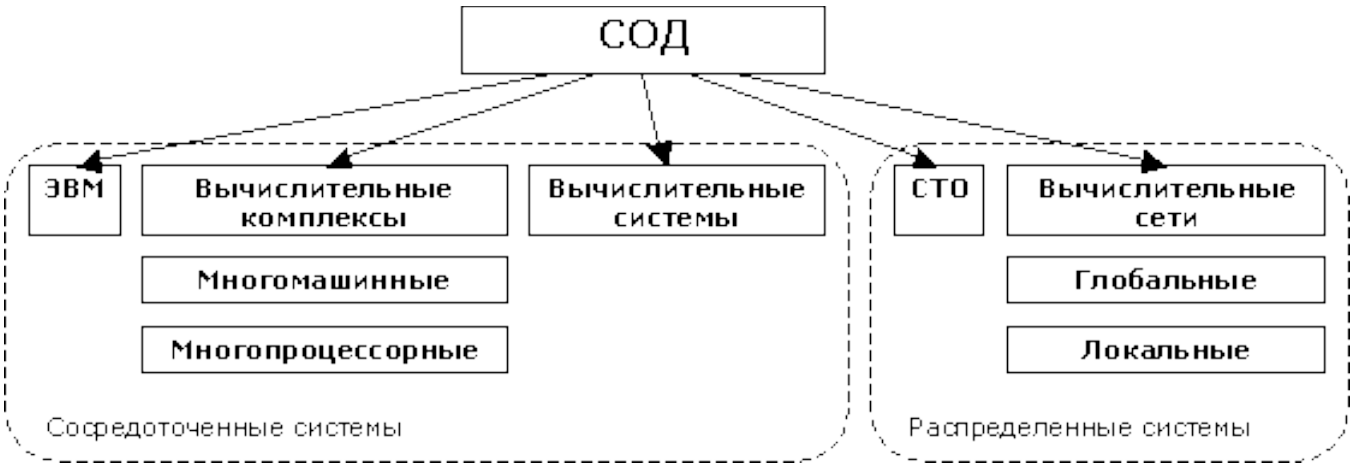

Рисунок 1.2 - Структура систем обработки данных

Со временем, технические и программные средства развились настолько, что работа в сети перестала вызывать какие-либо технологические затруднения. Это сильно снизило «проблемность» создания распределенных систем и сделало актуальными проблемы другого качества. Одной из таких проблем является идея «***виртуализации***», которая предполагает, что доступ к вычислительным ресурсам в некоторой достаточно общей вычислительной системе осуществляется через заранее известные ***виртуальные (мнимые) адреса***, реально не соответствующие конкретным ***физическим адресам***.

Примером реализации такой идеи является технология «***облачных вычислений***», которая интенсивно рекламируется группой разработчиков при активном участии корпорации Microsoft.

1.4 Рекомендуемая литература для самостоятельной подготовки

1. Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем: Учебник для вузов. – СПб.: Питер, 2006. - 712с. (46 экз.)
2. Бройдо В.Л. Вычислительные системы, сети и телекоммуникации: Учебное пособие для вузов. – СПб.: Питер, 2006. - 702с. (30 экз.)
3. Таненбаум Э. Архитектура компьютера. – СПб.: Питер, 2007. - 843с. (1 экз.)
4. Гордеев А.В. Операционные системы: учебное пособие для вузов. ─ СПб.: Питер, 2004. ─ 415с. (17 экз.)
5. Гордеев А.В. Системное программное обеспечение: учебное пособие для вузов. ─ СПб.: Питер, 2001. ─ 736с. (43 экз.)
6. Бойченко И.В. Программное обеспечение сетей ЭВМ. - Томск: ТМЦДО, 2005. - 294 с. (20 экз.)
7. Достоверный и правдоподобный вывод в интеллектуальных системах: учебное пособие для вузов / В. Н. Вагин, Е. Ю. Головина, А. А. Загорянская, М. В. Фомина; Ред. Д. А. Поспелов. - М. : Физматлит, 2004. - 704 с. (101 экз.)
8. Антамошин А. Н. Интеллектуальные системы управления организационно-техническими системами. - М. : Горячая линия-Телеком, 2006. - 160 с. (70 экз.)
9. Андрейчиков А.В. Интеллектуальные информационные системы: Учебник для вузов. - М.: Финансы и статистика, 2006. - 423с. (20 экз.)

1.5 Вопросы для самостоятельного контроля знаний

1. Каковы идейные парадигмы, определяющие историческое развитие компьютерных технологий?
2. Каким образом проявляется технологическая революция развития аппаратных средств цифровой вычислительной техники?
3. Чем вызваны идеи многоуровневой организации компьютерных технологий?
4. В чем заключается различие подходов автоматического и автоматизированного управлений?
5. Каковы тенденции применения идеи распределенных систем?
6. В чем суть идеи «виртуализации»?

2 ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ

Вычислительные технологии возникли и остаются тем концептуальным набором достижений современного социума, который неразрывно связывает себя с цифровой вычислительной техникой, именуемой как *электронные вычислительные машины* (ЭВМ).

Чтобы раскрыть этот набор концепций, в данном разделе учебного пособия рассматриваются следующие вопросы:

- *Парадигма* «программа-массив».
- *Компьютер* как вычислитель.
- *ОС* и системы разработки программного обеспечения.
- *Технологии расчетов* и моделирования.
- *Интегрированные системы* научных и инженерных исследований.
- *Системы* Mathematica, Maple, Mathcad, MATLAB, Simulink.

Все вычислительные технологии неразрывно связаны с технологическим ходом развития вычислительной техники, в котором традиционно выделяется соответствующая историческая последовательность этапов.

Особый интерес в этой последовательности представляют первые три этапа, обеспечившие формирование рассматриваемых концепций:

- *Нулевое поколение* (1492-1945) — механическая эра, включая электромеханические устройства вычислений;
- *Первое поколение* (1937,1945 -1953,1955) — электронные лампы, первые ЭВМ;
- *Второе поколение* (1954,1955 - 1962,1965) — полупроводники и транзисторы.

Соответствующие технологические достижения в развитии аппаратных средств цифровой вычислительной техники (ВТ) с полным основанием позволили говорить о специализированных объектах для вычислений — *компьютерах*.

Стали формироваться образы, которые в виде тезисов провозглашали: «*Компьютер — это супер дорогие счеты*».

В более позднее время, с появлением микросхем, стали говорить «*Компьютер — это большой калькулятор*».

Такая направленность мышления вызвана основными потребностями социума в сложных и затратных вычислениях:

- *для научных исследований*, включая астрономию, географию, исследование космоса и другие;
- *для военных целей*, включая создание новых видов вооружений и широкомасштабного применения этих вооружений;
- *для сложных* экономических расчетов;
- *для создания, развития и практического применения* «ядерных технологий».

Общество накопило множество вычислительных задач и создало математический аппарат для их решения.
Вычислительные машины стали обеспечивать технологическую основу для их решения.
Задачи стали, с тем или иным переменным успехом, решаться, повышая свою социальную значимость.
Все это и определило *основную социальную направленность* использования вычислительной техники.
Имеющийся социальный заказ стимулировал развитие специализированных областей математики, которые стали назваться как - *вычислительная математика*.
Соответственно, возникла потребность в математике, которая бы обеспечивала расчеты, технологически отличные от методов расчетов на базе *вычислителей нулевого поколения*.

Наконец, социум стал формировать профессии, которые связаны исключительно с трудовой деятельностью, ориентированной на применение компьютеров.
Соответствующая профессиональная деятельность стала *формировать мышление, создавать теории и реализовывать на практике продукцию*, которая применима только для собственных профессиональных нужд этих технологий.

Все, сказанное выше, позволяет говорить об отдельном технологическом направлении использования ВТ — о *вычислительных технологиях*.

2.1 Парадигма вычислительной технологии

В рамках упомянутого выше социального процесса, сформировалась профессия *программист*.
Программист стал элитой, которая была призвана *сформировать ту профессиональную прослойку общества*, что обеспечит потребности социума в плане решения вычислительных задач.
Как элита, формируемая из ученых и научно-технических специалистов высокой квалификации, *программисты стали формировать теоретические и технологические парадигмы*, адекватные поколению имеющихся аппаратных средств ВТ.

Первой такой парадигмой стала концепция «*Программа-массив*». Она была основной концепцией, вплоть до середины 60-х годов XX-го века.
Суть этой технологической парадигмы в том, что технология использования ЭВМ разбивается на три основных этапа:

- *подготовка и ввод* в ЭВМ исходных данных;
- *обработка исходных данных* программной или программным модулем;
- *вывод результатов вычислений* на внешние носители (хранители) инфор-

мации ЭВМ и последующая интерпретация результатов работы программы.

Технологическое расширение этой парадигмы можно представить как множество программ, *обеспечивающих решение сложной задачи*, целостной по своей целевой значимости.

Основные технологические характеристики парадигмы «программа-массив»:

- *программы «плоские»*, а данные и код находятся в одном адресном пространстве и не разделены между собой;
- *программные модули (программы) - слабо связаны* и только *через данные и процесс запуска отдельных программ*;
- *значимость программы* оценивается количеством операторов: *кода программы*;
- *направленность алгоритмов* — исключительно расчетная;
- *насыщенность программ деталями*, далекими от прикладной направленности алгоритма.

Очевидно, что *недостатки, присущие парадигме «программа-массив»*, стали очень быстро сказываться при усложнении программных систем. Это потребовало некоторого пересмотра классического подхода к парадигме и внесение ряда технологических изменений в пределах существующих потребностей применения ЭВМ.

Возникло стремление:

- *уменьшить излишнюю информационную загруженность* программ;
- *перейти от машинного кода к ассемблеру*, *макросам*, *мнемоникам*;
- *учесть специфическое не машинное восприятие человека* при написании и анализе программ;
- *обеспечить уменьшение объема* исходного кода программ;
- *повысить надежность и функциональную наполненность* разрабатываемого программного обеспечения (ПО).

Основным технологическим подходом, сохранившемся до настоящего времени, стало *разделение программ по специализации*:

- системное ПО;
- прикладное ПО;
- инструментальные средства разработки ПО.

Замечание

Если системное ПО было объектом внимания достаточно узкого круга системных программистов, то прикладное ПО и инструментальные средства разработки, получившие аббревиатуру **IDE**, стали достоянием широких масс специалистов, применяющих технологические достижения ВТ.

Именно стремительное развитие IDE - *Integrated Development Environment* и, связанных с ними языков программирования, обеспечили разработку вычислительных систем *моделирования, инженерных и научных расчетов*.

Со временем, системное ПО, являясь, по сути, вспомогательным, стало оказывать все большее и большее влияние на современные вычислительные технологии. Это связано с широким многообразием архитектур современных средств ВТ.

Замечание

Системное ПО, во многом, является закрытым и непонятным для прикладного программиста. *Так было всегда, и так будет в дальнейшем*. А для аргументации этого заявления, нужно учесть, что системное ПО является:

- *оторванным* от прикладного алгоритмического характера приложений;
- *более абстрактным* по функциональному назначению, по сравнению с конкретными приложениями;
- *самостоятельными технологическими объектами*, порождающими новые технологии.

Основные собственные проблемы системного ПО, которые активно проецируются на прикладное ПО, это:

- *проблемы модульности систем*;
- *проблемы именования ресурсов*;
- *проблемы доступа к ресурсам.*

Далее, перейдем к рассмотрению ряда практических реализаций вычислительных технологий, являющихся представителями двух актуализированных направлений:

- *вычислительным расчетам* и моделированию;
- *интегрированным системам* научных и инженерных расчетов.

2.2 Технологии расчетов и моделирования

Классический вариант парадигмы «программа-массив» очень быстро зашел в тупик, по причине сложности отношений между отдельными программами, а также из-за отсутствия согласованности этих программ по форматам представления данных.

Чтобы решить эту проблему:

- *стали создаваться языки*, более высокого уровня чем ассемблер, с развитыми средствами типизации данных;
- *стали создаваться библиотеки программ*, позволяющие объединять и хранить, для последующего использования, наиболее общие и полезные программные модули.

Со временем, эти подходы тоже исчерпали себя по причинам:

- *несогласованности интерфейсов* и различное качество библиотек от различных поставщиков ПО;

- *неспособность* или коммерческая нецелесообразность создания и сопровождения библиотек ПО для множества меняющихся архитектур ВТ.

Тем не менее, потребности в поддержке *сложных математически расчетов* и *компьютерного моделирования* постоянно возрастали, что привело к созданию множества проектов, реализованных на коммерческой основе. Ряд таких проектов *просуществовали до настоящего времени* и продолжают успешно использоваться как отдельные системы.
Далее, мы рассмотрим два таких проекта: систему *Mathematica* и систему *Maple*.

2.2.1 Система Mathematica

Mathematica — позицирует себя как система компьютерной алгебры, созданная компанией Wolfram Research. [1, 2].

Появившаяся в 1988 году, эта система содержит множество функций как для численных расчетов, так и для аналитических преобразований. Кроме того, эта система поддерживает:

- *работу с графикой и звуком*, включая построение двух- и трехмерных графиков функций;
- *рисование* произвольных геометрических фигур;
- *экспорт и импорт* изображений и звука.

Чтобы более наглядно представить потенциал этой системы, рассмотрим список ее возможностей по отдельным направлениям.

Аналитические преобразования

- Решение систем полиномиальных и тригонометрических уравнений, неравенств, а также трансцендентных уравнений, сводящихся к ним.
- Решение рекуррентных уравнений.
- Упрощение выражения.
- Нахождение пределов.
- Интегрирование и дифференцирование функций.
- Нахождение конечных и бесконечных сумм и произведений.
- Решение дифференциальных уравнений и уравнений в частных производных.
- Преобразования Фурье, Лапласа и Z-преобразование.
- Преобразование функции в ряд Тейлора, операции с рядами Тейлора: сложение, умножение, композиция, получение обратной функции и другие.
- Вейвлет-анализ.

Численные расчеты

- Вычисление значений обычных и специальных функций с произвольной точностью.
- Решение систем уравнений.
- Нахождение пределов.
- Интегрирование и дифференцирование.
- Нахождение сумм и произведений.
- Решение дифференциальных уравнений и уравнений в частных производных.
- Полиномиальная интерполяция функции от произвольного числа аргументов по набору известных значений.
- Преобразования Фурье и Лапласа, а также Z-преобразование.

Теория чисел

- Определение простого числа по его порядковому номеру, определение количества простых чисел, не превосходящих данное.
- Дискретное преобразование Фурье.
- Разложение числа на простые множители, нахождение наибольшего общего делителя (НОД) и наименьшего общего кратного (НОК).

Линейная алгебра

- Операции с матрицами: сложение, умножение, нахождение обратной матрицы, умножение на вектор, вычисление экспоненты, получение определителя.
- Поиск собственных значений и собственных векторов.

Графика и звук

- Построение графиков функций, параметрических кривых и поверхностей.
- Построение геометрических фигур: ломаных, кругов, прямоугольников и других.
- Воспроизведение звука, график которого задается аналитической функцией или набором точек.
- Импорт и экспорт звука и графики, во многих растровых форматах.
- Построение и манипулирование графами.

Разработка программного обеспечения

- Автоматическое генерирование C кода и его компоновка.
- Автоматическое преобразование компилируемых программ системы Mathematica в C код для автономного или интегрированного использования.
- Использование SymbolicC для создания, обработки и оптимизации C кода.
- Интеграция внешних динамических библиотек.
- Поддержка CUDA и OpenCL.

Таким образом, система Mathematica обеспечивает не только достаточно полный набор математических преобразований, но и включает в себя поддержку многих вспомогательных технологий, таких как:

- работа с графикой и звуком;
- реализация и оптимизация языков программирования;
- использование для вычислений различных аппаратных специализированных цифровых вычислителей.

Чтобы полностью раскрыть возможности этой системы, рассмотрим подробнее упомянутые технологии.

Технология CUDA или ***Compute Unified Device Architecture*** [3].
Это - программно-аппаратная архитектура, позволяющая производить вычисления с использованием графических процессоров NVIDIA, поддерживающих технологию GPGPU.
Сама технология GPGPU позволяет производить произвольные вычисления на видеокартах, начиная с чипа NVIDIA восьмого поколения — G80.
Эта технология присутствует во всех последующих сериях графических чипов, которые используются в семействах ускорителей GeForce, Quadro и NVidia Tesla.

Создана система разработки ***CUDA SDK***, которая позволяет программистам писать программы на специальном упрощенном диалекте языка C, включать специальные функции в текст программ на C, дает возможность организовывать доступ к набору инструкций графического ускорителя, управлять его памятью, организовывать на нем сложные параллельные вычисления.

Технология OpenCL или ***Open Computing Language*** [4].
Это - фреймворк для написания программ, связанных с параллельными вычислениями на различных графических процессорах (*GPU*) и центральных процессорах (*CPU*).
В него входят язык программирования, который базируется на стандарте C99, и API.
OpenCL является полностью открытым стандартом, реализующим *технологию GPGPU*.

Технология GPGPU или ***General-purpose graphics processing units***: «GPU общего назначения» [5].
Это - техника применения программируемых шейдерных (схем затенения) блоков и растровых конвейеров, что позволяет разработчикам ПО использовать потоковые процессоры для не графических данных.

Технология OpenGL или ***Open Graphics Library*** [6].
Это - открытая графическая библиотека и графическая API-спецификация, определяющая независимый от языка программирования и платформы, программный интерфейс для написания приложений, которые используют двумерную и трёхмерную компьютерную графику.
Включает более 250 функций для рисования сложных трёхмерных сцен из простых примитивов.
Используется при создании компьютерных игр, САПР, виртуальной реальности и визуализации в научных исследованиях.

На платформе MS Windows конкурирует с технологией Direct3D.

OpenAL или ***Open Audio Library***. [7].
Это - свободно распространяемый межплатформенный API для работы с аудиоданными.
Его особенностью является работа со звуком в 3D пространстве и использование эффектов EAX.
Поддерживается компанией Creative.

Дополнительно, *к заимствованным внешним языкам прогаммирования*, система Mathematica использует собственный интерпретируемый язык функционального программирования с одноименным названием.
Язык Mathematica допускает, так называемые, отложенные вычисления, используя оператор определения «`:=`».
Можно даже сказать, что система Mathematica написана на самом языке Mathematica.

Рассмотрим на конкретных примерах некоторые особенности этого языка.

Функциональное программирование

Пример кода, который последовательно 5 раз применяет функцию `f`:

```
NestList[f, x, 5]
```

Результатом будет список из аргумента `x`, одного, двух, трёх и так далее до пяти применений функции `f` к этому аргументу, всего 6 элементов списка:

```
{x, f[x], f[f[x]], f[f[f[x]]], f[f[f[f[x]]]], f[f[f[f[f[x]]]]]}
```

Пример 3-х кратного применения конкретной функции `Sin[x + 1]` имеет вид:

```
NestList[Sin[# + 1] &, x, 3]
```

Результатом будет:

```
{x, Sin[1 + x], Sin[1 + Sin[1 + x]], Sin[1 + Sin[1 + Sin[1 + x]]]}
```

Процедурное программирование

Mathematica также поддерживает процедурный стиль программирования:

```
For[i = 1; t = x, i^2 < 10, i++,
t = t^2 + i;
Print[t]]
```

Здесь, точка с запятой отделяет различные последовательные команды. Для осуществления стандартных циклов есть функции ***Do***, ***While***, ***For***.
Условные выражения осуществляются посредством функций ***If***, ***Which***, ***Switch***.

Таким образом, будучи изначально функциональным языком программирование Mathematica, тем не менее, имеет функции, которые позволяют писать код очень близкий к стандартным процедурным языкам программирования.

Программирование посредством задания правил

В системе Mathematica также можно задавать правила работы с теми или иными выражениями.
Таким образом, можно определять объекты подобно тому как это обычно делается в математике, задавая их свойства посредством правил:

```
f[x_ + y_] := f[x] + f[y];
f[a + b + c]
```

Результат:

```
f[a] + f[b] + f[c]
```

Объектно-ориентированное программирование.....

Mathematica позволяет явно задавать определения, связанные с определенным объектом, реализуя, тем самым, возможность *использования объектно-ориентированного стиля программирования*:

```
h /: f[h[x_], x_] := hf[x];
h /: p_[h[x_]] := ph[f, x];
h /: h[x_] + h[y_] := hsum[x, y];
```

Вычисляя с данными определениями:

```
h[a] + h[b] + p[h[r]] + h[h[x]]
```

результатом будет:

```
hsum[a, b] + ph[f, r] + ph[f, x]
```

2.2.2 Система Maple

Другим примером систем расчетов и моделирования, которую мы кратко рассмотрим, является ***Maple*** [8, 9].

Maple - программный пакет системы компьютерной алгебры, который *с 1984 года* выпускается компанией Waterloo Maple Inc.

Система Maple предназначена для символьных вычислений, хотя имеет ряд средств и для численного решения дифференциальных уравнений и нахождения интегралов.
Она обладает развитыми программными графическими средствами и имеет собственный язык программирования, напоминающий ООП Паскаль.
Последняя, зафиксированная мной версия Maple 16, выпущена 28 марта 2012 года.
Сама система реализуется через подразделение ***Maplesoft***, которое продает как студенческую, так и профессиональные версии ***Maple***, с существенной разницей в цене: 99 $ и 1995 $, соответственно.
Недавние студенческие версии, начиная с шестой, не имели вычислительных ограничений, но поставлялись с меньшим объемом печатной документации.

Следует отметить, что так же различаются студенческая и профессиональная версии пакета Mathematica.

2.3 Интегрированные системы научных и инженерных расчетов

Несмотря на большие возможности систем ***Mathematica*** и ***Maple***, всегда была и возрастает потребность в интегрированных пакетах, которые бы обеспечивали инженерные и научные расчеты с минимальными затратами на программирование.
Далее, мы рассмотрим три таких системы: ***Mathcad***, ***MATLAB*** и ***Simulink***.

2.3.1 Система Mathcad

Mathcad - система компьютерной алгебры, которая относится к классу систем *автоматизированного проектирования*. Ориентированна на подготовку интерактивных документов с вычислениями и визуальным сопровождением. Отличается легкостью использования и применения для коллективной работы. Имеет простой и интуитивный интерфейс пользователя. Для ввода формул и данных можно задействовать как клавиатуру, так и специальные панели инструментов.

Первоначально, Mathcad был задуман и написан в Массачусетском технологическом институте (MIT) Алленом Разводом [10].
Первая версия системы появилась в 1986 году.
Сейчас Mathcad выпускается корпораций PTC (Parametric Technology Corporation) [11], которая в 2010 году выпустила версию 15.0 этой системы.

На примере системы Mathcad хорошо видно как технологически развивалась парадигма «программа-массив».
От примитивного взаимодействия между программами через результаты отдельных вычислений, сохраняемых в собственных уникальных форматах, идет унификация программных модулей в специализированные библиотеки.
Затем, наблюдается переход к системам, связанным единой прикладной направленностью.
Далее, эти системы начинают ориентироваться отдельные технологические достижения, которые напрямую не связаны с прикладным назначением этих систем.
Наблюдается интеграция и включение в себя отдельных технологий, которые ассоциируются современными достижениями компьютерных технологий.

Версии Mathcad, до 13.1 включительно, основаны на подмножестве системы компьютерной алгебры Maple (*MKM, Maple Kernel Mathsoft*) [8].

Начиная с 14 версии, используется символьное ядро MuPAD [12].
Работа осуществляется в пределах рабочего листа, на котором ***уравнения*** и ***выражения*** отображаются графически, в противовес текстовой записи в языках программирования.
При создании документов-приложений используется принцип WISIWYG, - ***What You See Is What You Get***: «что видишь, то и получаешь»).

Публично, Mathcad позицируется как программа ориентированая на пользователей ***непрограммистов***, но Mathcad также используется в сложных проектах, позволяющих ***визуализировать*** результаты математического моделирования.
Для этого используются ***распределенные вычисления*** и ***традиционные языки программирования***.
Также Mathcad используется в крупных инженерных проектах, где значение имеет ***трассируемость*** и ***соответствие стандартам***.

С другой стороны, Mathcad удобно использовать для обучения, вычислений и инженерных расчетов. Открытая архитектура приложения сочетается с технологиями **.NET** и **XML**.
Это позволяют легко интегрировать Mathcad во многие ИТ-структуры и инженерные приложения.
Есть возможность создания электронных книг ***e-Book***.
Количество пользователей в мире оценивается около 1.8 млн.

Mathcad содержит множество операторов и встроенных функций для решения различных технических задач.
Программа позволяет выполнять численные и символьные вычисления, производить операции со скалярными величинами, векторами и матрицами, автоматически переводить одни единицы измерения в другие.
Среди возможностей Mathcad можно выделить:

- Решение дифференциальных уравнений, в том числе и численными методами.
- Построение двумерных и трёхмерных графиков функций в разных системах координат: контурные, векторные и другие.
- Использование греческого алфавита как в уравнениях, так и в тексте.
- Выполнение вычислений в символьном режиме.
- Выполнение операций с векторами и матрицами.
- Символьное решение систем уравнений.
- Аппроксимация кривых.
- Выполнение подпрограмм.
- Поиск корней многочленов и функций.
- Проведение статистических расчетов и работа с распределением вероятностей.
- Поиск собственных чисел и векторов.
- Вычисления с единицами измерения.
- Интеграция с системами САПР и использование результатов вычислений в качестве управляющих параметров.

С помощью Mathcad инженеры могут документировать все вычисления в процессе их проведения.

С другой стороны, версии Mathcad могут отличатся комплектацией и лицензией пользователя.
В разное время поставлялись версии **Mathcad Professional**, **Mathcad Premium**, **Mathcad Enterprise Edition**, отличаются комплектацией.
Для академических пользователей предназначена версия **Mathcad Academic Professor**, которая обладает полной функциональностью, но отличается лицензией пользователя и имеет соответственно меньшую стоимость.
Было также время, когда выпускались упрощенные и заметно «урезанные» студенческие версии этой системы.

Дальнейшее развитие технология Mathcad получила при создании ***Mathcad Application Server*** (MAS). Суть ***технологии MAS*** - в реализации удаленного доступа к программному обеспечению Mathcad или к уже готовым ***Mathcad-документам*** через ***WWW-интерфейс***: технология **Web Calc**.
Теперь пользователь MAS не нуждается в покупке Mathcad. Ему не требуется скачивать и запускать ***exe-файлы***, хотя это не исключается и определяется уровнем доступа.

2.3.2 Система MATLAB

MATLAB или «*Matrix* Laboratory» [13, 14].
Это - пакет прикладных программ для решения задач технических вычислений и одноименный язык программирования, используемый в этом пакете.
Считается, что **MATLAB** используют более 1000000 инженерных и научных работников. Он работает на большинстве современных операционных систем, включая Linux, Mac OS, Solaris и MS Windows.

MATLAB, как язык программирования, был разработан **Кливером Моулером** (*Cleve Moler*) в конце 70-х годов.
Целью разработки служила задача дать университетским студентам возможность использования программных библиотек Linpack и EISPACK без необходимости изучения ***языка Fortran***.
Поскольку, новый язык быстро распространился среди других университетов и был с интересом воспринят учеными, работающими в области прикладной математики, то он был переписан на языке C, а в 1984 году была основана компания MathWorks для дальнейшего развития этого пакета [14].

Первоначально, MATLAB предназначался *для проектирования систем управления*, но быстро завоевал популярность во многих других научных и инженерных областях.
Он стал широко использоваться и в образовании для преподавания линейной алгебры и численных методов.

MATLAB является высокоуровневым интерпретируемым языком программирования, включающим основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования.

Программы, написанные на MATLAB, бывают двух типов:

- Функции;
- Скрипты.

Функции имеют входные и выходные аргументы, а также собственное рабочее пространство для хранения промежуточных результатов вычислений и переменных.
Скрипты же используют общее рабочее пространство.
Как скрипты, так и функции не компилируются в машинный код и сохраняются в виде текстовых файлов. Существует также возможность сохранять так называемые *pre-parsed* программы: функции и скрипты, обработанные в вид, удобный для машинного исполнения.
В общем случае, такие программы выполняются быстрее обычных, особенно если функция содержит команды построения графиков.

Основной особенностью языка MATLAB являются его широкие возможности по работе с матрицами, которые выражены лозунгом: «думай векторно» (*Think vectorized*).
MATLAB предоставляет пользователю большое количество функций для анализа данных, которые покрывают практически все области математики, в частности:

- Матрицы и линейная алгебра — алгебра матриц, линейные уравнения, собственные значения и вектора, сингулярности, факторизация матриц и другие.
- Многочлены и интерполяция - корни многочленов, операции над многочленами и их дифференцирование, интерполяция и экстраполяция кривых, и другие.
- Математическая статистика и анализ данных - статистические функции, статистическая регрессия, цифровая фильтрация, быстрое преобразование Фурье и другие.
- Обработка данных - набор специальных функций, включая построение графиков, оптимизацию, поиск нулей, численное интегрирование, в квадратурах, и другие.
- Дифференциальные уравнения - решение дифференциальных и дифференциально- алгебраических уравнений, дифференциальных уравнений с запаздыванием, уравнений с ограничениями, уравнений в частных производных и другие.
- Разреженные матрицы - специальный класс данных пакета MATLAB, использующийся в специализированных приложениях.
- Целочисленная арифметика - выполнение операций целочисленной арифметики в среде MATLAB.

Следует отметить, что пакет MATLAB включает интерфейс взаимодействия с внешними приложениями, написанными на языках ***C*** и ***Fortran***.
Осуществляется это взаимодействие через **MEX-файлы**.

MEX-файлы представляют собой динамически подключаемые библиотеки, которые могут быть загружены и исполнены интерпретатором, встроенным в MATLAB.
Существует также возможность вызова подпрограмм, написанных на ***C*** или ***Fortran*** из MATLAB, как будто это встроенные функции пакета.
MEX-процедуры имеют также возможность вызывать встроенные команды MATLAB.

2.3.3 Система Simulink

Программа Simulink является приложением к пакету **MATLAB** [14, 15].
При моделировании с использованием **Simulink** реализуется принцип визуального программирования, в соответствии с которым, пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет расчеты.
При этом, в отличие от классических способов моделирования, пользователю не нужно досконально изучать язык программирования и численные методы математики.
Ему достаточно общих знаний требующихся при работе на компьютере и, естественно, знаний той предметной области в которой он работает.

Simulink является достаточно самостоятельным инструментом **MATLAB** и при работе с ним совсем не требуется знать сам **MATLAB** и остальные его приложения.

С другой стороны, доступ к функциям **MATLAB** и другим его инструментам остается открытым и их можно использовать в **Simulink.**
Часть входящих в состав пакетов имеет инструменты, встраиваемые в **Simulink**, например, **LTI-Viewer** приложения **Control System Toolbox –** пакета для разработки систем управления.
Имеются также дополнительные библиотеки блоков для разных областей применения, например, **Power System Blockset** – моделирование электротехнических устройств, **Digital Signal Processing Blockset** – набор блоков для разработки цифровых устройств и т.д).

При работе с Simulink пользователь имеет возможность модернизировать библиотечные блоки, создавать свои собственные, а также составлять новые библиотеки блоков.
При моделировании, пользователь может выбирать метод решения дифференциальных уравнений, а также способ изменения модельного времени, с фиксированным или переменным шагом.
В ходе моделирования имеется возможность следить за процессами, происходящими в системе.
Для этого используются специальные устройства наблюдения, входящие в состав библиотеки **Simulink**.
Результаты моделирования могут быть представлены в виде графиков или таблиц.

Основное преимущество Simulink заключается также в том, что он позволяет пополнять библиотеки блоков с помощью подпрограмм написанных как на языке **MATLAB,** так и на языках **C++, Fortran** и **Ada**.

2.4 Рекомендуемая литература для самостоятельной подготовки

1. Wolfram Research — Википедия. - http://ru.wikipedia.org/wiki/Wolfram_Research.
2. Wolfram Research: Mathematica, Научная и Техническая Программа. - http://www.wolfram.com/.
3. CUDA — Википедия. - http://ru.wikipedia.org/wiki/CUDA.
4. OpenCL — Википедия. - http://ru.wikipedia.org/wiki/OpenCL.
5. GPGPU — Википедия. - http://ru.wikipedia.org/wiki/GPGPU.
6. OpenGL — Википедия. - http://ru.wikipedia.org/wiki/OpenGL.
7. OpenAL — Википедия. - http://ru.wikipedia.org/wiki/OpenAL.
8. Maple — Википедия. - http://ru.wikipedia.org/wiki/Maple.
9. Maplesoft. - http://www.maplesoft.com/.
10. Mathcad — Википедия. - http://ru.wikipedia.org/wiki/Mathcad.
11. PTC — PTC Mathcad — Engineering Calculations Software. - http://www.ptc.com/product/mathcad/.
12. MuPAD 3.0: заявка на лидерство. - http://ko.com.ua/mupad_3_0_zayavka_na_liderstvo_17785.
13. MATLAB — Википедия. - http://ru.wikipedia.org/wiki/MATLAB.
14. MathWorks — MATLAB and Simulink for Technical Computing. - http://www.mathworks.com/.
15. Сообщество пользователей MATLAB и Simulink. - http://matlab.exponenta.ru/.

2.5 Вопросы для самостоятельного контроля знаний

1. В чем состоит историческая значимость парадигмы «программа-массив»?
2. Каковы концептуальные ограничения представлений о компьютере как сложном многофункциональном вычислителе?
3. Какова роль операционные систем и систем разработки программного обеспечения в развитии вычислительных технологий?
4. Какова значимость технологий расчетов и моделирования в общей концепции вычислительных технологий?
5. Каково назначение и особенности реализации интегрированных систем научных и инженерных исследований?
6. Каково назначение и основные характеристики проекта Mathematica?
7. Каково назначение и основные характеристики проекта Maple?
8. Каково назначение и основные характеристики проекта Mathcad?
9. Каково назначение и основные характеристики проекта MATLAB?
10. Каково назначение и основные характеристики проекта Simulink?

3 ТЕХНОЛОГИИ ХРАНЕНИЯ ИНФОРМАЦИИ

Несмотря на потрясающие достижения вычислительных технологий, парадигма «*Программа-массив*» стала? *не только искаженно отображать* концептуальные стремления теоретиков программирования и применения средств ВТ в социуме, но и *существенно тормозить эти стремления*.

Первые успехи применения цифровых средств ВТ в промышленности, потребовали создание и внедрение в социум, *новой парадигмы*, адекватной новым задачам автоматизации производства.
Такая парадигма была создана и получила первоначальное название «*Информационный подход*».
В дальнейшем, стали развиваться другие интерпретации этой идеи.
Мы рассмотрим одну из таких интерпретаций, которую назовем «***Технологии хранения информации***».

Несмотря на кажущуюся «узость» такой интерпретации, в данном разделе будет показано, что на самом деле такой взгляд имеет место быть и подтверждается целым рядом системных решений, которые сохраняют самостоятельное технологическое направление развития программного обеспечения средств цифровой ВТ.
Непосредственно, в данном разделе рассматриваются следующие вопросы:

- *Парадигма* информационного подхода.
- *Технологии структурирования и формализованного описания* предметной области.
- *Универсальные способы* представления, хранения и обработки информации.
- *СУБД*. Системы и технологии проектирования.
- *Конкретные технологии* ADO.NET, MS SQL Server, Oracle.

3.1 Парадигма информационного подхода

Как было показано в предыдущем разделе, основной технологический прорыв в сфере создания и применения компьютеров, обусловлен первыми тремя этапами их развития:

- *Нулевым поколением* (1492-1945), подготовившим и накопившим потенциал для создания цифровых средств ВТ;
- *Первым поколением* (1937,1945 - 1953,1955), реализовавшим первые ЭВМ на базе электронных ламп;
- *Вторым поколением* (1954,1955 - 1962,1965), которое, на базе полупроводников и транзисторов, обеспечило широкие *возможности внедрения*

средств ВТ в производство.

Главенствующая парадигма «Программа-массив», породившая бурное развитие «Вычислительных технологий», *стала очень быстро себя исчерпывать*.
Следует отметить, парадигма «Программа-массив» породила три основные проблемы:

- проблему *сложного взаимодействия* большого количества программ;
- проблему *подготовки и ввода* большого количества исходных данных;
- проблему *сохранения результатов расчетов* и, затем, *их обработку*.

Как стремление решить перечисленные проблемы, стала интенсивно развиваться *парадигма информационного подхода*.
Первоначально, она была сформулирована и стала развиваться как «*Технология проектирования предметной области*».
Такой подход обоснован реальными задачами промышленности по автоматизации задач управления производством.

Проектировщики АСУ столкнулись с явным фактором *преобладания значимости типизации и хранения данных* над фактором *сложности самих вычислений*.
Этот идейный антагонизм указанных парадигм создал объективные условия для появления и развития нового технологического направления.
В теоретическом плане были выделены две компоненты такой технологии:

- *технологии описания предметной области*, предполагающие теоретическое изучение сложных, в том числе и социальных объектов, и документирование этих описаний;
- *технологии универсального*, применительно к ЭВМ, *представления данных*, разработки универсальных средств хранения и манипулирования этими данными, а также разработки универсальных (машинных) языков, обеспечивающих инструменты программирования, применительно к этим универсальным преставлениям данных.

Первая компонента этих технологий, со временем, стала интегрироваться с другими традиционными подходами и называться просто *проектированием*.
Вторая компонента технологий сформировала собственное технологическое направление и стала называться *системами управления базами данных* (СУБД).
Со временем, появись и **смешанные технологии**, которые обеспечили *автоматизацию проектирования информационных систем*.

Чтобы подробнее раскрыть данную тему, кратко рассмотрим ряд качественных свойств, которые присущи сложным программно-техническим системам.
Такими свойствами являются: *переносимость и интероперабельность информационных систем*.
Продукты, которые сегодня принято называть информационными системами, появились много лет назад.

В основе первых информационных систем находились мэйнфреймы компании IBM, файловая система OC/360, а впоследствии, - ранние СУБД типа IMS и IDMS.
Эти системы прожили долгую и полезную жизнь, многие из них до сих пор эксплуатируются.

С другой стороны, полная ориентация на аппаратные средства и программное обеспечение корпорации IBM породила серьезную проблему "**унаследованных систем**" (legacy systems).
Это связано с тем, что *производственный процесс не позволяет прекратить или даже приостановить использование морально устаревших систем*, чтобы перевести их на новую технологию.
Многие исследователи, сегодня, заняты попытками решить эту проблему.
Серьезность проблемы унаследованных систем очевидно показывает, что информационные системы и, лежащие в их основе, базы данных являются слишком ответственными и дорогими продуктами, чтобы можно было позволить себе их переделку, при смене аппаратной платформы или даже системного программного обеспечения.
Это касается, главным образом, операционной системы и СУБД.

Для решения этой проблемы, программный продукт должен обладать свойствами *легкой переносимости* с одной аппаратно-программной платформы на другую.
Это не означает, что при переносе не могут потребоваться какие-нибудь изменения в исходных текстах; главное, чтобы такие изменения не означали переделки системы.

Другим естественным требованием к современным информационным системам является *способность наращивания их возможностей* за счет использования дополнительно разработанных, а еще гораздо лучше, уже существующих программных компонентов.
Этого требует обеспечение свойства, называемого *интероперабельностью*.
Интероперабельность - соблюдение определенных правил, а также привлечение дополнительных программных средств, обеспечивающих *возможность взаимодействия независимо разработанных программных* модулей, подсистем или даже функционально завершенных программных систем.

Возникает естественный вопрос: «Каким образом можно одновременно удовлетворить оба эти требования уже на стадии проектирования и разработки информационной системы?».
Ответ может быть следующий: *следуйте принципам Открытых Систем*.
Другими словами, максимально строго *придерживайтесь международных или общепризнанных фактических стандартов*, во всех необходимых областях.
Какие же стандарты следует иметь в виду при разработке информационной системы сегодня.

При использовании текущей технологии, информационная система пишется на некотором языке программирования, в нее встраиваются операторы языка **SQL** и, наконец, включаются какие-либо вызовы библиотечных функций операционной системы.
Значит прежде всего, следует обращать внимание на *степень стандартизации* используемого языка программирования.

На сегодняшний день, приняты международные стандарты языков Фортран, Паскаль, Ада, Си и, совсем недавно, Си++.
Очевидно, что:

- **Фортран**, даже в своем наиболее развитом виде Фортран-95, не является языком, подходящим для программирования информационных систем.
- **Паскаль** не удобен тем, что в его стандарт не включены средства раздельной компиляции. Следовательно, его использование - вряд ли разумно и практично.
- **Язык Ада** пригоден для любых целей. На нем можно писать и информационные системы, что и делают американские и некоторые отечественные военные. Недостаток его - неоправданная «громоздкость».
- *Наиболее хороший стандарт*, на сегодняшний день, существует для языка **C**. Опыт многих лет, прошедших после принятия стандарта, показывает, что, при использовании стандарта **Си ANSI/ISO**, проблемы переноса программ, связанные с особенностями аппаратуры или компиляторов практически не возникают.
- *Важной вехой* является факт принятия стандарта языка **C++**. Это означает, что можно говорить уже о мобильном программировании на C++, в том же смысле, в котором можно говорить об этом сегодня по отношению к C.

Завершим рассуждения о стандартизации языков, упомянув вопросы взаимодействия с базами данных, которые традиционно связываются сейчас с языком **SQL**.
Деликатность этих вопросов вызвана тем, что SQL с самого своего зарождения являлся сложным языком со *смешанной декларативно-процедурной семантикой* и далеко не идеальным синтаксисом.
Тем не менее, судьба распорядилась так, что именно SQL, несмотря на наличие других кандидатов, стал практически единственным используемым языком *реляционных баз данных*.
Сейчас имеется два общепринятых стандарта SQL:

- *Стандарт SQL/89* отражает все особенности первых попыток стандартизации, когда многие важные аспекты в нем - не определены или отданы на определение в реализации. С другой стороны, большинство современных коммерческих реляционных СУБД на самом деле соответствуют стандарту SQL/89.
- *Стандарт SQL/92* является существенно более продвинутым, но язык

SQL/92 настолько сложен, что возникают вопросы о его практической целесообразности. Тем не менее, имеется практическая возможность создания достаточно переносимых программ с использованием SQL/89. Для этого нужно максимально локализовать те части программы, которые содержат не стандартизованные конструкции, и стараться не использовать расширения языка, предлагаемые в конкретной реализации.

Аналогичная ситуация существует и в области *операционных систем*.
Существующий сегодня набор стандартов происходит от интерфейсов операционной системы Unix: **SVID**, **POSIX**, **XPG** и другие.
В большинстве современных ОС эти стандарты поддерживаются, но, как правило, любая ОС содержит множество дополнительных средств.
Поэтому, если стремиться к достижению переносимости приложения, следует *стараться ограничить себя* минимально достаточным набором стандартных средств.
В случае, когда нестандартное решение некоторой операционной системы позволяет существенно оптимизировать работу информационной системы, разумно *предельно локализовать* те места программы, в которых это решение используется.

Сказанное выше, позволяет поговорить о ***middleware***: «*системных*» или «*полусистемных*» средствах поддержки *интероперабельности программных компонентов*, начиная от средств межпроцессных взаимодействий.
Такие средства, проходя через *сетевой механизм вызова удаленных процедур*, заканчиваются средствами *CORBA* (*Common Object Request Broker Architecture*).

Завершая рассмотрение вопросов создания *переносимых и интероперабельных* информационных систем, необходимо упомянуть о *профилях* или *профайлах* стандартов, в окружении которых разрабатывается система.
Чем правильнее выбран профиль, тем более вероятно, что системе удастся прожить долгую и счастливую жизнь.

Таким образом, первые шаги информационного подхода подняли большой пласт проблем, многие из которых еще только требовали своего решения.
Это позволило сосредоточить усилия на создании технологий, обеспечивающих массовое и долговременное хранение данных, доступных, в дальнейшем, для решения научных и производственных задач.
Эти технологии оформились в самостоятельное направление, обобщенно обозначаемое как СУБД.

3.2 Инструментальные средства хранения данных

Очень быстро технологическое направление, призванное создавать инструментальные средства хранения данных, управления данными и доступа к ним, стало отдельной технологий, которая стала называться СУБД или системы управления базами данных.
Были сформированы три основные базовые модели представления данных, которые проявляли свои специфические черты и стали развиваться практически независимо:

- *иерархическая модель* закрепилась за файловыми системами и языками программирования: представление и описание объектов, ООП и другие;
- *реляционная модель*, традиционно ассоциированная с таблицами и имеющая достаточно развитый математический аппарат, стала обслуживать все широкие потребности промышленности и социума;
- *сетевая модель*, как наиболее сложная и обобщающая модель, стала предметом будущих перспективных исследований в области искусственного интеллекта и технологий управления данными.

Фактически все современные практически используемые СУБД ассоциируются, прежде всего, с реляционной моделью.

Реляционная модель стала современной *технологической парадигмой* представления, хранения и управления данными, которая, ментально, ассоциируется с языком SQL.
Сам язык SQL рассматривается и представлен в двух ипостасях (сущностях):

- *язык структурированных запросов* - собственно сам SQL, ориентированный на клиента СУБД;
- *язык процедурных расширений*, предназначенный для программирования самой СУБД на стороне сервера.

Поскольку, сам язык SQL не является языком программирования, в общепринятом смысле, поскольку не имеет средств для автоматизации операций с данными, каждый производитель СУБД стал вводить свои процедурные расширения:

- *хранимые процедуры* - *stored procedures*;
- *процедурные языки-«надстройки»*, позволяющие программировать «внутри» СУБД.

Практически, в каждой современной СУБД применяется ***свой процедурный язык***.
Со временем, был введен стандарт для процедурных расширений, который был представлен спецификацией SQL/PSM:

- *Persistent Stored Modules* или, - постоянно хранимые модули.
- *Имеется также* международный стандарт ANSI - ISO/IEC 9075-4:2003, который регулирует эту проблему.

Перечень процедурных расширений, для наиболее популярных сейчас СУБД, приведен в таблице 3.1.

Таблица 3.1. - Наиболее популярные процедурные расширения языка SQL

СУБД	***Краткое название***	***Расшифровка***
InterBase/Firebird	PSQL	Procedural SQL
IBM DB2	SQL PL	SQL Procedural Language (расширяет SQL/PSM); также в DB2 хранимые процедуры могут писаться на обычных языках программирования: Си, Java и другие.
MS SQL Server/ Sybase ASE	Transact-SQL	Transact-SQL.
MySQL	SQL/PSM	SQL/Persistent Stored Module.
Oracle	PL/SQL	Procedural Language/SQL, который основан на языке Ада.
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language (очень похож на Oracle PL/SQL)

Сейчас, корпорации Microsoft и Oracle, являющиеся известными производителями СУБД, внесли дополнительные изменения в свои процедурные расширения SQL, в основном для написания тригеров баз данных (БД):

- *Microsoft* включила язык ***C#***;
- *Oracle* включила язык Java.

Основное преимущество и популярность реляционной модели - это ассоциация ее с *табличной формой представления данных*:

- это — удобно для запросов к СУБД;
- это — удобно для формирования отчетной документации.

С другой стороны, основные недостатки реляционной модели:

- *избыточность* представления и хранения данных;
- *значительные расходы* ресурсов на поиск и извлечение данных;
- *сложные и затратные механизмы* сортировки данных.

Перечисленные недостатки, в свою очередь, породили *собственные технологии проектирования* баз данных:

- *в теоретическом плане* были разработаны правила (шаблоны), получившие название **нормальных форм** (пять НФ);
- *в плане создания баз данных* появились *специализированные интегрированные системы разработки*, уникальные для каждого производителя СУБД.

Такая парадоксальная ситуация сложилась как по многим причинам «*корпоративной технологической замкнутости*» самих разработок, так и из стремления монополизировать большинство сопутствующих технологий в процессе конкурентной борьбы.
Соответственно, все это отразилось и на *инструментальных средствах проектирования* информационных систем, которые стали входить, как часть общей технологической цепочки, в системы разработки программного обеспечения, контролируемые производителями СУБД.

3.3 Системы и технологии проектирования БД

Конкретные технологии хранения информации рассмотрим на примерах разработок двух конкурирующих между собой корпораций: Oracle и Microsoft.
Начнем с корпорации Oracle - как безусловном лидере данного технологического направления.

Oracle или *Oracle Corporation:* это - американская корпорация, являющаяся крупнейшим в мире *разработчиком программного обеспечения* для организаций, а также крупным *поставщиком серверного оборудования* [1 - 3].
Компания специализируется на выпуске СУБД, связующего программного обеспечения и бизнес-приложений:

- ERP- и CRM-систем;
- специализированных отраслевых приложений.

Наиболее известный продукт компании - **Oracle Database**, который она выпускает с момента своего основания.

Сама компания основана в 1977 году.
Имеет подразделения в более чем 145 странах.
По состоянию на 2011 год, насчитывает 108 тыс. сотрудников.
Штаб-квартира корпорации расположена в США: штат Калифорния, рядом с городом Сан-Франциско.
В 1992 году компания выпустила 7-ю версию своей СУБД Oracle Database, в которой поддерживались триггеры, хранимые процедуры и декларативные ограничения целостности.
В 1994 году, ею приобретается у фирмы DEC подразделение, разрабатывающее СУБД Rdb и все права на этот продукт, начиная с этого времени поставлять несколько систем управления базами данных.
В 1995 году, компания приобретает компанию-разработчика первой в истории многомерной СУБД ***Express*** и инструментарий OLAP на ее основе.
В этом же году, корпорация вошла на рынок связующего программного обеспечения, выпустив ***Oracle Web Application Server*** и объявив стратегические интересы в развитии технологий для трехуровневой архитектуры информационных систем.

Уже в 1997 году была выпущена версия 8 СУБД Oracle Database, в которой поддержаны *элементы объектно-ориентированного проектирования и программирования*, начиная с этого момента компания позиционирует продукт как одновременно объектно-ориентированной и реляционной СУБД.
В 1998 году, Oracle стала первой производительницей интегрированных ERP-пакетов и оборудовала свой комплект бизнес-приложений веб-доступом. В результате, любую операцию в ERP-системе стало возможно осуществлять из браузера.
В следующем году, «интернет-стратегия» нашла отражение в наименованиях продуктов Oracle Database и Oracle Application Server, выпущеных с суффиксом «*i*» после номера версии - 8*i*. Заявлены приоритеты во встраивании обработки XML на стороне СУБД и встраивании Java-машины в СУБД.
В результате поглощения в 2010 году корпорации Sun Microsystems, к Oracle перешли активы MySQL AB и свободно распространяемая СУБД MySQL, которая стала отмечаться как свободная альтернатива Microsoft SQL Server. К Oracle также перешла значительная часть активов, связанная технологиями Java: языком, платформами J2ME, J2SE, J2SE, виртуальной машиной HotSpot.

Oracle всегда активно использовал технологии Java в своих продуктах: в СУБД была включена виртуальная машина Java собственной разработки *Aurora JVM;* выпускается средство разработки на Java - **JDeveloper**.
С начала 2000-х годов, была выпущена большая серия ***связующего ПО***, поддерживающего стандарты Java, а также было проявлено активное участие в проекте *Java Community Process*.
Следует отметить, что корпорация Oracle выпускает достаточно широкий спектр средств разработки. Непосредственно на разработку приложений Java ориентированы следующие средства:

- *JDeveloper* — одна из первых разработок инструментальных систем для Java;
- *NetBeans* - унаследованный актив от Sun Microsystems;
- ***Entperise Pack for Eclipse*** - коллекция надстроек и расширений IDE Eclipse для нужд разработки J2EE.

Пионерские технологии Oracle, во многом, связаны с серией средств разработки ***Designer/Developer***, включающих **Oracle Forms** и **Oracle Reports**.
Долгое время, - это были основные среды разработки для ***Oracle E-Business Suite***.
В настоящее время, эти средства еше поддерживаются, но разработчикам предоставляются средства миграции унаследованных *Forms-приложений* на платформу ***J2EE***.
Среди свободных средств разработки, выпускаемых компанией, следует отметить:

- *Apex* - свободный программный каркас быстрой разработки веб-приложений, встроенный в СУБД;

- *SQL Developer* - бесплатное средство разработки и отладки для SQL и PL/SQL.

Теперь перейдем к рассмотрению технологий корпорации Microsoft, которые в плане нашей тематики представлены СУБД MS SQL Server и, во-многом заимствованными у конкурентов, теоретическими и практическими изысканиями, известными под общим названием ADO.NET:

- *Microsoft SQL Server* - система управления реляционными СУБД, разработанная корпорацией Microsoft [4, 5]. Использует основной язык запросов - Transact-SQL, который был создан совместно Microsoft и Sybase.
- *Transact-SQL*, сокращенно T-SQL, является реализацией стандарта ***ANSI/ISO*** по структурированному языку запросов SQL с расширениями. Он используется для работы с базами данных размером *от персональных до крупных баз данных* масштаба предприятия. Конкурирует с другими СУБД в этом сегменте рынка.

Рассматривая историю вопроса, следует обратиться к 1988 году, когда был анонсирован новый продукт ***Ashton-Tate/Microsoft SQL Server***.
Через четыре года, в начале 1992 года, команда разработчиков SQL Server оказалась на распутье:

- С одной стороны, *уже имелась клиентская база SQL Server*, использующая операционную систему OS/2, которые ждали 32-битную версию SQL Server.
- С другой стороны, *не было точно известно, когда же выйдет OS/2 2.0*. Представители IBM заявляли, что выпуск новой версии состоится осенью 1992 года. Многие воспринимали эти слова со скепсисом.

В июле 1993 года, Microsoft выпустила Windows NT 3.1 и, в течение 30 дней после ее выхода, команда разработчиков SQL Server выпустила первую версию Microsoft SQL Server для ***Windows NT***. Выход был весьма успешен: росли продажи как самой СУБД, так и ОС для нее.
Далее, разработка версии ***SQL Server 2005***, получившей кодовое обозначение ***Yukon***, началась параллельно с подготовкой 64-битной версии ***SQL Server 2000*** под кодовым названием ***Liberty***:

- *Liberty* по функционалу представляла собой ту же самую 32-битную версию, которая отличалась лишь большими возможностями масштабирования.
- *Новый функционал* должен был реализоваться в составе ***Yukon***.

Подобные интриги всегда были свойственны корпорации Microsoft и часто приводили к серьезной критике этой корпорации.
Выйдя на рынок СУБД гораздо позже, чем корпорация Oracle, Microsoft создала язык T-SQL ориентируясь на стандарт ***SQL-92***.
Язык T-SQL сразу стал использовать дополнительный синтаксис для *хранимых*

процедур и обеспечивать поддержку транзакций СУБД с управляющим приложением.

С целью подготовки кадрового состава своих клиентов и популяризации своей продукции, компания выпустила продукт - *Microsoft SQL Server Express*, который является бесплатно распространяемой версией SQL Server.
Данная версия имеет ряд технических ограничений, которые делают ее непригодной для развертывания больших баз данных.
С другой стороны, она:

- *вполне годится* для ведения программных комплексов в масштабах небольшой компании.
- *Она содержит* полноценную поддержку новых типов данных, включая XML-спецификации, и является полноценным сервером СУБД.
- *Она также включает* все необходимые компоненты программирования и поддержку национальных алфавитов и Unicode, он используется в приложениях при проектировании информационных систем или для самостоятельного изучения.
- *Нет никаких препятствий* для дальнейшего развертывания накопленной базы данных на более функциональных типах Microsoft SQL Server.
- *В 2007 году*, Microsoft даже выпустила отдельную утилиту с графическим интерфейсом для администрирования данной версии, которая также доступна для бесплатного скачивания с сайта корпорации.

Следует отметить, что корпорация Microsoft является пионером многих маркетинговых начинаний, которые, во-многом, и обеспечили ее коммерческий успех.
Часто, это приводило к серьезным юридическим проблемам, заслуженно портящим ее репутацию.
Но, в данном случае, предложение социуму бесплатных версий своих продуктов, является новшеством, которое подхватили многие разработчики программных систем.

Непосредственно по отношению ***Microsoft SQL Server Express*** следует учесть следующие ограничения:

- *поддерживается 1 процессор*, но система может быть установлена на любой сервер;
- *используется только 1 Гб* адресуемой памяти;
- *максимальный размер базы данных* не превышает 4 Гб;
- *нет возможности экспорта/импорта* данных;
- *В версиях 2008 и 2008 R2* отсутствует встроенный планировщик заданий ***Agent SQL Server***, но имеется возможность создавать скрипты с командами на языке T-SQL.

3.4 Рекомендуемая литература для самостоятельной подготовки

1. Oracle - Википедия. - http://ru.wikipedia.org/wiki/Oracle.
2. Oracle. - http://www.oracle.com/index.html.
3. О компании Oracle. - http://www.oracle.com/ru/corporate/index.htm.
4. Microsoft SQL Server - Википедия. - http://ru.wikipedia.org/wiki/Microsoft_SQL_Server.
5. Business Intellegence. - http://www.microsoft.com/en-us/sqlserver/default.aspx.

3.5 Вопросы для самостоятельного контроля знаний

1. В чем суть и значимость парадигмы информационного подхода?
2. Какие наиболее известны технологии структурирования и формализованного описания предметной области?
3. Что представляют собой универсальные способы представления, хранения и обработки информации?
4. Что такое СУБД?
5. Какие известны технологии проектирования информационных систем?
6. Что такое технология ADO.NET и как она связана с MS SQL Server?
7. Какие известны информационные технологии корпорации Oracle?
8. Какие особенности работы с СУБД имеются в языке Java?

4 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ТЕХНОЛОНИИ

В то время как, технологии хранения информации, замкнувшись сами в себе, стали развиваться обособленно, предоставляя лишь интерфейсы доступа к хранилищам информации, проблемы программирования постоянно находились на острие внимания социума.

Программирование неразрывно связано с ***языками программирования***, на которых человек пишет программы. Кардинальное различие между человеческим восприятием и эффективностью кодирования программ всегда было предметом разработки различных технологий, повышающих эффективность этой сферы деятельности.

Революционные прорывы, произошедшие в технологиях создания аппаратных средств цифровой ВТ, окончательно развернули технологические подходы к программированию в сторону создания ***языков высокого уровня*** и использования ***интегрированных инструментальных средств разработки***, которые переложили большинство проблем подготовки программного обеспечения на специализированное ПО компьютеров.

Чтобы подробнее и комплексно раскрыть эту проблематику, в данном разделе рассмотрим следующие вопросы:

- Парадигма объектного подхода.
- Объектно-ориентированное программирование.
- Виртуальные машины. Java Virtual Machine. Технология .NET.
- Компонентное программирование.
- Инструментальные среды разработки.

4.1 Парадигма объектного подхода

В реальном мире человек выделяет:

- ***материальные объекты*** живой и неживой природы;
- ***полевые энергетические структуры***, способные взаимодействовать с материальными объектами.

Объекты живой природы, наделенные **психикой**, называются **субъектами**. ***Обобщенное*** же понятие объект используется в науке и социуме, - как нечто целостное, которое:

- ***выделяется*** некоторыми границами и ***называется именами***;
- ***обладает*** свойствами и, возможно, активным действием;
- ***подвергается***, хотя бы чисто теоретически, некоторым действиям, которые можно анализировать.

Все более интенсивное развитие компьютерных технологий и все более

интенсивное их проникновение в социум породило большое разнообразие самих технологий. Возникла необходимость адаптировать многие теоретические концепции и практические приемы их использования к природной ментальности человека. В результате возникла необходимость в парадигме и технологиях объектного подхода.

В компьютерных технологиях, под ***естественным базовым объектом***, следует рассматривать программу:

- программа существует в среде, образованной аппаратной частью компьютера и операционной среды ОС;
- программа имеет целевое (прикладное) назначение;
- программу можно назвать и выполнять с ней действия, характерные для действий с объектами реального мира.

Для формирования парадигмы объектного подхода следует отметить два важнейших состояний программы как объекта:

- **статическое состояние программы**, в котором она рассматривается как файл некоторой файловой системы; в таком состоянии к ней применимы концепции парадигмы «проектирования предметной области»;
- **динамическое состояние программы**, когда она существует в среде ОС как **процесс**; в таком состоянии к ней применимы концепции парадигмы «программа-массив».

Фактически, **парадигма объектного подхода** должна объеденить (синтезировать) в единое целое противоречия, обусловленные статическим и динамическим состояниями программы, и породить технологии, оптимальным образом компенсирующие недостатки указанных противоречий.

Поскольку ***программа*** является **элементарной единицей** всех компьютерных технологий, то развитие парадигмы объектного подхода идет в двух направлениях:

- **во-внутрь**, посредством декомпозиции самой программы;
- **во-вне**, посредством синтеза приложений на базе имеющихся программ.

Компьютерные технологии, связанные с **декомпозицией программы**:

- теоретическое оформление концепции объектно-ориентированного программирования (ООП);
- разработка новых языков программирования и преобразование старых языков, обеспечивающих ООП;
- разработка инструментальных средств, способных реализовать технологию ООП.

Компьютерные технологии, связанные с **интеграцией (синтезом) программ**:

- теоретическое оформление концепций архитектур ***вычислительных комплексов*** (ВК), ***вычислительных систем*** (ВС) и ***рассредоточенных архитектур*** СОД;
- разработка языков, поддерживающих ***компонентное программирование***;
- разработка технологий интегрированных систем на базе ***плагинов*** и архитектуры ***CORBA***.

Поскольку компьютерные технологии, связанные с **декомпозицией программы**, нашли свое воплощение в языках, поддерживающих концепции ООП, базовым понятием такого подхода является понятие объекта.

Объект - это мыслимая или реальная ***сущность***, обладающая отличительными характеристиками, важными в некоторой предметной области:

- обладающими **уникальной идентичностью**;
- выделяющиеся четко определенным **поведением**;
- имеющие точно определяемое **состояние**.

Уникальность (identity), это — мета-свойство объекта, позволяющее определить, указывают ли две ссылки на один и тот же объект или на разные объекты.

Поведение (behavior), это - действия и реакции объекта:

- выраженные в терминах передачи сообщений и изменения состояния;
- видимые извне и воспроизводимые активность объекта.

Состояние (state), это - совокупный результат поведения объекта, понимаемый как одно из стабильных условий, в которых объект может существовать, и охарактеризованных количественно.

В любой момент времени состояние объекта включает в себя:

- **перечень свойств** объекта - статическая часть состояния;
- **текущие значения** этих свойств - обычно динамическая часть состояния.

Концептуально, ООП призвано усилить ***семантическую выразительность*** языков программирования.

Как парадигма, ООП должно постоянно формировать у программиста ***объектные представления предметной области***.

Фактически, ООП увеличивает в каждой программе ***мета-описательную часть предметной области***, слабое присутствие которой является недостатком предыдущих парадигм.

Практически, ООП формализует теоретические ***представления о технологии создания программного обеспечения***, которые недостаточно или неявно присутствуют в познавательной деятельности человека.

Очевидно, языки ООП не решают всех проблем создания программ и не решают всех проблем компьютерных технологий, в целом. Причина состоит в

объективном противоречии между необходимостью:

- сохранить ***универсальность и простоту языка***, необходимые для его изучения и успешного практического применения;
- иметь **семантические средства языка**, необходимые для удобного и адекватного описания предметной области.

Наиболее распространенная **модель ООП**, реализованная во многих языках, предполагает определение объекта через ***понятие класса***.

Формально **класс** - это ***статичное описание*** и ***представление*** множества возможных объектов, которые будут иметь общий шаблон ***уникальности***, ***поведения*** и ***состояния***.

Каждое описание класса должно содержать **специальные методы**, которые вызываются при создании и уничтожении объектов этого класса:

- **конструктор** (constructor) - выполняется при создании объектов;
- **деструктор** (destructor) - выполняется при уничтожении объектов.

Каждый такой класс должен быть представлен как **необходимая внешняя часть программного обеспечения** в виде отдельного файла или как часть некоторой библиотеки.

Сам объект, обладающий описанными в классе свойствами, возникает как ***экземпляр класса*** в процессе порождения его с помощью ***конструктора класса***. Без статического описания класса объект не может быть создан, хотя многие языки ООП позволяют производить ***экспорт*** и ***импорт*** объектов. В любом случае, наличие статического описания класса при использовании объекта является обязательным условием.

С другой стороны, **каждый класс** можно рассматривать как объект, ***у которого есть свойства, присущие свойствам класса***: имя, список полей и их типы, список методов, список аргументов для каждого метода и другие. Такой шаблон, задающий различные классы, называется **мета-классом**.

Все классы большинства языков ООП обладают рядом общих свойств.

Инкапсуляция (encapsulation), это - сокрытие реализации класса и отделение его внутреннего представления от внешнего (интерфейса). *Это свойство присуще не только классам ООП. Другое дело, что пропаганда использования свойств инкапсуляции в ООП более развита.*

Наследование (inheritance), это - отношение между классами, при котором класс использует структуру или поведение другого класса (одиночное наследование), или других (множественное наследование) классов.

Наследование вводит иерархию "***общее/частное***", в которой подкласс наследует от одного или нескольких более общих ***супер-классов***. Подклассы обычно дополняют или переопределяют унаследованную структуру и поведение.

Хотя н*аследование - одно из основных и мощных свойств языков ООП, тем не менее, это свойство порождает проблему неадекватного изменения поведения объектов, при*

изменении (модификации) унаследованных классов.

Полиморфизм (polymorphism), это - **положение теории типов**, согласно которому имена могут обозначать объекты разных, но имеющих общего родителя классов. Следовательно, любой объект, обозначаемый полиморфным именем, может по-своему реагировать на некий общий набор операций.

*Отметим, что со всех точек зрения, **полимофизм**, это — пожалуй самое мощное средство ООП, сближающее формальные языки программирования с естественными языками общения человека.*

4.2 Виртуальные машины и технологии

Компьютерные технологии, связанные с ***интеграцией (синтезом) программ***, имеют возможно большее разнообразие, чем подходы ООП. Естественно, они не столь тщательно проработаны, поскольку охватывают более широкую область применения. Особое место здесь следует выделить ***виртуальным машинам***.

Виртуальная машина (ВМ, *virtual machine*), это — термин, имеющий следующие общие значения:

- ***программная или аппаратная система***, эмулирующая аппаратное или программное обеспечение некоторой платформы (***target***) и исполняющая программы для ***target***- платформы на ***host***-платформе;
- ***программная или аппаратная система***, виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже ОС, например, «***песочница***»;
- ***спецификация некоторой вычислительной среды***, например, «виртуальная машина языка программирования Си».

Виртуальная машина исполняет некоторый машинно-независимый код, например, байт-код, шитый код, р-код, машинный код реального процессора.

Помимо процессора, ВМ может эмулировать работу, как отдельных компонентов аппаратного обеспечения, так и целого реального компьютера, включая BIOS, оперативную память, жесткий диск и другие периферийные устройства. На одном компьютере может функционировать несколько виртуальных машин.

Концепция виртуальной машины как совокупности ресурсов, которые эмулируют поведение реальной машины, появилась в Кембридже в конце 1960-х годов, как расширение концепции виртуальной памяти манчестерской вычислительной машины ***Atlas***.

Общая **особенность виртуальных машин** состоит в том, что ***конкретная ситуация в этом рабочем пространстве*** соответствует ожидаемой.
Процесс не имеет никаких средств для определения того, является ли представленный ему ресурс действительно физическим ресурсом этого типа,

или же он имитируется действиями других ресурсов, которые приводят к аналогичным изменениям содержимого рабочего пространства процесса. Например, процесс не может определить, монопольно ли он использует процессор или же в режиме ***мультипрограммирования*** вместе с другими процессами. В виртуальной машине ни один процесс не может монопольно использовать никакой ресурс, и все системные ресурсы считаются ресурсами потенциального совместного использования. Наконец, использование виртуальных машин ***обеспечивает развязку между несколькими пользователями***, работающими в одной вычислительной системе, обеспечивая определенный уровень защиты данных.

Сама идея виртуальной машины лежит в основе целого ряда операционных систем, в частности, IBM VM/CMS и DEC VAX/VMS. В общем случае, виртуальные машины могут использоваться для:

- защита информации и ограничения возможностей программ, например, песочница;
- исследования производительности ПО или новой компьютерной архитектуры;
- эмуляции различных архитектур, например, эмулятор игровой приставки;
- оптимизации использования ресурсов мейнфраймов и прочих мощных компьютеров, например, IBM eServer;
- вредоносного кода для управления инфицированной системой, например, вирус PMBS, обнаруженный **в 1993 году**, а также руткит SubVirt, созданный **в 2006 году** компанией Microsoft Research, создавали виртуальную систему, в пределах которой ограничивался пользователь и все защитные программы: антивирусы и прочие.
- моделирования информационных систем с клиент-серверной архитектурой на одной ЭВМ, например, эмуляция компьютерной сети с помощью нескольких виртуальных машин;
- упрощения управления кластерами, например, когда виртуальные машины могут просто мигрировать с одной физической машины на другую во время работы;
- тестирование и отладка системного программного обеспечения.

Далее, рассмотрим два примера виртуальных машин.

Java Virtual Machine (**Java VM**, **JVM**), это — виртуальная машина ***Java*** или основная часть исполняющей системы ***Java***, которая называется ***Java Runtime Environment*** (JRE) [1].

Виртуальная машина Java интерпретирует и исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java: ***javac***.

JVM может также использоваться для выполнения программ, написанных на других языках программирования, например, исходный код на языке ***Ada*** может быть транслирован в байт-код Java, который затем может выполниться с помощью JVM.

JVM является ключевым компонентом платформы Java. Так как виртуальные машины Java доступны для многих аппаратных и программных платформ, язык Java может быть использован:

- как связующее программное обеспечение;
- как самостоятельная платформа.

В 1996-м году компания ***Sun Microsystems*** выпустила первую версию документа «Голубая книга JVM», в котором описана спецификация виртуальной машины Java, ставшего отраслевым стандартом ***де-факто*** для платформы Java.

Теперь, кратко рассмотрим технологические идеи использования виртуальных машин корпорацией Microsoft, известные широкому кругу специалистов как ***технология .NET***. Основу ее составляет ***общеязыковая исполняющая среда*** (CLR) [2].

Common Language Runtime или ***CLR***, это — виртуальная машина:

- интерпретирующая и исполняющая код на языке CIL, в который компилируются программы, написанные на ***.NET-совместимых языках программирования***: C#, Managed C++, Visual Basic .NET, Visual J#;
- компонент пакета корпорации Microsoft: ***.NET Framework***.

Среда CLR является реализацией ***спецификации CLI***: *Common Language Infrastructure*. CLR интерпретирует и исполняет код на языке ***CIL***, а также предоставляет ***MSIL-программам***, написанным на языках высокого уровня, поддерживающих ***.NET Framework***, доступ к библиотекам классов .NET Framework, или, так называемой, ***.NET FCL***: *Framework Class Library*.

Несмотря на «***теоретическое насилие***» концептуальных идей использования классов, многие практические реализации используют альтернативные подходы ООП. Такой альтернативой является «***Компонентно-ориентированное программирование***» или КОП [3].

Общее классическое представление КОП на структуру приложения показано на рисунке 4.1. В англоязычной среде, для этого понятия используется аббревиатура ***COP*** или - ***component-oriented programming***.

Компонентно-ориентированное программирование - парадигма программирования, ключевой фигурой которой является ***компонент***.

Компонент в программировании, это - множество классов и языковых конструкций, объединенных по общему признаку. На практике они поддерживаются ***фреймворками*** - программными каркасами.

Рис. 4.1. Структура приложения в классическом представлении КОП.

В большинстве языков программирования нет языковых конструкций прямо отражающих понятие компонента. Обычно компоненты реализуются с помощью стандартных конструкций, таких как классы. Чтобы подробнее разобраться с отличием КОП от классической модели ООП, рассмотрим их основные отличия и кратко обсудим ряд проблем традиционного подхода.

Обычно выделяют три отличия ***КОП*** от ***ООП***:

- Компонент - «независимый модуль программного кода, предназначенный для повторного использования и развертывания».
- Может содержать «множество классов».
- Как правило, независим от конкретного языка.

В общем случае, КОП включает в себя набор ограничений, налагаемых на механизм ООП. Это было сделано ***для повышения надежности больших программных комплексов***.

В ООП известна «***Проблема хрупких базовых классов***», которая возникает при изменении реализации ***типа-предка***.

Хрупкий базовый класс (ХБК) - фундаментальная проблема ООП. Она заключается в том, что малейшие правки в деталях реализации базового класса могут привнести ошибку в производные классы. В худшем случае, это приводит к тому, что любая успешная модификация базового класса требует предварительного изучения всего дерева наследования, и зачастую невозможна без создания ошибок, даже в этом случае.

Проблема ХБК сильно снижает ***ценность наследования***. В общем случае, она - не решаема, и является одним из существенных недостатков ООП.

Проблема ХБК может быть обобщена и на системы, разработанные не на ООП языках, и не использующие понятие «класс». Любое повторное использование готового кода (без копирования), как части нового кода, может повлечь за собой такую проблему.

В современных парадигмах программирования, разработанных под влиянием ООП, используются понятия «***связи***» и «***связность***». Наблюдается тенденция, подразумевающая ***ослабление связей***.

Наследование же в понимании ООП создает ***сильнейшую возможную связь***, и, таким

образом, должно использоваться с большой осторожностью. Возможные методы борьбы - ***замена наследования агрегацией***.

При агрегации вложенный ***объект базового класса*** описывается явно, ***как часть объекта производного класса***, и производный класс может пользоваться только публичным интерфейсом базового класса.

Таким образом, **производный класс** не может зависеть от деталей реализации базового класса, что решает проблему.

Немного истории: в 1987 году Вирт, ***унифицировав язык Оберон***, предложил ***шаблон проектирования*** или, ***паттерн***, для написания блоков программ. Блок, удовлетворяющий требованиям этого паттерна, стал называется ***компонентом***. Данный паттерн сформировался при изучении проблемы хрупких базовых классов, возникающей при построении ***объемной иерархии классов***. Сам паттерн заключался в том, что компонент компилировался отдельно от других, а на стадии выполнения необходимые компоненты подключались динамически.

В 1989 году Мейер предложил ***идею единого взаимодействия между вызываемым и вызывающим компонентами***. Эта идея воплотилась в виде готовых решений: ***CORBA***, ***COM***, ***SOAP*** и ***Java***. Впоследствии, поддержка идеи КОП осуществилась в ***компонентном Паскале***.

Наиболее ярко все эти идеи ООП и КОП воплотились и хорошо видны в программных системах, именуемых ***инструментальные средства (среды) разработки***.

4.3 Инструментальные средства разработки

Инструментальные средства (среды) разработки, обозначаемые ***ИСР*** или ***IDE***, - сами являются прикладным программным обеспечением, поэтому для них характерны все черты прикладного ПО.

До тех пор, пока:

- программное обеспечение было в основном проприоретарным;
- отсутствовали или были слабыми средства сетей поддержки ПО;
- отсутствовали развитые графические средства;
- программирование было только уделом профессионалов;
- ***компонентное программирование*** рассматривалось как вспомогательный элемент самой технологии программирования или как область научных исследований.

В таких условиях ***модульность ПО*** реализовалась:

- библиотеками программ;
- библиотеками (пакетами) классов;
- патчами версий ПО;
- дистрибутивами ПО.

С другой стороны, такие ***современные факторы***, как:

- появление свободно распространяемого (непроприоретарного) ПО;

- бурное развитие Интернет-технологий;
- конкурентная борьба за качество и скорость обслуживания клиентов;
- развитие сетевых медиа технологий;
- выдвинули КОП в первые ряды технологий создания прикладного ПО.

Одна из технологий КОП, это — **плагины**.

Плаги́н (*plug-in*) - независимо компилируемый ***программный модуль***, динамически подключаемый к ***основной программе***, и предназначенный для расширения и/или использования ее возможностей. Это понятие также может переводиться как «***модуль***». Плагины часто выполняются в виде **разделяемых библиотек**. В общем случае, технология изготовления плагина определяется разработчиком основной программы.

Основные принципы работы плагина:

- Основное приложение предоставляет сервисы, которые плагин может использовать.
- Плагину предоставляется возможность зарегистрировать себя в основном приложении;
- Плагину также предоставляется протокол обмена данными с другими плагинами.
- Плагины являются зависимыми от сервисов, предоставляемых основным приложением и зачастую отдельно не используются.
- Основное приложение независимо оперирует плагинами, предоставляя конечным пользователям возможность динамически добавлять и обновлять плагины, без необходимости внесения изменений в основное приложение.

Примеры использования плагинов:

- В растровом графическом редакторе может быть фильтр, который каким-либо образом изменяет изображение, палитру и прочие атрибуты изображения.
- В виде плагинов выполняется поддержка форматов файлов для звуковых и видео проигрывателей, пакетов офисных приложений, программ обработки звука и графики.
- В программах обработки звука, плагины выполняют обработку и создание звуковых эффектов: такие как ***мастеринг***, применение эквалайзера сжатия динамического диапазона.
- Некоторые плагины изменяют технические характеристики звука: глубину и частоту дискретизации.
- Большой популярностью пользуются плагины для почтовых программ: спам-фильтры, плагины для проверки писем антивирусом и другие.

Плагины Java

Как всегда, разработчики Java находятся в первых рядах создания новых компьютерных технологий. Работы проводятся под руководством ***OSGi Allianse*** — организацией открытых стандартов [4, 5].

OSGi - *Open Services Gateway Initiative*, это - спецификация динамической плагинной (модульной) шины для создания приложений на языке Java.

Общее архитектурное решение OSGi представляется рисунке 4.2, заимствованным из источника [4].

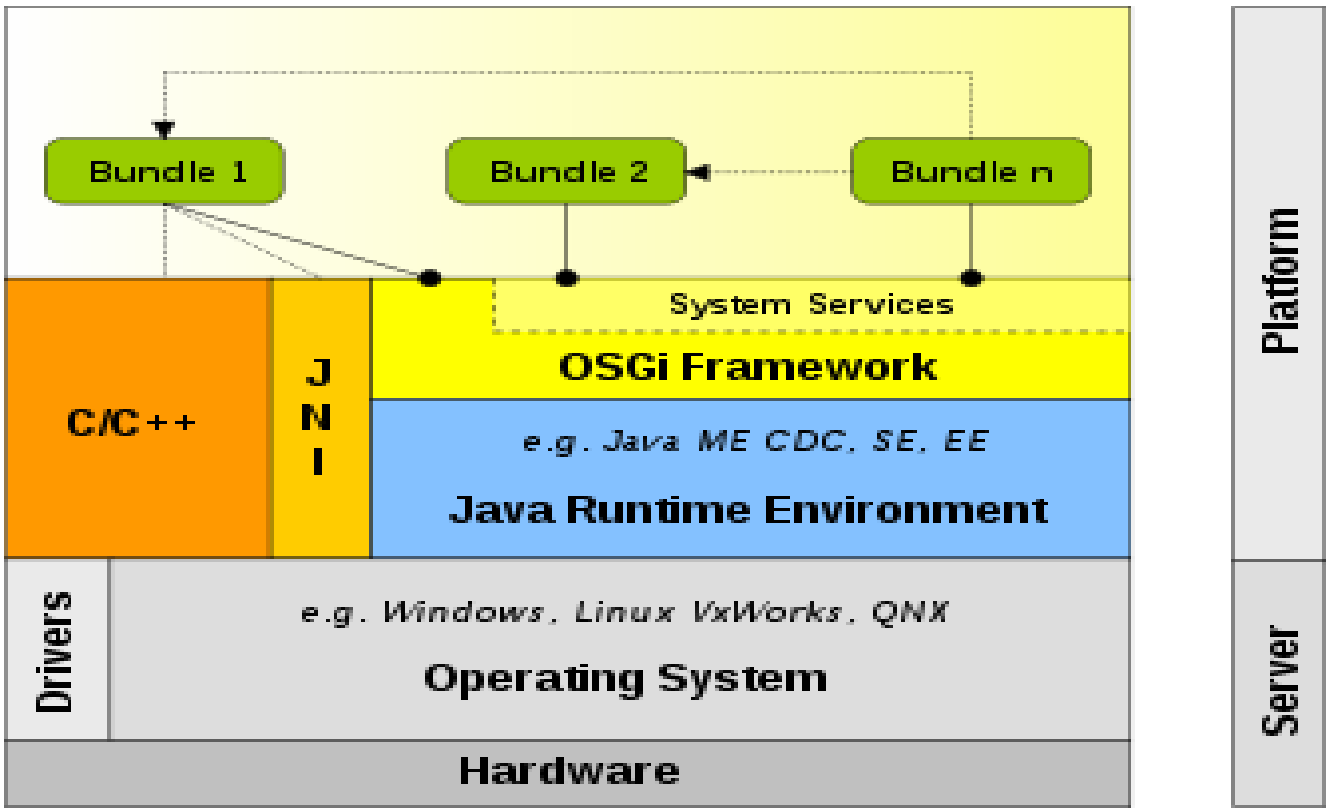

Рис. 4.2. Наборы служб (bundles) OSGi [4].

Изначально данная спецификация разрабатывалась для создания встроенных систем, в частности, для автомобилей BMW. Сейчас на базе OSGi строят многофункциональные ***десктоп-приложения***, например, ***Eclipse SDK***, и различные системы ***Enterprise***.

Версии стандартов OSGi стали разрабатываться с 2000 года:

- OSGi Release 1 (R1): май 2000 г.
- OSGi Release 2 (R2): октябрь 2001 г.
- OSGi Release 3 (R3): март 2003 г.
- OSGi **Release 4 (R4)**: октябрь 2005 / сентябрь 2006 г.
- Core Specification (R4 Core): октябрь 2005 г.
- Mobile Specification (R4 Mobile / JSR-232): сентябрь 2006 г.
- OSGi **Release 4.1**: май 2007 г.
- OSGi **Release 4.2**: сентябрь 2009 г.
- Enterprise Specification: март 2010 г.

В течение последних нескольких лет, OSGi разрабатывала основанную на Java служебную платформу «***The Dynamic Module System for Java***», которая могла управляться удаленно. Основная часть этой разработки - ***framework***, который определяет модель жизненного цикла приложения и служебного реестра.

На основе этого ***framework*** было создано огромное количество ***OSGi-служб***:

- Log.
- Configuration management.
- Permission. Admin.
- Start Level.
- Application Tracking.
- Signed Bundles.
- Declarative Services.

- Preferences.
- Http Service, запускающий сервлеты.
- XML parsing для обработки данных XML.
- Device Access для доступа к устройствам.
- Package Admin.
- User Admin.
- IO Connector.
- Wire Admin.
- Jini -сетевая архитектура для распределенных систем.
- UPnP Exporter - универсальный plag and play
- Power Management для управления питанием.
- Device Management для управления устройствами.
- Security Policies.
- Diagnostic/Monitoring and Framework Layering.

С открытым исходным кодом существуют ***четыре реализации стандарта OSGi***:

- ***Apache Felix***, ориентированный на веб-технологии;
- ***Knopflerfish***, являющийся названием одноименной фирмы;
- ***Equinox***, являющийся базовой частью проекта Eclipse:
- ***Concierge OSGi***, предназначенный для мобильных устройств и встраиваемых систем.

Основная идея фреймворка OSGi - все в системе есть плагины: в терминах OSGi - бандлы (bundles). Фреймворк OSGi задает ***динамическую шину приложений***.

Основной способ взаимодействия между бандлами - сервисы: объекты, зарегистрированные в ядре системы с заявленными реализованными интерфейсами. Бандлы регистрируют сервисы для ***предоставления определенной функциональности*** другим бандлам. Архитектура таких сервисов представлена на рисунке 4.3, заимствованном из [5].

Помимо этого, OSGi предоставляет:

- механизм создания и обработки событий,
- управление импортом/экспортом Java-пакетов и библиотек,
- набор ***классслоадеров***, - загрузчиков классов;
- методы адресации ресурсов.

Само понятие "**динамическая шина**" обозначает, что можно, не перезапуская приложение, устанавливать, подключать, отключать и обновлять модули системы. Это очень удобно в частности для Enterprise приложений, где важен высокий ***uptime*** — высокая готовность приложений.

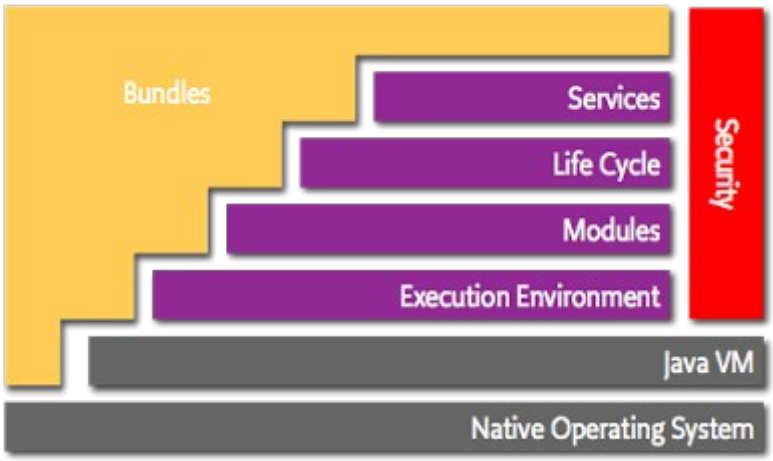

Рис. 4.3. Архитектура сервисов OSGi [5].

В общем случае, ***бандл платформы OSGi*** (OSGi bundle):

- содержит java-классы и другие ресурсы, которые вместе могут реализовывать некие функции;
- предоставляет сервисы и пакеты другим плагинам.

Конструктивно, бандл может быть **каталогом** либо **jar-архивом**. Бандл, который ***содержит в себе фреймворк и управляет жизненным циклом других бандлов*** называется **системным**.

Изначально OSGi разрабатывалась для встроенных систем, поэтому возникла проблема экономии ресурсов. Для ее решения вводится ***понятие жизненного цикла бандла***.

Если система спроектирована правильно, то:

- нет смысла держать в памяти все бандлы;
- неактивные бандлы можно выгружать;
- можно снова загружать бандлы по мере надобности.

Жизненный цикл бандла - набор состояний, в которых он может находиться. В рамках модели OSGi, этот набор состояний представляется рис. 4.4, заимствованным из [6]. Здесь отражены следующие состояния:

- **INSTALLED** - успешно установлен;
- **RESOLVED** - разрешены все зависимости. Бандлу доступны все Java-классы и те бандлы, от которых он зависит. Данное состояние показывает, что бандл готов к старту;
- **STARTING** - бандл стартует. Метод *BundleActivator.start()* выполняется и пока не вернул значения;
- **ACTIVE** - бандл успешно запустился. Метод *BundleActivator.start()* вернул значение;
- **STOPPING** - останавливается бандл. Метод *BundleActivator.stop()* вызван, но пока не вернул значения;
- **UNINSTALLED** - бандл не установлен (удален), соответственно он не может переходить в другие состояния. Жизненный цикл бандла завершен.

Для каждого бандла, подключенного к фреймворку, находящемуся как минимум в состоянии INSTALLED, существует связанный с ним **объект *Bundle***, который используется для управления жизненным циклом бандла.

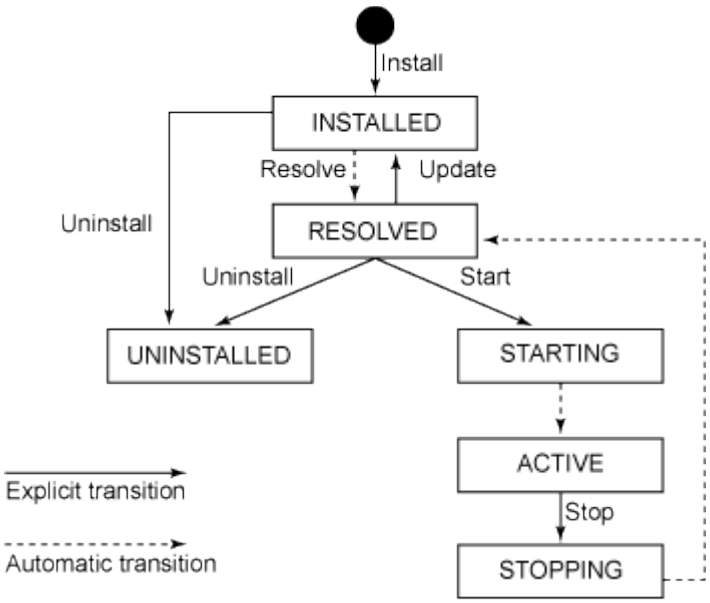

Рисунок 4.4 - Состояния бандла OSGi [6].

В завершение обсуждения этого раздела, рассмотрим плагины проекта **IDE Eclipse** [7], который использует технологию ***Equinox***. На основе ***Equinox*** построена среда разработки ***Eclipse 3.0+***, претендующая на звание отраслевого стандарта компонентной сборки программ.
Equinox — проект Eclipse, который представляет собой фреймворк, реализующий спецификацию ***OSFi R4***. Equinox, по свой сути, является системой поддержки плагинов, которые позволяют разработчикам создавать приложения в виде набора бандлов, использующих общие сервисы и инфраструктуру. В версии ***Eclipse 3.0***, ***Equinox*** заменил старую систему поддержки плагинов, использовавшуюся в более ранних версиях.

Таким образом, технология OSGi предоставляет *сервис-ориентированную платформу* с поддержкой модульности для разработки приложений.

4.4 Рекомендуемая литература для самостоятельной подготовки

1. Java Virtual Machine - Википедия. - http://ru.wikipedia.org/wiki/Java_Virtual_Machine.
2. Common Language Runtime - Википедия. - http://ru.wikipedia.org/wiki/Common_Language_Runtime.
3. Компонентно-ориентированное программирование - Википедия. - http://ru.wikipedia.org/wiki/%CA%EE%EC%EF%EE%ED%E5%ED%F2%ED%EE-%EE%F0%E8%E5%ED%F2%E8%F0%EE%E2%E0%ED%ED%EE%E5_%EF%F0%EE%E3%F0%E0%EC%EC%E8%F0%EE%E2%E0%ED%E8%E5.
4. OSGi - Википедия. - http://ru.wikipedia.org/wiki/OSGi.
5. OSGi Alliance. - http://www.osgi.org/Technology/WhatIsOSGi.
6. Understanding Eclipse's new bundle-management mechanism. - http://www.ibm.com/developerworks/opensource/library/os-eclipse-bundlemgmt/?S_TACT=105AGY75.
7. Проект Eclipse. - http://www.rsdn.ru/article/devtools/eclipse.xml.

4.5 Вопросы для самостоятельного контроля знаний

1. В чем суть и значимость парадигмы объектного подхода?
2. Какие существуют наиболее известные подходы во множестве концепций объектно-ориентированного программирования?
3. Какова идея виртуальных машин?
4. Что такое Java Virtual Machine?
5. Когда и зачем появилась технология .NET?
6. Что такое - компонентное программирование?
7. Зачем нужны и какие известны инструментальные среды разработки?

5 ОФИСНЫЕ ТЕХНОЛОГИИ

Офисные технологии — результат интеграции технологических достижений средств вычислительной техники применительно к прикладному направлению, связанному с *индивидуальной автоматизированной обработкой информации*.
Первоначально, такая индивидуальная автоматизированная обработка информации рассматривалась как *набор приложений разных производителей*, который комплексно обеспечивает электронную подготовку документов с последующим переносом их на бумажный носитель.
Со временем, такой подход стал изменяться в сторону *объединения отдельных рабочих мест* в группы коллективного пользования.
Окончательно, офисные технологии стали рассматриваться как элементы систем автоматизации различных уровней.

В данном разделе, офисные технологии рассматриваются в трех аспектах:
- как *набор приложений*;
- как *элемент системы документооборота*;
- как *элемент интеграции* со стационарными хранилищами информации.

5.1 Офисный набор приложений

Офисный пакет— набор приложений, предназначенных для обработки электронной документации на персональном компьютере.
Компоненты офисных пакетов:
- *распространяются*, как правило, только вместе;
- *имеют* схожий интерфейс;
- *хорошо взаимодействуют* друг с другом.

Как правило, офисный пакет содержит следующий набор компонентов:
- *Текстовый процессор* — средство для создания сложных документов, содержащих текст, таблицы, графику и другое;
- *Табличный процессор* — средство для массовых табличных вычислений;
- *Программу подготовки и демонстрации презентаций* — позволяющую создавать красочные и впечатляющие электронные презентации;
- *Упрощенную СУБД* — позволяющую управлять базами данных или обеспечивать доступ к СУБД на уровне языка SQL;
- *Графическую программу* — позволяющую редактировать графические форматы файлов;
- *Редактор формул* — позволяющий создавать и редактировать математические формулы.

Свободные офисные пакеты

- GNOME Office — офисный пакет проекта GNOME;
- Calligra Suite — офисный пакет из состава оболочки KDE;
- OpenOffice.org — офисный пакет, сравнимый по возможностям и информационно совместимый с офисным пакетом Microsoft Office;
- LibreOffice — ответвление разработки OpenOffice.org с более прозрачной разработкой и свободным лицензированием;
- Kingsoft Office Suite Free — китайский офисный пакет для ОС Windows.

Проприоретарные офисные пакеты

- Microsoft Office — один из наиболее известных офисных пакетов, на данный момент последней является *шестнадцатая* версия, известная также, как Microsoft Office 2016;
- IBM Lotus Symphony — бесплатный офисный пакет корпорации IBM, основанный на OpenOffice.org;
- Ashampoo Office;
- Ability Office — британский дешёвый офисный пакет, появившийся в 1985 году. На данный момент существует уже десять версий этого продукта, который включает в себя:
  - Write — текстовый обработчик.
  - Spreadsheet — табличный редактор.
  - Database — редактор реляционных баз данных.
  - Photopaint — редактор растровой графики.
  - Presentation — создатель презентаций.
  - PhotoAlbum — создатель электронных фотоальбомов.
- Corel WordPerfect Office — пакет Corel Corporation;
- Lotus SmartSuite — офисный пакет корпорации IBM, информационно совместим с OpenOffice.org
- StarOffice — офисный пакет корпорации Sun, информационно совместим с OpenOffice.org
- SoftMaker Office — пакет немецкой компании SoftMaker Software GmbH;
- Kingsoft Office — китайский офисный пакет;
- iWork — офисный пакет Apple для Mac OS X и iOS;
- Облако@Mail.Ru — онлайновый офисный редактор, позволяющий создавать текстовые документы, а также таблицы и презентации.

Идеалогия развития офисных пакетов достаточно хорошо просматривается по динамике и составу выпуска первых пакетов MS Office, которая представлена в таблице 5.1.

Хорошо видно, что основу первых пакетов составляли:

- графический редактор *Word*;
- электронные таблицы *Excel*;
- система презентаций *PowerPoint*.

Таблица 5.1 — Первые выпуски пакетов MS Office

Название	***Состав***	***Год выпуска***
Office 1	Word 3, и другие.	19 ноября1990
Office 2	Word 4, и другие.	1992
Office 3	Word 5, Excel 4, PowerPoint 3, и другие.	1993
Office 4.2	Word 6, Excel 5, PowerPoint 4, и другие.	1994
Office 4.2.1	Word 6, Excel 5, PowerPoint 4, и другие (первый выпуск для PPC, последний для 68K).	1994
Office 98 (8.0)	Word/Excel/PowerPoint 98.	15 марта, 1998

Дальнейшее совершенствование офисных систем предполагало включение в них:

- ПО типа СУБД *Access*;
- программу рисования *Paint*;
- редактор формул *MathType*.

Существенные изменения в идеалогии офисных приложений появились вместе с *тенденцией развития систем автоматизации предприятиями*.
Проблема возникла из-за невозможности прямого переноса офисных технологий в *системы делопроизводства*, которые стали необходимы в системах автоматизации предприятий.

Чтобы определить основные свойства проблемы, рассмотрим *концепцию систем документооборота*.

5.2 Системы документооборота

Часто приходится слышать слова «*делопроизводство*», «*документооборот*», «*электронный архив*», «*деловые процедуры*» и другие.
Как правило, все эти понятия используются как синонимы для пропаганды отдельными разработчиками своих решений и технологий:

- *первоначально*, появился термин - автоматизированные системы документооборота (АСД);
- *затем*, стали использовать термин система электронного документооборота (СЭДО).

Система автоматизации документооборота, **система электронного документооборота (СЭДО)** — автоматизированная многопользовательская система, сопровождающая процесс управления работой иерархической организации с целью обеспечения выполнения этой организацией своих функций.
При этом предполагается, что процесс управления опирается на человекочитаемые документы, содержащие инструкции для сотрудников организации, необходимые к исполнению.

Основные понятия АСД даны в документе: «*ГОСТ 51141-98 Делопроизводство и архивное дело. Термины и определения*».
Тем не менее, до сих пор ***нет четкого понимания***, какой должна быть АСД.

Рассмотрим основные определения:
Документационное обеспечение управления (ДОУ) охватывает вопросы документирования, организации работы с документами в процессе осуществления управления и систематизацию их архивного хранения.
Документирование представляет собой создание документов: *составление, оформление, согласование и изготовление*.
Делопроизводство: комплекс мероприятий по обеспечению ДОУ предприятия или организации. Иногда говорят, что ДОУ является основной функцией делопроизводства.
Организация работы с документами - обеспечение движения, поиска, хранения и использования документов.
Систематизация архивного хранения документов - определение правил хранения создаваемой в организации информации, ее поиска и использования для поддержки принятия управленческих решений и деловых процедур.
Документооборот - движение документов в рамках ДОУ.
Деловая процедура - последовательность определенных операций (работ, заданий, процедур), совершаемых сотрудниками организаций для решения какой-либо задачи или цели в рамках деятельности предприятия или организации.
Электронный архив решает задачи систематизации архивного хранения электронных документов в рамках ДОУ.

Делопроизводство отвечает за документационное обеспечение управления предприятием.

Деловые процедуры отвечают за ведение бизнеса или выполнение целевой функции и являются способом осуществления практического управления предприятиями и учреждениями.

Делопроизводство и деловые процедуры

Отличительные особенности *делопроизводства* покажем на примере *деловых процедур* по продаже товара клиенту:

1. Клиент звонит в компанию для размещения заказа.
2. Заказ регистрируется в базе данных клиентов.
3. Выписывается счет на товар.
4. Счет передается в бухгалтерию.
5. Бухгалтерия получает деньги за товар, что фиксируется в бухгалтерской системе.
6. Товар отгружается со склада, что отмечается в складской базе данных.
7. Выписывается счет-фактура и накладная на товар.
8. Товар отгружается клиенту.
9. Счет-фактура и накладная передаются в бухгалтерию.

В этой деловой процедуре к *делопроизводству* имеют отношение пункты:

- 3 (создание счета),
- 4 (передача счета),
- 7 (создание счета-фактуры и накладной)
- 9 (передача счета-фактуры и накладной).

Рисунок 5.1 - Отличие делопроизводства от деловых процедур

Если продажи устроены более сложно, например, при наличии формальных внутренних отношений между отделом продаж, складом и бухгалтерией, то в делопроизводстве могут появиться дополнительные процедуры.

Таким образом, делопроизводственные операции как бы вплетаются в деловые процедуры там, где их необходимо сопроводить документами.

Для государственных организаций, деловые процедуры могут состоять исключительно из делопроизводственных операций.

Основное отличие *делопроизводства* от *деловых процедур*, состоит в их функциональной разнице:

- делопроизводство отвечает за документационное обеспечение управления предприятием;
- деловые процедуры - за ведение бизнеса или выполнение целевой функции и являются способом осуществления практического управления предприятиями и учреждениями.

Во всех случаях, делопроизводство включает в себя документационное обеспечение деловых процедур.

Замечание

Имеется различие между продуктами и технологиями автоматизации на Западе и в России.
Автоматизированные решения для российских предприятий должны в большей мере учитывать *наличие бумажных документов в делопроизводстве*.
Следовательно, российские предприятия *требуют менее жесткую схему автоматизации деловых процедур*.

Западные системы автоматизации делопроизводства в России

Основные решения для делопроизводства и деловых процедур условно разделяются на четыре основные категории (не включая средства создания документов и складов данных):

1. Системы *workflow* (автоматизации деловых процедур);
2. Системы *groupware* (коллективной работы);
3. Системы *управления документами* (в основном обеспечивают регистрацию, хранение и поиск документов);
4. Системы *электронной почты* (служат для обмена документами).

До недавнего времени серьезные затруднения вызывали:

1. отсутствие законченных продуктов для автоматизации российского делопроизводства;
2. отсутствие активного спроса на средства автоматизации деловых

процедур;

3. путаница в позиционировании продуктов западных производителей для целей автоматизации делопроизводства и другие.

Три источника и три составные части ДОУ

Процесс принятия управленческого решения включает в себя:

- получение информации,
- ее переработку;
- анализ,
- подготовку и принятие решения.

Эти составные части самым тесным образом связаны с документационным обеспечением управления.

Принято выделять три основные задачи, решаемые в делопроизводстве (ДОУ):

1. *Документирование* (составление, оформление, согласование и изготовление документов).
2. *Организация работы* с документами в процессе осуществления управления (обеспечение движения, контроля исполнения, хранения и использования документов).
3. *Систематизация архива* документов.

ДОУ оказывает непосредственное влияние на качество принятия управленческих решений.

С ростом масштабов предприятия и численности его сотрудников вопрос об эффективности ДОУ становится все более актуальным.

Рисунок 5.2 - Компоненты управления предприятием

Основные проблемы, возникающие при этом, выглядят примерно так:

1. Руководство теряет целостную картину происходящего.
2. Структурные подразделения, не имея информации о деятельности друг друга, перестают слаженно осуществлять свою деятельность. Неизбежно падает качество обслуживания клиентов и способность организации поддерживать внешние контакты.

3. Следствием этого становится падение производительности труда; ощущение недостатка в ресурсах: людских, технических, коммуникационных и другие.
4. Приходится расширять штат, вкладывать деньги в оборудование новых рабочих мест, помещения, коммуникации, обучение сотрудников.
5. Для производственных предприятий увеличение штата может повлечь изменение технологии производства, что потребует дополнительных инвестиций.
6. В ситуации неоправданного роста штата, падения производительности, необходимости инвестиций в производство появляется потребность в увеличении оборотного капитала, что, в свою очередь, может привести к новым кредитам и уменьшить плановую прибыль.

В итоге, дальнейшее расширение предприятия происходит чисто экстенсивным путем за счет ранее накопленной прибыли или увеличения дефицита бюджета.
Наиболее распространенное решение - это автоматизация отдельных рабочих мест (АРМ):

- секретаря-референта,
- менеджера,
- бухгалтера или руководителя.

Основными недостатками такого подхода, как правило, являются:

- отсутствие способов организации электронного информационного обмена между сотрудниками и подразделениями предприятий;
- отсутствие функциональной связи автоматизации прикладных процедур с автоматизацией делопроизводственных.

Таким образом, перед предприятием, стремящимся создать эффективную среду по обработке информации для совершенствования качества управления, стоят серьезные задачи:

1. Совершенствование всей работы по подготовке и обработке документной информации путем создания механизма документационного обеспечения предприятия (ДОУ).
2. Выбор правильной стратегии автоматизации, включая верный выбор продуктов.

5.3 Интеграция офисных приложений и СУБД

5.4 Рекомендуемая литература для самостоятельной подготовки

5.5 Вопросы для самостоятельного контроля знаний

6 ТЕХНОЛОГИИ АВТОМАТИЗИРОВАННОГО УПРАВЛЕНИЯ

В предыдущих разделах данного пособия, мы рассматривали идейные парадигмы, которые *фундаментально влияли* на направления развития компьютерных технологий.
Такой подход оправдывает себя *на начальных этапах развития*, позволяя связать воедино многие технологические достижения.

Естественным образом, все технологические достижения средств цифровой ВТ имели соответствующий *отклик в сфере промышленного производства*. Промышленность пыталась и, при возможности, внедряла у себя все успешные решения, которые она была способна освоить.
Знаменательным событием здесь по праву можно считать появление новой науки XX-го века: **кибернетики**.

Автором термина «***кибернетика***» официально считается ***Норберт Винер***, который *в 1945 - 1948 годах* предложил изучать общие закономерности процессов управления и передачи информации в машинах, живых организмах и обществе [1].

В России кибернетика долгое время считалась *«лженаукой» вплоть до 1970 г*. Все исследования, связанные с управлением, проводились в рамках других научных направлений, каких как прикладная математика, радиотехника (радиолокация), медицина (биология) и другие [2].

Основной моделью кибернетики является модель управления некоторым объектом посредством устройства управления *на основе обратной связи*: измерения выхода объекта, преобразования его и воздействие результатом этого преобразования на вход.
Классическая идейная схема такой (кибернетической) модели представлена на рисунке 6.1.

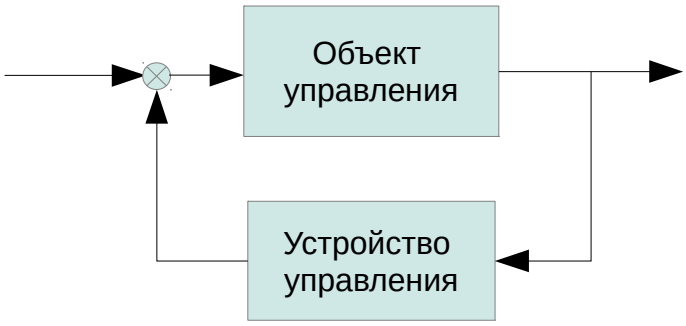

Рисунок 6.1 - Кибернетическая модель объекта управления с обратной связью.

Наиболее общими *способами воздействия* с ***помощью обратной связи***, могут быть:

- *положительная обратная связь*, - тогда получается *генератор*;
- *отрицательная обратная связь*, - тогда получаем *систему управления*.

Таким образом, буквально *начиная с первого поколения* средств цифровой ВТ, в промышленность стали проникать кибернетические идеи, которые пройдя первичную апробацию, возвращались в науку ***в виде конкретных задач***, требуемых промышленному производству.

В данном разделе, мы рассмотрим ряд таких задач и увидим как они влияли на современные компьютерные технологии.

6.1 Компьютерные технологии в промышленности

Классическое развитие модели, показанной на рис. 6.1, предполагает, что *обратная связь* формируется устройством, в частности, компьютером. Такие системы получили название *автоматические системы управления* (**САУ**).

Исторически, наука кибернетика стала разрабатывать и продвигать технологии САУ, потому что к окончанию **Второй мировой войны** *накопился большой материал по математическим моделям объектов управления*, вызванный интенсивным развитием военно-промышленных комплексов ведущих стран мира.
С другой стороны, математический аппарат для работы с этими моделями был накоплен в ходе всех предыдущих научно-технических достижений.

Социальный заказ САУ - очевиден и объясняется *потребностями в управлении ресурсами враждующих сторон*, которые, имея мировой масштаб военных действий и технологическую конкуренцию между собой, подтолкнули развитие соответствующих *военных технологий*.
После окончания активных военных действий и, благодаря присущей социуму инерции развития, высвобожденные ресурсы были направлены на развитие новых технологий управления.
Эти технологии, стимулируя новый качественный виток вооружений, создали необходимые ресурсы для своего дальнейшего развития.

Успехи САУ породили концепции, согласно которым:

- *все конечные операции* воздействия, на объекты физического мира, можно будет выполнять автоматически и, в дальнейшем, - *роботизировать*.
- *управление автоматами* (работами), частично или полностью, может *формироваться посредством человека*.

В отличие САУ, такие системы стали называться: *автоматизированные системы управления* (**АСУ**).

Идеи АСУ сразу же столкнулись со *множеством проблем*:

- человек, за рядом частных случаев, не способен *непосредственно наблюдать* выходы объектов;
- человек не способен *непосредственно воздействовать* на входы объектов;
- человек не может *в реальном времени перерабатывать* выходные сигналы объектов во входные сигналы с учетом математической модели объекта;
- человек, как субъект, обладающий волей, *может самостоятельно формулировать и реализовывать цели управления* уже в процессе управления, *меняя ранее поставленные цели или противодействуя им*.

Несмотря на перечисленные проблемы, очень скоро стало совершенно ясно, что *объектами управления*, *социальным заказом*, *средой* и *ресурсами АСУ* стало *промышленное производство*, которое всегда управлялось человеком.

Кибернетическая модель, представленная на рисунке 6.1, стала рассматриваться как *теоретическая концепция*, которая уже реализована в социальной среде.
Такая модель позволила:

- *переосмыслить* многие взаимодействия между людьми, участвующими в управлении производством;
- *дало научное объяснение* многим, не решенным ранее вопросам;
- *стимулировало* развитие математического аппарата, адекватного построенным моделям.

Хотя, *большинство математических моделей АСУ* ни концептуально, ни синтаксически не напоминают наборы систем дифференциальных уравнений, столь широко используемых в моделях САУ, тем не менее, многие свойства АСУ проявляют себя как системы управления с обратной связью.
Это делает модель рисунка 6.1 универсальной и позволяет выявить *преемственность всех кибернетических моделей*.

С другой стороны, наблюдается существенное отличие технологий АСУ от технологий САУ.
Это порождается принципиальным различием в изучении ***взаимодействия объектов*** и ***субъектов*** реального мира:

- человек, участвующий в управлении, *рассматривается как субъект социума*, имеющий сознание, собственные целевые установки и волю для достижения своих целей;
- человек всегда рассматривается *как часть некоторой социальной струк-*

туры, которая наделяет его правами, обязанностями и ответственностью, включая юридическую ответственность;
- человек, как элемент АСУ, *является одновременно элементом некоторого предприятия* — юридического лица, рассматриваемого как элементарный субъект социума.

Таким образом, *технологии АСУ* - это многоуровневые технологии управления отдельным юридическим лицом (предприятием).
Сложность такого управления — очевидна и традиционного разделяется на три уровня:
- *АСУП* – верхний уровень – управление предприятием;
- *АСУПП* – средний уровень – управление производственными процессами;
- *АСУТП* – нижний уровень – управление технологическими процессами.

Разделение на указанные уровни не является искусственным, а, наоборот, отражает их качественное различие, максимально проявленное в отличиях между АСУП и АСУТП.
Верхний уровень — АСУП отражает взаимодействие предприятия с социумом.
В общем случае, этот уровень рассматривается как:
- *комплекс* программных, технических, информационных, лингвистических, организационно-технологических средств;
- *действия* квалифицированного персонала, предназначенные для решения задач планирования и управления различными видами деятельности предприятия.

Интерпретируя АСУП, посредством рисунка 6.1, можно сказать, что объектом управления является само предприятие (юридическое лицо):
- *Входами* объекта управления являются материальные, информационные и людские ресурсы, которые преобразуются предприятием в *выходы* - продукты производства.
- *В качестве управления* рассматриваются руководящие органы внешней среды и ведущие руководители предприятия, которые с учетом нормативных документов, сформированных внешней средой (социумом), воздействуют на входы предприятия.

Очевидно, что модели такого управления являются динамическими во времени и во многом определяются внешней средой, которая зависит от принадлежности предприятия государству с некоторым общественным строем, географическим положением, отношением предприятия с другими юридическими лицами, вхождением предприятия в другие корпоративные структуры, а также многими экономическими и политическими факторами.
С другой стороны, структура (модель) этого уровня управления является максимально устойчивой и «прозрачной», поскольку опирается на нормативные документы, сформированные и контролируемые социумом до момента создания предприятия как юридического лица.

Среди зарубежных аналогов к категории АСУП принято соотносить:

- *MRP* - *Material Requirement Planning* — планирование потребности в материалах;
- *ERP* - *Enterprise Resource Planning* - планирование ресурсов предприятия.

Нижний уровень — АСУТП отражает отношение человека (работника предприятия) к средствам производства: станкам, производственным линиям, энергетическим установкам и другим.
В общем случае, этот уровень рассматривается как комплекс технических и программных средств предназначенный для автоматизации управления технологическим оборудованием на промышленных предприятиях.

Характерной особенностью этого уровня является тот факт, что производство считается тем более автоматизированным, чем меньшее физическое участие в нем принимает человек.
В этом случае, модель рисунка 6.1 применима в буквальном смысле ко многим элементам АСУТП.
Идеалом, здесь является участие человека в качестве наблюдателя за процессом производства и специалиста по ремонту неисправного оборудования.
В таком случае, АСУТП рассматривается набор (комплекс, система) САУ, объединенных, настраиваемых и управляемых человеком или группой лиц.

Среди зарубежных аналогов АСУТП принято сопоставлять, хотя - это во-многом не так, системы SCADA - *Supervisory Control And Data Acquisition* - Диспетчерское управление и сбор данных.

Средний уровень управления — АСУПП, - *наиболее слабо* определен и структурирован.
Это связано с большим количеством задач управления, которые не только специфичны для конкретного вида производства, но и требуют участия значительного количества различных специалистов, активно и разносторонне влияющих на сам уровень управления.
Многие теоретики АСУ, особенно на ранних стадиях формирования теории считали, со временем произойдет слияние АСУП и АСУТП, в результате *процессов расширения* их «вниз» и «вверх» соответственно.
Многие исследователи вообще интерпретируют АСУПП как *уровень подготовки производства*, неявно исключая с этого уровня сам факт управления.

Зарубежным аналогом АСУТП принято считать системы *MES* - *Manufacturing Execution System* - производственная исполнительная система, функции и задачи которой определяются Международной ассоциацией производителей и пользователей систем управления производством (*MESA International*).
В 1994 году, эта организация сформировала модель MESA-11, определив задачи характерные для АСУПП.
Несмотря на значительное развитие, в последующие годы, множества компьютерных технологий, ассоциация MESA не смогла предложить перспективных

теоретических моделей, кроме обобщенного перечня функций, которые должны выполнять MES-системы.
Поэтому, под давлением потребностей автоматизации производства, *в 2004 г.* была разработана новая упрощенная модель *Collaborative Manufacturing Execution System (c-MES)*, которая удалила из MESA-11 функции, относящиеся к:

- составлению производственных расписаний;
- управлению техническим обслуживанием и ремонтами;
- а также задачи цехового документооборота.

6.2 CALS-технологии

Наряду с кибернетической моделью, представленной рисунке 6.1, на развитие технологий АСУ существенное значение оказали, так называемые, ***CALS-технологии*** - *Continuous Aquisition and Life cycle Support* или непрерывная информационная поддержка поставок и жизненного цикла продукции.

CALS-технологии [3], это — современный подход к проектированию и производству высокотехнологичной и наукоемкой продукции, который заключается в использовании *компьютерной вычислительной техники* и *современных информационных технологий* на всех стадиях жизненного цикла изделия.
Их применение обеспечивает единообразные способы управления процессами и взаимодействия всех участников этого жизненного цикла:

- заказчиков продукции;
- поставщиков и производителей продукции;
- персонала: эксплуатационного и ремонтного;
- международные организации, требующие реализацию продукции в соответствии с системой международных стандартов,
- правила, регламентирующие взаимодействие всех участников жизненного цикла посредством электронного обмена данными.

В практическом плане, наиболее полно технология АСУ проявила себя в системах SCADA или *системах диспетчерского управления и сбора данных*.
Хотя системы SCADA не тождественны АСУТП, а несколько шире, потому что охватывают:

- *системы сбора первичных данных* (полевой уровень), находящийся по иерархии управления ниже АСУТП.
- *на верхнем уровне*, системы SCADA включают функции диспетчеризации, которые управления входят в состав АСУПП.

Значимость систем SCADA состоит в полной логической завершенности набора их функций, необходимых для решения практически значимых задач производства.
В задачах диспетчерского управления, участие человека-диспетчера значитель-

но выше, чем участие человека-оператора в АСУТП.
Хотя, задачи диспетчеризации являются специфическими для различных производств, тем не менее, имеются производства, где диспетчеризация технологических процессов является достаточно важной функцией, если не основной.
Именно такие производства сформировали необходимый социальный заказ, который обеспечил жизнеспособность технологии SCADA.
Сложность и техническая емкость таких систем породили *отдельное техноло-гическое направление* диспетчеризации, включающее проектирование, разработку и программно-аппаратное наполнение таких систем, доступное для реализации только крупным исполнителям, например, таким как корпорация ***Siemens AG***.
Обычно, *основными отличительными признаками* таких АСУ являются:

- *множество удаленных от центра обработки* точек сбора измерительной информации;
- *необходимость хранения* одномерных и многомерных данных (трендов) для последующего, возможного анализа ситуаций, вызвавших сбои или аварии производства;
- *наличие специализированных*, для восприятия человеком, систем представления информации (диспетчерские пульты или стенды), необходимые для обеспечения самого процесса управления.

Следует особо отметить и учесть, что развитие технологий АСУ сопровождалось *формированием документации*, в которой отражались нормативные акты на определения и сам объект автоматизации.
В России (СССР), этот пакет документов выпускался в виде различных ГОСТ-ов, которые формируют пакет документов под общим названием ЕСД АСУ - *единая система документации на АСУ*.

Исторически, общие определения и положения АСУ изложены в *ГОСТ-ах серии 24* начала 80-х годов [4].
В дальнейшем, более популярным становится более расширительный термин *автоматизированные системы* (АС), используемый в *ГОСТ-ах серии 34* [5 - 7].

Применительно к программному обеспечению, входящему в состав АС, разработан пакет документов ЕСПД - Единый стандарт программной документации, содержащий *ГОСТы серии 19* [8 - 10].

Вся перечисленная нормативная документация отражает и ориентирована на *канонический (каскадный) метод* проектирования АС, когда последовательность этапов создания АС начинается и завершается строго последовательно.
Дальнейшие модификации канонического метода проектирования в виде *итерационной* и *спиральной моделей*, применимы и эффективны только в той мере, в

какой удается преодолеть трудности, свойственные самим моделям.

В отличие от САУ, которые в большинстве случаев могут удовлетвориться незначительным усложнением кибернетической модели, показанной на рисунке 6.1, разносторонние и, во многом противоречивые требования к системам и подсистемам АСУ, требовали применения более адекватных моделей.
Основой послужил *функциональный подход описания технологических (бизнес) процессов* в виде последовательности операций, преобразующих входные материальные и информационные объекты при ограничениях, заданных на управляющие сигналы и используемые ресурсы.
Такая модель была предложена департаментом военно-воздушных сил США в 1981 году.
Она получила кодовое название **IDEF0** - *Integrated Computer Aided Manufacturing DEFinition*.

Основные макроопределения модели IDEF0 показаны на рисунке 6.2.
Рекомендации по ее применению определены Госстандартом России в документах *Р 50.1.028 — 2001* [11, 12].

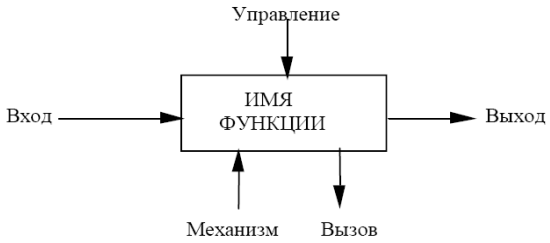

Рисунок 6.2 - Макроопределения модели IDEF0

Важность модели IDEF0 трудно переоценить. Обладая емким, графическим и строго стандартизированным языком, эта модель легко и однозначно отображает основные функции АС любого уровня сложности.
На языке этой модели можно легко систематизировать сложный и многоплановый материал предварительного обследования предприятий и формировать один из важнейших документов, определяющих требования к характеристикам АСУ: «*Техническое задание на АС*».

Очевидно, что в рамках одной научной парадигмы сложно однозначно описать все аспекты технологий проектирования и создания автоматизированных систем.
В частности, многие авторы, выделяя информационную компоненту АСУ,

пишут статьи, монографии и учебники по *автоматизированным информационным системам* (АИС), а, особо выделяя сетевую компоненту, пишут о *корпоративных информационных системах* (КИС).

6.3 Промышленные шины предприятия

Как было отмечено выше, многие практические аспекты АСУ, например, архитектура АСУПП и ее типовые реализации, остаются только в виде концептуального набора задач, имеющих отдельные частные решения. Все это говорит об имеющемся нереализованном потенциале концепции АСУ, ждущем новых технологических решений. В частности, ведутся разработки по созданию промышленных (сервисных) шин предприятия: **ESB** — *Enterprise Service Bus*. *Эти шины* рассматриваются как подход к созданию *распределенных корпоративных информационных систем*, способных решить *задачи автоматизации документооборота*, а также послужить основой объединения разнородных приложений, порожденных результатами «***островной автоматизации***» предприятий.

6.4 Рекомендуемая литература для самостоятельной подготовки

1. Кибернетика - Википедия. - http://ru.wikipedia.org/wiki/%CA%E8%E1%E5%F0%ED%E5%F2%E8%EA%E0.
2. Самарский А. Основы экономики будущего. К 85-летию со дня рождения В.М.Глушкого. - http://propaganda-journal.net/print/111.html (Основы экономики будущего.pdf)
3. CALS-технологии - Википедия. - http://ru.wikipedia.org/wiki/CALS-%F2%E5%F5%ED%EE%EB%EE%E3%E8%E8.
4. ГОСТ 24 (Мнемосхема). - (24.pdf)
5. ГОСТ 34 — комплекс стандартов на АС (Мнемосхема). - (34.pdf)
6. ГОСТ 34.601-90 — Стадии и этапы создания АС.doc
7. ГОСТ 34.602-89 — Техническое задание на создание АС.doc
8. ГОСТ 19.201.78 — порядок построения и оформления технического задания на разработку программы или программного изделия.doc
9. ГОСТ 19.102.77 — Стадии разработки программ и программной документации.doc.
10. ГОСТ 19 (Мнемосхема). - (19.pdf)
11. Р 50.1.028-2001 — Методология функционального моделирования. - (IDEF0 GOST RF.doc)
12. Методология функционального моделирования IDEF0. - (РД IDEF0-2000.pdf)

6.5 Вопросы для самостоятельного контроля знаний

1. Каково концептуальное значение основной кибернетической модели автоматического управления?
2. Чем необходимо дополнить математическую модель, показанную на рис.6.1, если задана система дифференциальных уравнений объекта управления и класс функций, описывающих устройство управления с точностью до параметров?
3. В чем недостатки модели, показанной на рис.6.1, при описании подсистем АСУ?
4. В чем принципиальное различие между подсистемами АСУ: АСУП, АСУПП и АСУТП?
5. Какие проблемы в настоящее время имеются в плане практической реализации АСУ?
6. Что такое SCADA и ее значение для реализации АСУ?
7. В чем основные сложности реализации SCADA-систем?
8. Какие нормативные документы, определяют технологию проектирования и создания АСУ?
9. В чем различие нормативных требований на АСУ (АС) и программное обеспечение?
10. Какая модель и почему более адекватно описывает требования к АСУ по сравнению с моделью, показанной на рис. 6.1?
11. Что общего между моделями, показанными на рис. 6.1 и рис. 6.2?
12. Какие современные компьютерные технологии наиболее актуальны для тематики АСУ?

7 ТЕХНОЛОГИИ ВЗАИМОДЕЙСТВИЯ ОТКРЫТЫХ СИСТЕМ

Когда технические возможности ЭВМ вышли за рамки чисто расчетных задач, стало необходимым решать задачи:

- *сбора информации* от удаленных источников;
- *сохранения этой информации* в центрах обработки;
- *передачи информации* на другие компьютеры.

Сначала, такие задачи рассматривались как вспомогательные и решались в рамках отдельных специализированных способов и подходов с использованием технологий телекоммуникации.
Со временем, многообразие способов передачи информации между компьютерами, а также - между компьютером и периферийными устройствами, стало порождать *проблемы несовместимости* различных телекоммуникационных технологий и *проблемы избыточности* разных технических решений, требующих соответствующего программного обеспечения.

Если посмотреть на структуру памяти отдельной ЭВМ, показанную на рисунке 7.1, то мы увидим достаточно *сложную иерархию устройств*, предназначенных для хранения информации.

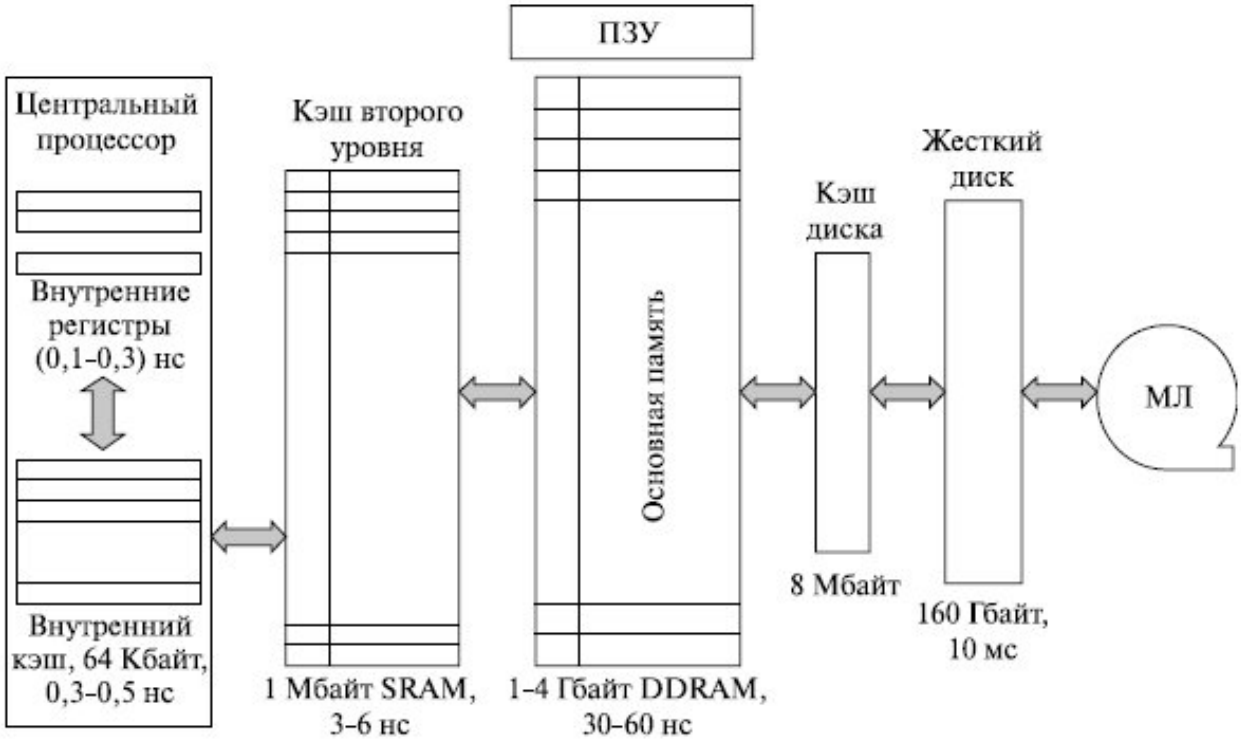

Рисунок 7.1 - Обобщенный вид памяти ЭВМ

Такая, *во-многом оригинальная* структура устройств памяти, поддерживается средствами операционной системы (ОС).
Очевидно, что создание и сопровождение специального ПО, обеспечивающего взаимодействие многих ЭВМ, - очень сложная и трудоемкая задача.

Критическая ситуация возникла с появлением персональных компьютеров (ПК). Их малые вычислительные мощности требовали создания компьютерных сетей, пригодных как *для обмена информацией*, так и для увеличения вычислительных ресурсов ПК.
Ситуация осложнялась с развитием различных информационных технологий, практическое внедрение которых сдерживалось отсутствием, адекватных задачам, технологий взаимодействия удаленных систем.
Многие успешные *телекоммуникационные технологии* военного, государственного и закрытого корпоративного назначения не могли обеспечить потребности общества.
В результате, возник *социальный заказ на создание публичных (открытых) сетей*.
Теоретические концепции таких сетей и их последующее практическое применение стало известно как *технология взаимодействия отрытых систем*.

7.1 Парадигма взаимодействия открытых систем

Прародителем технологии взаимодействия открытых систем следует считать модель **DoD**, *Department of Defense* — Министерство обороны США, которая появилась ***в 1969 году***.
Модель DoD была положена в основу экспериментальной сети *DARPA* - *Defense Advanced Research Projects Agency* и, в последующем, стала известна как сеть **Интернет**.
Следует отметить, что первый стандарт основных транспортных протоколов Интернет, известный как стек протоколов TCP/IP, появился только ***в 1983 году***, а современное состояние сети было окончательно сформировано только ***к 1993 году***.

Несмотря на большую теоретическую и практическую значимость модели DoD, содержащую иерархию из четырех уровней протоколов, многие технологические концепции ее остались недостаточно проработанными и требовали уточнения.
Сама модель была слишком универсальной. Ее реализация была ориентирована на глобальные сети и не учитывала как *проблемы локальных сетей*, так и многие *проблемы использования этих сетей*, в условиях промышленного производства.

Первой и **базовой моделью** современной технологии следует считать сетевую модель **OSI** - *Open System Interconnection basic reference model*, которая появилась ***в 1978 году***.
В русской транскрипции, ее принято обозначать как *ЭМВОС*, - *базовая эталонная модель взаимодействия открытых систем*.
Сама модель была представлена в виде *иерархии семи уровней* [1, 2].
В таблице 7.1, показано соотношение уровней моделей *ЭМВОС* и *DoD*.

Таблица 7.1. - Сопоставление уровней моделей ISO и DoD

Модель ISO	***Модель DoD***
7. Прикладной уровень 6. Уровень представления 5. Сеансовый уровень	I. Уровень приложений
4. Транспортный уровень	II. Транспортный уровень
3. Сетевой уровень	III. Межсетевой уровень
2. Канальный уровень 1. Физический уровень	IV. Уровень доступа к сети

Хотя модель ЭМВОС имела достаточно детальную проработку, подкрепленную международными стандартами, она *не удовлетворила всех возложенных на нее ожиданий*.
Многие претензии, предъявляемые к ней, являются противоречивыми и отражают различные объективные и субъективные проблемы, накопленные в практическом использования локальных и глобальных сетей связи.
Наиболее существенные претензии сводятся к следующему:

- *канальный уровень* недостаточно проработан и требует разделения его на два дополнительных уровня иерархии, которые бы гораздо лучше смогли учесть потребности производителей сетевого оборудования и программистов, пишущих драйвера для операционных систем;
- *претензии к сетевому уровню* вызваны различными концептуальными взглядами на проблемы построения глобальных и локальных сетей;
- *верхние уровни* модели ISO являются слишком абстрактными и не отражают специфику многих сетевых приложений;
- *большое количество* иерархически расположенных уровней снижают эффективность реализации сетевого программного обеспечения операционных систем.

Несмотря на успешную реализацию большинства протоколов модели ISO, перечисленные выше недостатки, ограничили ее применение в основном теоретическими аспектами.
В результате, модель *DoD* обеспечила успешное развитие в большинстве *Интернет-технологий*.

7.2 Компьютерные сети и телекоммуникации

Первоначальная базовая архитектура сетевых приложений, основанная на модели DoD и разработанная к середине 90-х годов, была представлена схемой, показанной на рис. 7.2.

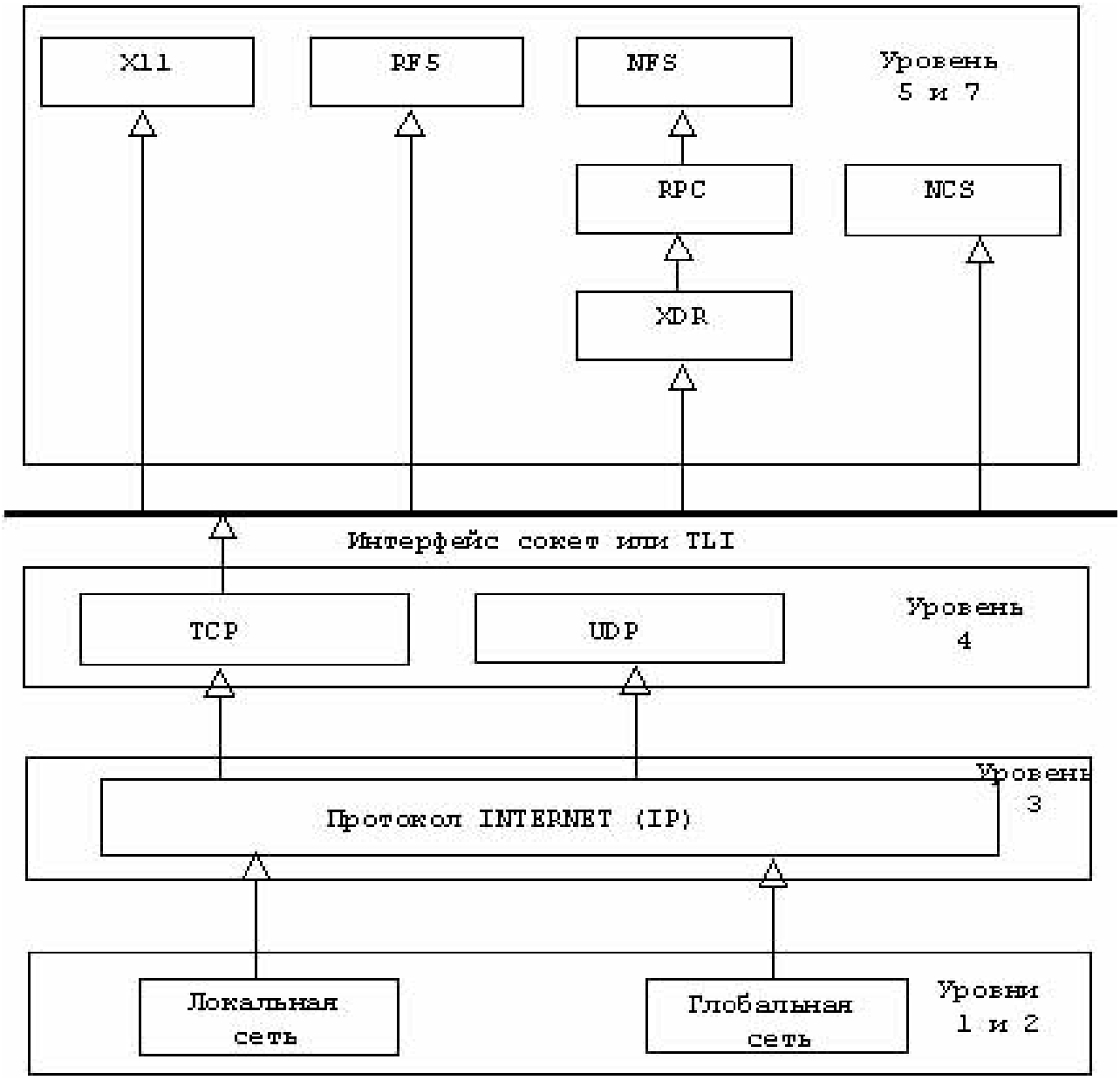

Рисунок 7.2 - Интерпретация базовых сетевых приложений сети Интернет

Основные обозначения этой схемы, отражают базовую многоуровневую иерархию используемых протоколов и приложений:

- *библиотеку сокетов* - интерфейс над протоколами TCP или UDP;
- *библиотеку TLI* (Transport Level Interface) – тоже интерфейс над протоколами TCP или UDP;

- *NFS* (Network File System) - управление распределенными файловыми системами;
- *RFS* (Remote File Sharing) – тоже, что и NFS;
- *X11* - X Window System v.11 - работа с окнами;
- *XDR* (eXternal Data Representation) - представление данных;
- *RPC* (Remote Procedure Call) - вызов удаленных процедур фирмы Sun Microsystems;
- *NCS* (Network Computing System) - вызов удаленных процедур.

В конце 90-х годов, развитие сетевых технологий столкнулось с новой проблемой - разрушительным воздействием на компьютерные системы *программных вирусов* и систем *нарушения авторизации* пользователей (хакерством).
Сети стали угрозой уже существующим компьютерным технологиям.
Все предыдущие сетевые модели не содержали в себе средств противодействия указанным факторам среды.
Как следствие, наряду с расширением сетевой инфраструктуры и общей ориентацией приложений на работу в сети, *многие концепции* сетевого взаимодействия претерпели значительные изменения и стали развиваться в рамках других (прикладных) технологий.

7.3 Интеграция сетевых и объектно-ориентированных технологий

Особое влияние на сетевые технологии оказала автоматизация промышленного производства, развивающаяся в рамках ***локальных сетей***.
Практическим заказчиком изменений выступили системы SCADA, в которых сетевые технологии, в плане взаимодействия открытых систем, выступают основным интегрирующим средством, соединяющим распределенные центры обработки информации.
SCADA - *supervisory control and data acquisition*, - диспетчерское управление и сбор данных.

Новым теоретическим концептом стала идея объектно-ориентированного подхода (ООП) в программировании.
Суть идеи основана на формировании и *именовании сетевых объектов*, взамен старым методам адресации точек приложений в сети.
Такой подход значительно упрощает проектирование, создание и эффективность использования столь сложных и интегрированных АСУ как SCADA-системы.
Реализация и *стандартизация* такого *сетевого объектно-ориентированного подхода* была выполнена в рамках проекта CORBA.
CORBA - *Common Object Request Broker Architecture* - общая архитектура брокера объектных запросов, которая является технологическим стандартом напи-

сания распределенных приложений и продвигается консорциумом (рабочей группой) OMG.

Следует отметить, что первые инструментальные средства для проекта CORBA были реализованы на платформе Java (*технология RMI - Remote Method Invocation*).

Фактически, интеграция сетевых технологий поставила проблему реализации распределенных систем, что можно образно представить рисунком 7.3.

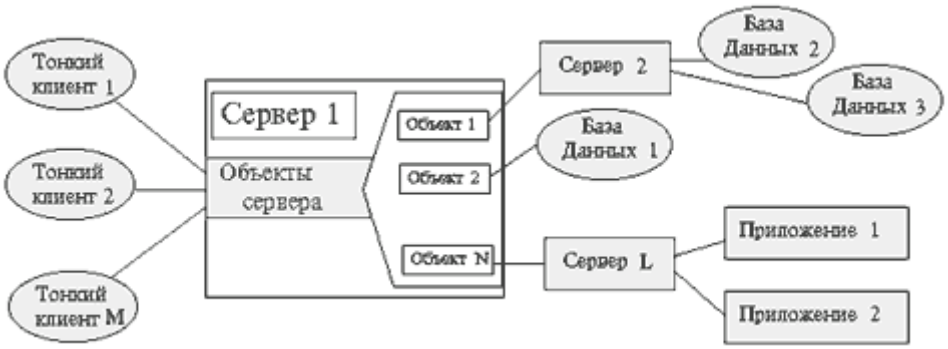

Рисунок 7.3 — Обобщенное представление сетевой интеграции прикладных объектов

На сегодняшний день выделяются три различные технологии, поддерживающие концепцию распределенных объектных систем. Это технологии ***RMI***, ***CORBA*** и ***DCOM***.

7.3.1 Технология RMI

Технология RMI может быть представлена рисунком 7.4.

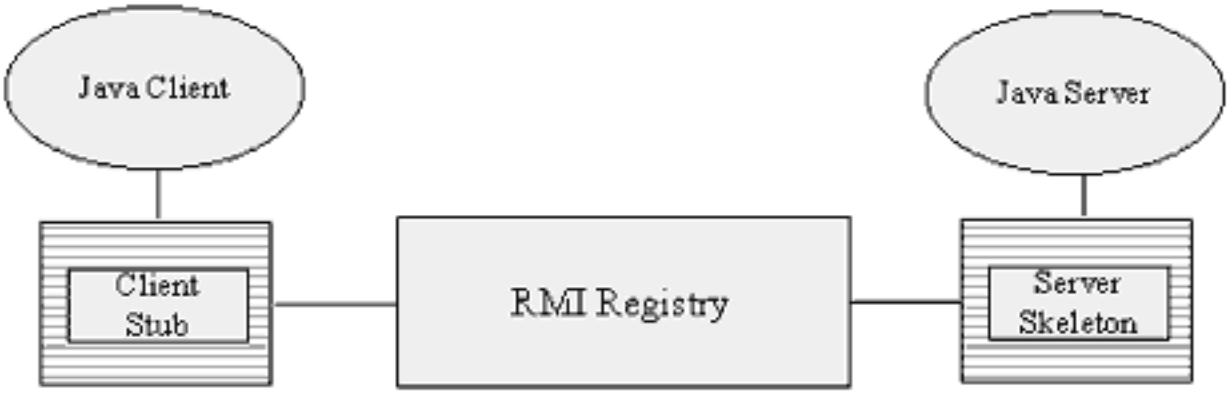

Рисунок 7.4 — Общее представление технологии RMI

Client Stub - переходник для клиента и ***Server Stub*** - переходник для сервера порождены от общего интерфейса, но различие между ними в том, что:

- *client stub* служит просто для подсоединения к *RMI Registry*;
- *server stub* (*Server Skeleton*) используется для связи непосредственно с функциями сервера.

7.3.2 Технология DCOM

Технология DCOM (*Distributed Component Object Model*) была разработана компанией Microsoft в качестве решения для распределенных систем ***в 1996-м году***.
Сейчас, DCOM является главным конкурентом ***CORBA***, хотя контролируется он теперь уже не Microsoft, а группой ***TOG*** (*The Open Group*), аналогичной OMG.
Как технология, DCOM представляет собой расширение архитектуры COM до уровня сетевых приложений, что показано на рисунке 7.5.

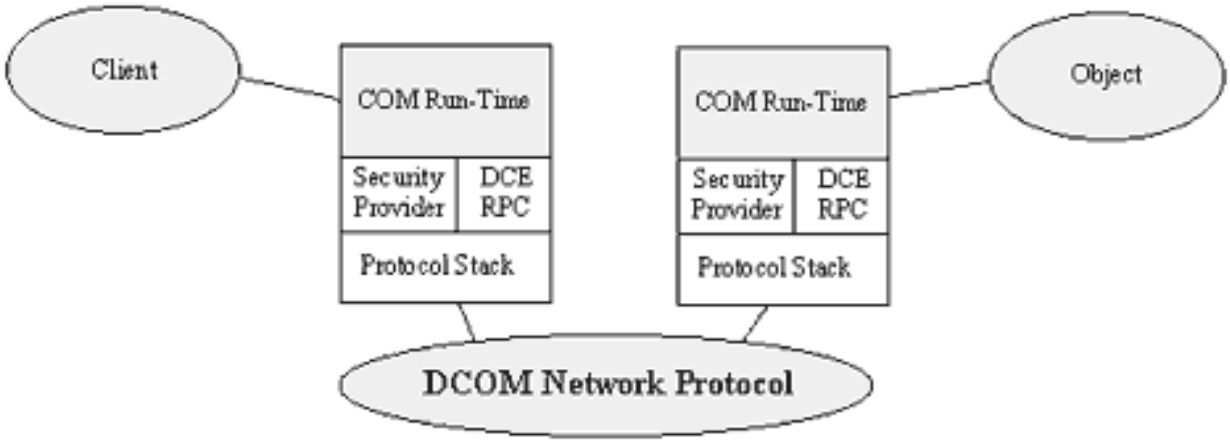

Рисунок 7.5 — Общее представление технологии DCOM

7.3.3 Технология CORBA

Технология CORBA является наиболее обобщенной среди указанных моделей взаимодействия в сети.
Технология CORBA - *Common Object Request Broker Architecture* — разрабатывается ***OMG*** (*Object Managment Group*) ***с 1990-го года***.
Она позволяет вызывать методы у объектов, находящихся в сети где угодно, так, как если бы все они были локальными объектами.

Основная структура CORBA 2.0 ORB показана на рисунке 7.6.

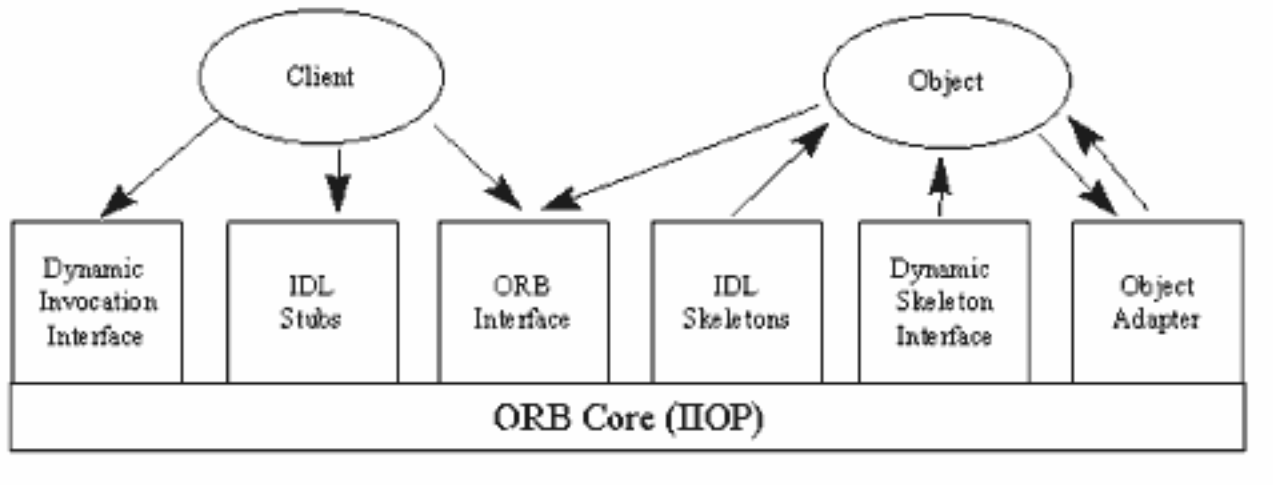

Рисунок 7.6 — Основная структура технологии CORBA

Здесь показаны следующие элементы:

- *Dynamic Invocation Interface (DII):* позволяет клиенту находить сервера и вызывать их методы во время работы системы;
- *IDL Stubs:* определяет, каким образом клиент производит вызов сервера;
- *ORB Interface:* определяет общие как для клиента, так и для сервера сервисы;
- *IDL Skeleton:* обеспечивает статические интерфейсы для объектов определенного типа;
- *Dynamic Skeleton Inerface:* общие интерфейсы для объектов, независимо от их типа, которые не были определены в IDL Skeleton;
- *Object Adapter:* осуществляет коммуникационное взаимодействие между объектом и ORB.

Замечание

Несмотря на существенную значимость сетевых технологий, они в любом случае продолжают играть вспомогательную роль, обеспечивая потредности приложений. Новые, более общие представления о приложения задаются моделью SOA, которая рассматривается в следующей теме.

7.4 Рекомендуемая литература для самостоятельной подготовки

1. ГОСТ 28906-91 Взаимосвязь открытых систем. Базовая эталонная модель ВОС. - http://www.rfcmd.ru/sphider/docs/InfoSec/GOST_28906-91_ISO_7498-84.htm (ГОСТ 28906-91.doc)
2. Бойченко И.В. Программное обеспечение сетей ЭВМ. Учебное пособие. - Томск, 2005. - 294 с. (Бойченко.pdf)

7.5 Вопросы для самостоятельного контроля знаний

1. Каково концептуальное значение модели взаимодействия открытых систем?
2. В чем недостатки и основная критика модели OSI?
3. В чем недостатки и основная критика модели DoD?
4. В чем ограниченность интерпретации базовых сетевых приложений, представленных табл. 7.1?
5. Какие проблемы в настоящее время имеются в плане практического использования архитектур, представленных табл. 7.1?
6. Что такое проект CORBA и в чем его практическое назначение?
7. В чем ограниченность проекта CORBA?
8. Какие известны практические реализации в плане проекта CORBA?
9. Какие известны перспективные направления в плане развития сетевых технологий?
10. Что такое технология RMI и ее концептуальная значимость?
11. Какие известны практические реализации модели DCOM?
12. Какие современные сетевые технологии наиболее актуальны для тематики АСУ?

8 СЕРВИСНЫЕ ТЕХНОЛОГИИ

Рассмотренные ранее технологии, в той или иной степени, были ориентированы на *технические аспекты* применения ЭВМ.
Хотя все элементы вычислительной техники и программное обеспечение тоже стоят денег, а также развивается комптьютерная индустрия, сами результаты работы ЭВМ тоже могут что-то стоить и на этих результатах можно организовывать то, что называется *бизнес*.

8.1 Парадигма сервисных технологий

Успехи применения ЭВМ в различных областях науки и техники, естественным образом, стали формировать парадигму: «*все есть сервис*».

Термин **сервис** прочно вошел в компьютерную терминологию ***в 90-х годах***.
Сервис — *это то, за что можно брать деньги.*
Парадигма сервиса захватила умы широкого круга специалистов.

Все есть сервис:

- Everything as a Service - все как услуга.
- Infrastructure as a service - инфраструктура как услуга.
- Platform as a service - платформа как услуга.
- Software as a service - программное обеспечение как услуга.
- Hardware as a Service - аппаратное обеспечение как услуга.
- Workplace as a Service - рабочее место как услуга.
- Data as a Service - данные как услуга.
- Security as a Service - безопасность как сервис.

Как любая парадигма, *завышенно широкого спектра действия*, концепция сервиса первоначально содержала больше эмоциональности, чем конструктивности.

С другой стороны — *это заявка на будущее*: все, что делается, должно быть бизнесом и приносить деньги.
Возник тезис: «Технологии, которые в перспективе прямо или косвенно не способны делать деньги, обречены на «вымирание».
В поддержку этого тезиса можно перечислить *ряд проблем* развития и сопровождения программного обеспечения.

В 1983 году Билл Гейтс, после выхода версии 2.0 MS DOS, объявил, что фирма Microsoft будет выпускать полностью совместимые версии операционной системы.

Вся последующая история развития как MS DOS, а потом и MS Windows — это не столько развитие компьютерных технологий, сколько *развитие бизнеса* на базе стремления сделать ОС, удовлетворяющую требованиям широкого круга пользователей.
Именно Microsoft опробовала и внедрила *основные технологии* маркетинга, продажи и сопровождения ОС, а также сопутсвующего ОС программного обеспечения.

С развитием Интернет-технологий, появился термин - *software on-demand*: *программное обеспечение по требованию*.
В дальнейшем **SoD** стали трактовать как **SaaS** — *software as a service*.

Общая тенденция всех этих сервисных движений ведет к давней мечте всех капиталистов — *аутсорсингу*.

Аутсорсинг - *outsourcing*: *outer-source-using* - использование внешнего источника/ресурса.
Аутсорсинг - передача организацией на основании договора определённых *бизнес-процессов* или *производственных функций* на обслуживание другой компании, специализирующейся в соответствующей области.

В отличие от услуг сервиса и поддержки, имеющих разовый, эпизодический, случайный характер и ограниченных началом и концом, на аутсорсинг передаются обычно функции по профессиональной поддержке *бесперебойной работоспособности* отдельных систем и инфраструктуры на основе длительного контракта (не менее 1 года).

Наличие *бизнес-процесса* является отличительной чертой *аутсорсинга* от различных других форм оказания услуг и абонентского обслуживания.

Завершая обсуждение тенденций аутсорсинга и Интернет технологий, можно сказать следующее:

- налицо *тенденции возврата к майнфреймам*, обеспечивающим массовое обслуживание сервисов на коммерческой основе;
- *превращение клиентских персональных компьютеров* в сетевые графические станции, обеспечивающие интеррактивное взаимодействие с пользователем.

Опуская ряд важных вопросов, например, «*Специализацию глобальных сервисов*» и «*Обсуждение организаций, специализирующихся на предоставлении сервиса*» рассмотрим:

- *модель SOA*, как теоретическую базу модели сервиса;
- *www-технологии*, как основную технологическую основу реализации модели сервиса;
- *облачные технологии*, как комерческую площадку сервисов.

8.2 WWW-технологии и проект SOA

Сервисно-ориентированная архитектура (*service-oriented architecture*, *SOA*) — подход к разработке программного обеспечения, в основе которого лежат *сервисы со стандартизированными интерфейсами*.
На рисунке 8.1 показаны основные компоненты интерфейсов SOA.

Рисунок 8.1 — Основные интерфейсы SOA

С помощью SOA реализуются *три аспекта ИТ-сервисов*, каждый из которых способствует получению максимальной отдачи от ИТ в бизнесе:

- *Сервисы бизнес-функций.* Суть этих сервисов заключается в автоматизации компонентов конкретных бизнес-функций, необходимых потребителю.
- *Сервисы инфраструктуры.* Данные сервисы выполняют проводящую функцию, посредством платформы, через которую поставляются сервисы бизнес-функций.

- *Сервисы жизненного цикла.* Эти сервисы являются своего рода «обёрткой», которая в большинстве случаев поставляет ИТ-пользователям «*настоящие сервисы*». Сервисы жизненного цикла отвечают за дизайн, внедрение, управление, изменение сервисов инфраструктуры и бизнес-функций.

В общем случае, модель SOA предоставляет *более высокий уровень функциональности*, чем объектно-ориентированный подход, что хорошо показано на рисунке 8.2.

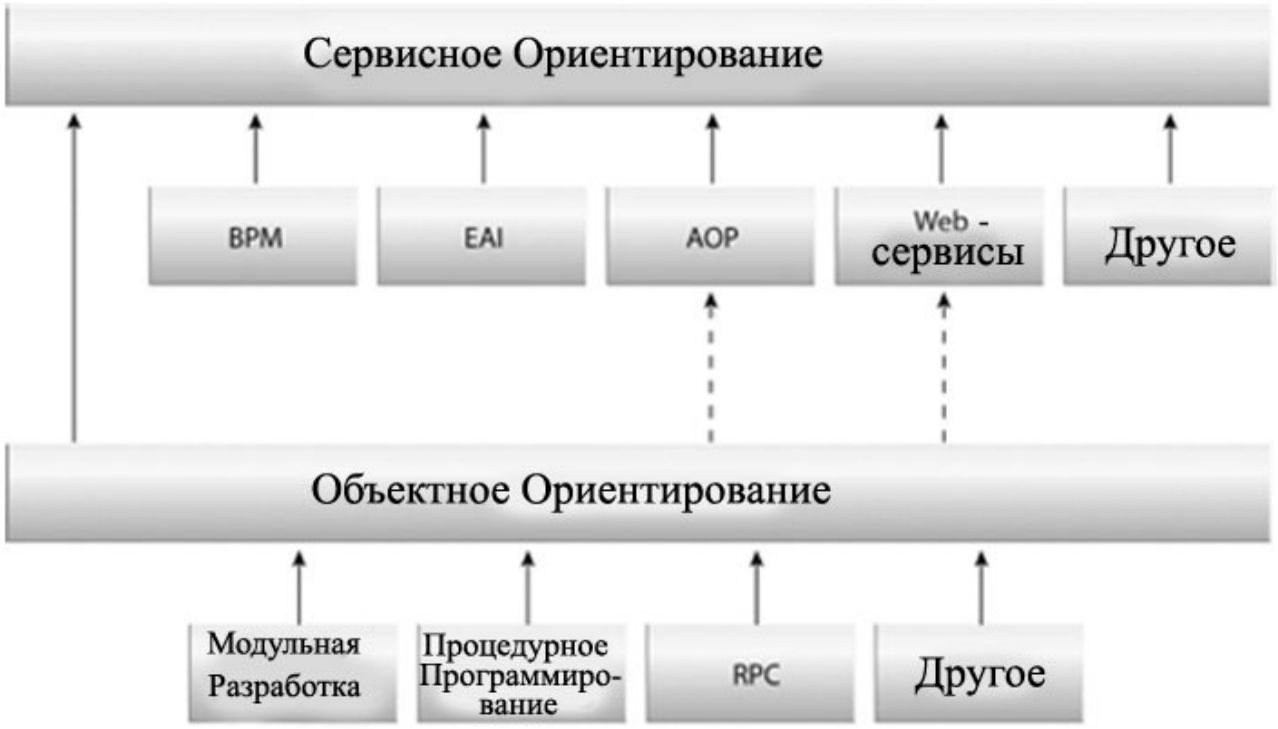

Рисунок 8.2 — Соотношение парадигм SOA и ООП

Появление *сервисно-ориентированного подхода* произвело очередную реформу в теории разработки программного обеспечения, *оставив в прошлом концепцию ООП*.
Как известно, повторное использование программного кода упрощает разработку больших информационных систем.
До недавнего времени, с этой целью традиционно применялся объектно-ориентированный подход, подразумевающий жёсткое объединение компонентов и объектов приложения в одно целое.
В парадигме ООП от разработчика требуется *знание прикладного программного интерфейса*, в котором объединены атрибуты и методы, сообща реализующие необходимый функционал. Но поскольку объектные системы обычно создаются

на основе какого-то одного языка программирования (Delphi, C++, C#, Java и других) и фиксированных механизмов обмена информацией между объектами и модулями информационной системы, то и в ООП сохраняются все зависимости и ограничения.
Такой подход удобен не всегда - в частности, он не позволяет оперативно реагировать на изменение ситуации и, к примеру, проектировать новомодные системы, опирающиеся на концепцию «*ресурсы по требованию*».
Кроме того, для модификации объектных систем нередко приходится переписывать коды связанных объектов и методов.

Свести эти ограничения к минимуму позволяет технология SOA, которая многими уже признана как революция в технологии программирования.

Модель SOA имеет и *свои недостатки*: *проблемы обнаружения и поиска нужного провайдера сервиса*.
Эти проблемы можно сформулировать в виде *двух вопросов*:

- *Как потребителю найти провайдера* службы, которую он хочет вызвать?
- *Как потребитель может быстро и надежно вызвать* службу в медленной и ненадежной сети?

Существует прямое решение обеих этих проблем – подход, называемый *ESB*.

ESB - *Enterprise Service Bus* – сервисная шина предприятия.
Учитывая, что основу реализации моделей SOA составляют www-технологии, был разработан специфльный протокол взаимодействия этих моделей — *SOAP*.

SOAP - *Simple Object Access Protocol* — простой протокол доступа к объектам.
Фактичестки, - это протокол обмена структурированными сообщениями в распределённой вычислительной среде.
Для реализации SOAP используется технология «*SOAP over HTTP*».
Суть этой технологии составляют три варианта взаимодействия потредителя и провайдела сервиса:

- *Синхронный прямой* вызов;
- *Синхронный вызов через* посредника (брокера);
- *Асинхронный вызов через* посредника (брокера).

8.2.1 Синхронный прямой вызов

Метод прямого вызова Web-службы "SOAP over HTTP" предполагает *обращение потребителя сервиса*, к некоторой службе, которая должна дать ему *адрес провайдера сервиса.*
Для решения этой проблемы существует *спецификация UDDI*.

UDDI - *Universal Description Discovery and Integration*, описывающий Web-

службу, которая является каталогом для поиска других Web-служб.
Идея заключается в развертывании UDDI-службы по хорошо известному потребителю адресу; потребитель может использовать UDDI для поиска других Web-служб.
Взаимодействие потребителя, UDDI-службы и провайдеров сервиса котировок показаны на рисунке 8.3.

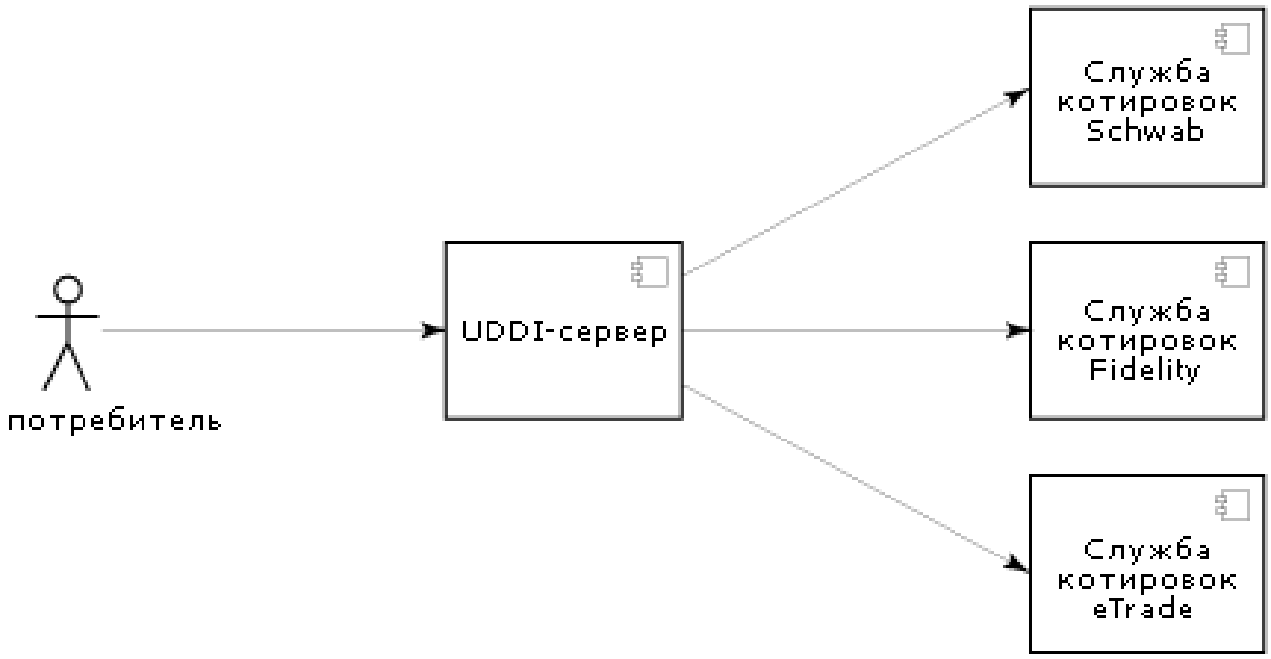

Рисунок 8.3 — Прямой вызов службы через UDDI

Получив адрес, потребитель самостоятельно обращается к провайдеру сервиса.

Недостатки прямого вызова можно сформулировать так:

- *необходимость знания URI оконечной точки* провайдера потребителем для вызова службы. Он использует UDDI как каталог для поиска этого URI.
- *если имеется несколько провайдеров*, UDDI содержит несколько URI, и потребитель должен выбрать один из них.
- *если провайдер меняет URI оконечной точки*, он должен повторно зарегистрироваться на сервере UDDI, для того чтобы UDDI хранил новый URI. Потребитель должен повторно запросить UDDI для получения нового URI.

В сущности это означает, что каждый раз, когда потребитель хочет вызвать службу, он должен запросить в UDDI URI оконечных точек и выбрать один из них. Это ведет к затратам потребителем значительных усилий при периодических операциях запроса в UDDI и выбора провайдера.
Этот подход также *вынуждает потребителя выбирать провайдера* каким-либо способом, по всей видимости, из эквивалентного списка.

8.2.2 Синхронный вызов через посредника

Одним из способов упрощения проблемы является *введение брокера*, показанного на рисунке 8.4, который работает как промежуточное звено при вызове Web-службы.

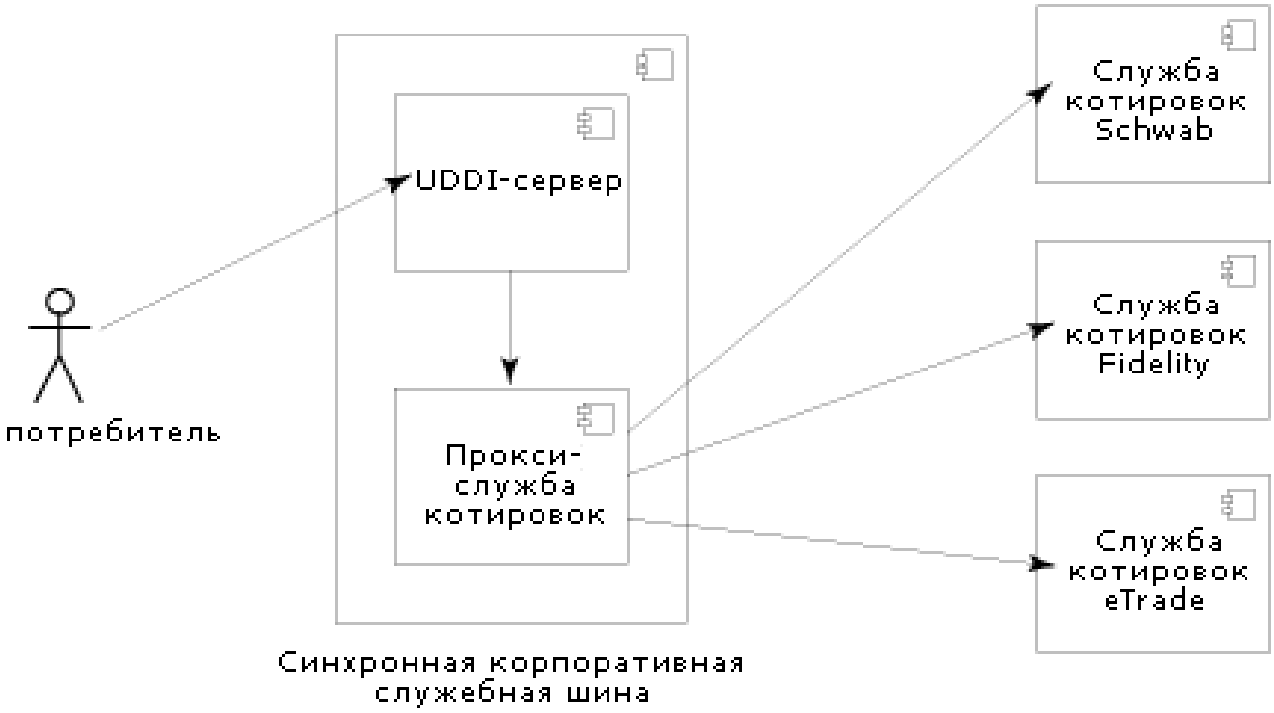

Рисунок 8.4 — Синхронный вызов через брокера

Потребитель вызывает прокси-службу брокера, который, в свою очередь, вызывает службу провайдера.
UDDI возвращает только один URI, и потребитель не должен делать выбор.
Потребитель может даже и не знать, что оконечная точка является прокси-службой; он знает только о том, что может использовать этот URI для вызова Web-службы. Брокер связывает потребителя с провайдерами служб.

Недостатком синхронного подхода:

- Потребитель должен ждать окончания выполнения службы – *поток должен быть заблокирован на время работы службы*.
- Если служба выполняется длительное время, *потребитель может прекратить ожидание ответа*.
- Когда потребитель *выполняет запрос, но не может ждать*.
- Когда у потребителя *возникает аварийная ситуация во время блокировки работы*, даже после перезапуска ответ будет потерян и вызов нужно бу-

дет повторить.

8.2.3 Асинхронный вызов через посредника

Общим решением, указанных выше проблем, является *вызов службы потребителем в асинхронном режиме*, который показан на рисунке 8.5.

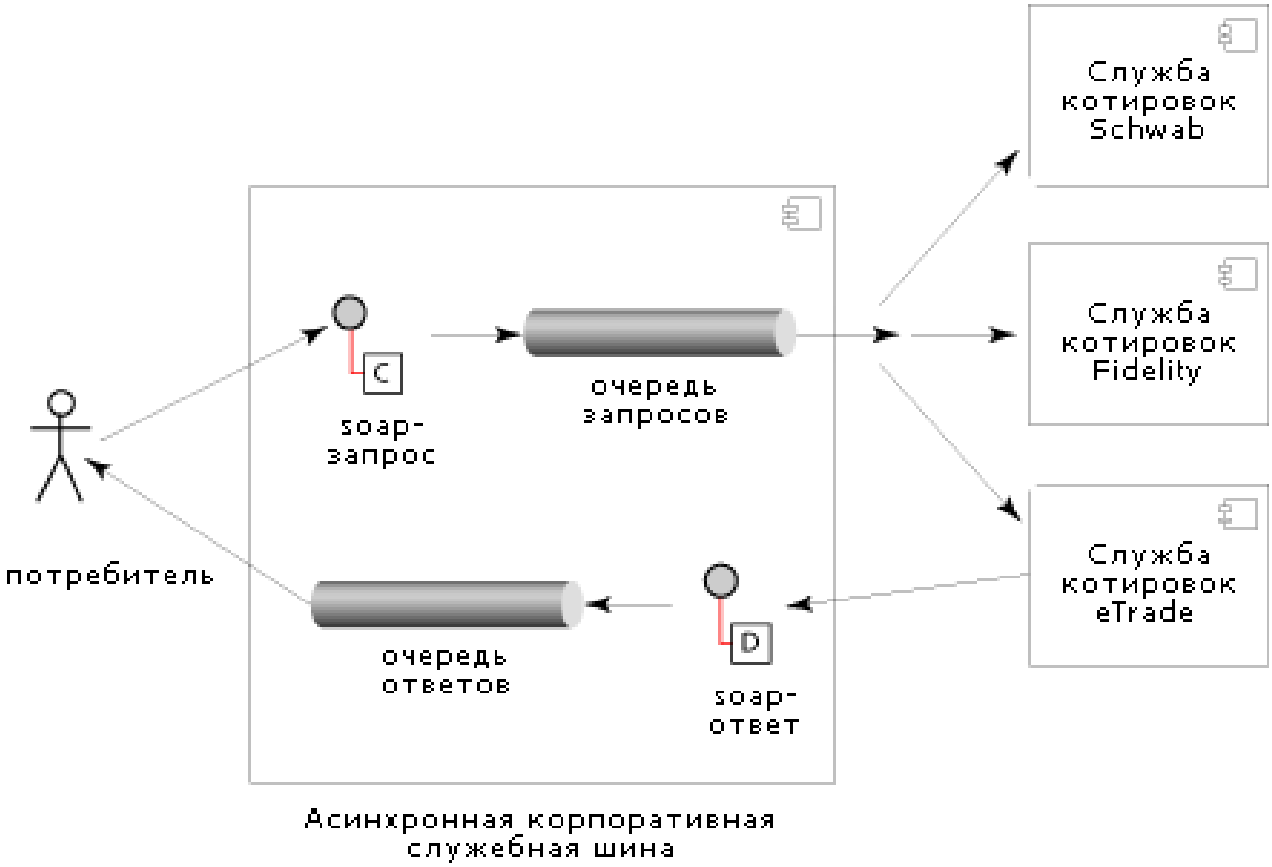

Рисунок 8.5 — Работа через брокера в асинхронном режиме

При таком подходе потребитель использует один поток для передачи запроса и второй для получения ответа:

- Потребитель *не должен блокировать работу* при ожидании ответа и может в это время выполнять другую работу. Следовательно, потребитель намного менее чувствителен к продолжительности работы службы у провайдера.
- Брокер, предоставляющий возможность потребителю вызывать Web-службу асинхронно, *реализуется при помощи системы обмена сообщениями*, которая использует очереди сообщений для передачи запроса и получения ответа.
- Аналогично синхронной прокси-службе *пара очередей сообщений высту-*

пает как один адрес, использующийся потребителем для вызова службы независимо от количества возможных прослушиваемых провайдеров.

8.3 Облачные вычисления и «виртуализация»

Концепция облачных вычислений является одной из новомодных современных концепций компьютерных технологий является, так называемый, *cloud computing*.

Облачные вычисления - *cloud computing* - технология распределенной обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как *Интернет-сервис*.
Термин «Облако» используется как метафора, основанная на изображении *сложной инфраструктуры*, за которой скрываются все технические детали.

Документы IEEE, опубликованные в 2008 году, указывают, что «Облачная обработка данных — *это парадигма*, в рамках которой информация постоянно хранится на серверах в Интернет и *временно кэшируется на клиентской стороне*, например, на персональных компьютерах, игровых приставках, ноутбуках, смартфонах и тому подобных устройствах».

Для обеспечения согласованной работы ЭВМ, которые предоставляют услугу облачных вычислений, используется специализированное ПО, обобщенно называющееся «*middleware control*».
Это ПО обеспечивает:

- *мониторинг* состояния оборудования,
- *балансировку* нагрузки,
- *обеспечение ресурсов* для решения задачи.

Для облачных вычислений *основным предположением* является *неравномерность запроса ресурсов* со стороны клиента(ов).
Для сглаживания этой неравномерности для предоставления сервиса между реальным железом и *middleware* помещается ещё один слой - *виртуализация серверов*.
Серверы, выполняющие приложения, виртуализируются и балансировка нагрузки осуществляется как средствами ПО, так и средствами распределения виртуальных серверов по реальным, соответственно порождая различные *модели развертывания*.

8.3.1 Частное облако

Частное облако - *private cloud* - инфраструктура, предназначенная для исполь-

зования одной организацией, включающей несколько потребителей, например, подразделений клиентами или подрядчиками данной организации.
Частное облако может находиться в собственности, управлении и эксплуатации как самой организации, так и третьей стороны или какой-либо их комбинации, а также может физически существовать как внутри так и вне юрисдикции владельца.

8.3.2 Публичное облако

Публичное облако - *public cloud* - инфраструктура, предназначенная для свободного использования широкой публикой.

Публичное облако может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций или какой-либо их комбинации.

Публичное облако *физически существует в юрисдикции владельца* - поставщика услуг.

8.3.3 Гибридное облако

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

8.3.4 Общественное облако

Общественное облако - *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 может физически существовать как внутри так и вне юрисдикции владельца.

8.4 Рекомендуемая литература для самостоятельной подготовки

1. SOA: плюсы и минусы. - http://samag.ru/archive/article/892.
2. Моделирование SOA: Часть 1. Идентификация сервисов. - http://cmcons.com/articles/soa_i_web-servisy/modelirovanie_soa_chast_1_identifikatsija_servisov/.
3. SOAP Версия 1.2 Часть 0: Учебник для начинающих. - http://www.w3.org/2002/07/soap-translation/russian/part0.html.
4.

8.5 Вопросы для самостоятельного контроля знаний

В разработке.

9 ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

9.1 Интеллектуальные информационные технологии

В разработке.

9.2 Системы искусственного интеллекта

В разработке.

9.3 Робототехника

В разработке.

9.4 Рекомендуемая литература для самостоятельной подготовки

В разработке.

9.5 Вопросы для самостоятельного контроля знаний

В разработке.

10.10 Список тем для курсового проектирования

С целью методического обеспечения по дисциплине «Современные компьютерные технологии», обучающимся предлагается следующий краткий перечень тем для курсового проектирования.

№	Тематика курсового проекта (работы)
1	Современные компьютерные технологии создания интегрированных систем научных и инженерных расчетов. Компонентное создание систем на примерах систем Mathematica, Maple, Mathcad, MatLab, Simulink.
2	Технология обработки данных ADO.NET.
3	Технология СУБД Oracle.
4	Технология СУБД MS SQL Server.
5	Технология СУБД MySQL.
6	Объектно-ориентированные технологии Java.
7	Объектно-ориентированные технологии .NET.
8	Компонентное программирование среды Eclipse.
9	Офисные технологии. Стандарт ODF.
10	Офисные технологии. Система openOffice (libreOffice).
11	Офисные технологии. Система MS Office.
12	Офисные технологии. Интеграция СУБД и openOffice (libreOffice).
13	CALS-технологии в промышленности.
14	Стандартизация ООП в проекте CORBA.
15	Стандартизация сервисных технологий в проекте SOA.
16	Технология промышленной шины ESB.
17	Сервисные технологии SaaS и «Облачные вычисления».
22	Технологии поисковых систем.
23	Технологии мультимедиа.

Учебное издание

Резник Виталий Григорьевич

СОВРЕМЕННЫЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

Учебное пособие предназначено для изучения магистрами второго года обучения теоретического материала в формате лекций, а также самостоятельной и индивидуальной работы по дисциплине «Современные компьютерные технологии» на уровне основной образовательной программы магистратура направления подготовки 01.04.02 «Прикладная математика и информатика» профиля «Математическое и программное обеспечение вычислительных комплексов и компьютерных сетей».

Учебное пособие

Усл. печ. л. 11,55. Тираж ___. Заказ .

Томский государственный университет

систем управления и радиоэлектроники

634050, г. Томск, пр. Ленина, 40