МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Дагестанский Государственный Технический Университет»

ВВЕДЕНИЕ В СОВРЕМЕННЫЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

Гаджимахадова Л.М., Исабекова Т.И.

Учебное пособие

2021

Оглавление

Введение

Компьютерные технологии (КТ) стали достаточно давно и широко применяться в современном обществе. Сложно назвать область деятельности, где компьютеры не применялись бы совсем. Соответственно обучение в вузах также интенсивно использует различные технологии. Практически все дисциплины в технических вузах могут изучаться в компьютерных классах или, другими словами, использовать ту или иную КТ.

Столь широкие представления о понятии КТ, особенно в современном контексте, размывают описания и трактовку самих технологий, делают их обыденными и бессодержательными. Это вынуждает использовать отдельные частные толкования предмета, дополнительно модифицированные в рамках отдельных изучаемых дисциплин. Результатом такого когнитивного процесса является формирование у студентов и аспирантов разрозненных представлений, которые формируются вокруг отдельных и наиболее ярких корпоративных технологических достижений. В качестве следствия, у будущих специалистов наблюдается снижение адекватности восприятия современных технологических достижений и увеличение подверженности влиянию текущих рекламных воздействий.

Целью данного учебного пособия является формирование у обучающихся целостного и адекватного представления о современных компьютерных технологиях (СКТ), которые уже были получены ими в процессе изучения отдельных дисциплин по направлению «Информатика и вычислительная техника», обеспечиваемых учебным процессом кафедры.

В когнитивном плане, указанная цель достигается формированием у обучающихся компетенции, формулируемой как: способность разрабатывать специальное математическое и программное обеспечение систем управления и обработки информации, механизмов принятия решений в следующих областях профессиональной деятельности: информация и информационные системы, экономика, энергетика, промышленность, образование.

В методическом плане, указанная цель достигается последовательностью изложения учебного материала, в основе которой лежит принцип обсуждения существующих КТ в процессе их формирования хорошо известными этапами развития аппаратных средств вычислительной техники. Именно такой подход позволяет наглядно продемонстрировать идейные парадигмы существующих технологий, их органическую связь и направленность развития. В результате такого подхода и с учетом направленности образования, весь учебный материал удается изложить в последовательности девяти разделов, первый из которых более подробно обосновывает выбранный подход, а также группирует остальные разделы в хорошо известные технологические направления.

Второй раздел посвящен обсуждению вычислительных технологий, знаменующих начало новой «цифровой» эры развития общества и являющихся до настоящего времени базовой КТ проникающей во все другие технологии.

Третий раздел пособия описывает технологии, которые существенно расширили горизонты возможного применения ЭВМ: технологий хранения цифровой информации, дополнивших недостатки вычислительных технологий и создавших основу актуальности и жизнеспособности научному направлению, известному как «Информатика».

Четвертый раздел дает описание объектно-ориентированных технологий, которые синтезировали достижения предыдущих КТ в новое качество, обеспечив создание информационных объектов как продукта, который можно производить, продавать и потреблять.

Пятый раздел дает дает представление об офисных технологиях, обеспечивших доступ КТ широким слоям общества.

Шестой и седьмой разделы посвящены технологиям автоматизации производства и технологиям взаимодействия открытых систем. Их можно изучать в любом порядке, поскольку несмотря на различную направленность они очень тесно связаны в плане дополнения друг друга.

Восьмой раздел содержит описание нового технологического уровня, названного как сервисные технологии. На указанном уровне объектами производства выступают уже не отдельные информационные объекты, а целые информационные технологии.

Девятый раздел завершает учебный материал дисциплины. Он посвящен интеллектуальным системам и технологиям, которые, хотя и существуют как официальное научное направление, фактически обозначают все технологии будущего.

В целом, несмотря на указанную выше взаимосвязь выделенных групп технологий, каждая из них на протяжении многих лет продолжает сохранять свою специфическую уникальность и практическую автономность. Это вызвано разными причинами, которые еще требуют своего анализа, поэтому изложение учебного материала каждого раздела не изобилует взаимными ссылками. По этой же причине, каждый раздел имеет свой список использованной литературы.

1 Текущее состояние компьютерных технологий

Образовательное направление «Информатика и вычислительная техника», уточняемое специализацией «Системный анализ, управление и обработка информации (информация и информационные системы, экономика, энергетика, промышленность, образование)», достаточно точно соотносит пропорции внимания, которые должны быть уделены познавательным объектам обозначаемым как программное обеспечение и ЭВМ. В этом контексте, основная направленность обучения касается описания технологий, реализованных в программном обеспечении ЭВМ. Сами же ЭВМ рассматриваются как базовая исполнительная часть программного обеспечения, дающая нужный технологический эффект.

Высказанное мнение вполне укладывается в общие представления о современных компьютерных технологиях (СКТ), отводя самим ЭВМ унитарную вспомогательную роль исполнения. Это также подтверждается текущим состоянием технологий, когда приложения, например текстовые редакторы, работают на разных платформах ЭВМ.

С другой стороны, вполне понятно, что, если убрать исполнительную часть (ЭВМ), то не будет смысла говорить ни о программном обеспечении, ни о компьютерных технологиях. Более того, было время, когда ЭВМ не существовали. Соответственно, не существовали компьютерные технологии. Тем не менее, существовала деятельность, которая касалась системного анализа, управления и обработки информации.

Таким образом, для полноценного раскрытия тематики обучения, следует рассмотреть предмет изучения в трех плоскостях:

- идейные парадигмы обработки информации;
- этапы развития аппаратных средств ЭВМ;
- известные парадигмы компьютерных технологий.

Учитывая, что парадигмы (модели, образцы) представляют собой совокупность фундаментальных научных установок, представлений и терминов, принимаемых и разделяемых научными сообществами, можно будет перейти к обоснованию разделения компьютерных технологий на хорошо узнаваемые группы. Это должно обеспечить обоснованность разделения учебного материала изучаемой дисциплины, а также обозначить ту нить изложения, которая свяжет выделенные группы технологий.

1.1 Идейные парадигмы обработки данных

Вопросам назначения и использования ЭВМ, а также прогнозам их развития, посвящено множество публикаций, которые в разной степени раскрывают указанную тему.

В одном из написанных еще во времена СССР учебников для вузов [1] группой авторов Ларионовым А.М., Майоровым С.А. и Новиковым Г.И. для студентов, обучающихся по специальности «Электронные вычислительные машины», дано определение понятию «Системы обработки данных». Дословно, это определение звучит так: «Система обработки данных (СОД) – совокупность технических средств и программного обеспечения, предназначенная для информационного обслуживания пользователей и технических объектов». Далее, проводя анализ возможностей применения СОД, авторы выделяют два класса систем:

- ***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. Полная классификация СОД показана на рисунке 1.1.

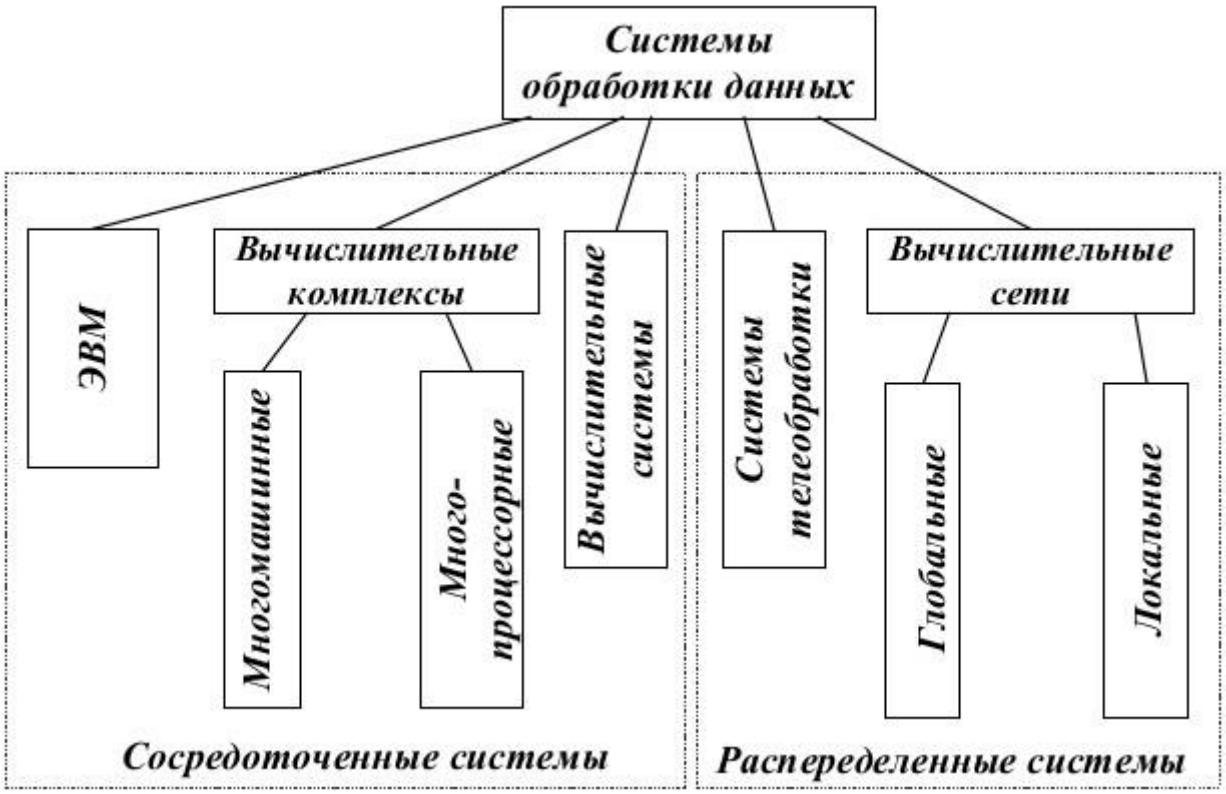

Рисунок 1.1 - Структура систем обработки данных

Важность, если не гениальность, приведенной классификации состоит в том, что в ней выделены главные признаки, по которым можно классифицировать все компьютерные технологии, выделяя главную идейную основу их применения. Раскроем более подробно выделенные на рисунке 1.1 понятия.

Сосредоточенные системы охватывают класс вычислительной техники, рассматривающий каждый его элемет как отдельное изделие. Каждое такое изделие

характеризуется своей целостностью как в концептуальном плане, так и в плане его реализации.

ЭВМ — отдельное типовое решение, соответствующее общепринятому определению, включающему в себя аппаратную часть и системное программное обеспечение, понимаемое как операционная система ОС.

Вычислительные комплексы (***ВК***) — это изделия на базе средств вычислительной техники, которые уточняются по двум классификационным признакам: *многомашинные* и *многопроцессорные*.

Многомашинные ВК — изделия, компоненты которых можно рассматривать как отдельные ЭВМ, необязательно одной платформы.

Замечание

В состав многомашинных ВК, кроме отдельных ОС, включается также программное обеспечение, объединяющее ВК в единое целое.

Многопроцессорные ВК — системы, включающие более одного процессора, использующего общую память (основную память).

Замечание

Многие современные компьютеры имеют многоядерные архитектуры процессора. С точки зрения рассматриваемой классификации, каждое ядро учитывается как отдельный процессор.

Вычислительная система (***ВС***) — ВК, на котором установлено одно или несколько прикладных программных систем. Соответственно, характеристики ВС рассматриваются как набор характеристик ВК и характеристик прикладного программного обеспечения.

Распределенные системы добавляют новое технологическое качество — взаимодействие через коммуникационную среду.

Системы телеобработки (***СТ***) — разделяют вычислительные системы минимум на две части, обеспечивая последующее их взаимодействие на базе соединения «*точка-точка*».

Вычислительные сети — технологически развивают взаимодействие ВС (вычислительных систем), вводя понятие *сетевого адреса*, и снимают ограничение на само взаимодействие, обеспечивая соединения «*один-комногим*» и «*многие-ко-многим*». Дополнительно, классификация *вычислительных сетей* уточняется по признакам: *локальные* и *глобальные*.

Локальные вычислительные сети характеризуются тем, что каждая из них имеет единую политику сетевого взаимодействия ЭВМ, которая также отражается на каждой ВС, входящей в такую сеть.

Замечание

Пространственная протяженность локальной сети, хотя и влияет на всю систему, в смысле масштабирования, но главным является понятие автономной системы, определяемой единым протоколом обмена данных.

Глобальные вычислительные сети снимают ограничения, накладываемое понятием *автономной системы*. В плане взаимодействия такие системы характеризуются тем, что для каждой отдельной ВС уже недостаточно одного протокола *внутренней маршрутеризаци* и должно поддерживаться множество протоколов *внешней маршрутеризации*.

Таким образом, модель СОД предоставляет нам набор достаточно узнаваемых и ярких признаков, которые обязательно должны учитываться при рассмотрении компьютерных технологий.

1.2 Этапы развития аппаратных средств ЭВМ

В 1964 году, хорошо известная корпорация IBM выпустила серию компьютеров IBM 360, обеспечив ее технологическое применение хорошим описанием, ставшим образцом на многие годы. Кроме того, IBM объявила эту серию - «*компьютеры третьего поколения*», что само по себе вызвало общественный интерес и стимулировало исследования посвященные поколениям ЭВМ.

Замечание
Здесь и далее, термины вычислительная машина, а также общепринятые сокращения ВМ и ЭВМ, понимаются как синонимы и не уточняются определениями.

Долгие годы «поколения ЭВМ» изучались, классифицировались и уточнялись их исторические границы. Ценность таких исследований имеет не только историческое значение, поскольку одновременно проходило обсуждение различных достижений компьютерных технологий, а также проводились прогнозы их будущего развития и использования. Имеется ГОСТ 15971, дающий следующее определение: «Поколение вычислительных машин — это классификационная группа ВМ, объединяющая ВМ по используемой технологии ее устройств, а также по уровню развития функциональных свойств и программного обеспечения и характеризующая определенный период в развитии промышленности средств вычислительной техники».

Достаточно подробное описание поколений ЭВМ можно найти в учебнике. Здесь же проводится их краткое изложение с целью выделения наиболее значимых классификационных признаков, по которым строится изложение учебного материала данного пособия.

1.2.1 Нулевое поколение (1492-1945)

Это поколение имеет еще и обобщенное название «механическая» эра, поскольку сюда относят механические, а позднее, и электромехнические вычислительные устройства.

1.2.2 Первое поколение (1937,1945 -1953,1955)

Основной признак этого поколения — построение вычислительных машин с использованием схем на базе электронно-вакуумных ламп, вместо электромеханических реле. С этого времени появляется возможность говорить об электронно-вычислительных машинах (или ЭВМ).

Не вдаваясь в детали первенства тех или иных релизаций ЭВМ, а также о преимуществах электронных переключателей над электромеханическими реле, отметим конкуренцию двух архитектурных решений: *гарвардской архитектуры* и *архитектуры фон Неймана*.

Гарвардская архитектура *ЭВМ* была реализована в первом американском компьютере Mark 1, под руководством инженера IBM Говарда Эйкена, и продемонстрирована реководству IBM в 1941 году. Сама идея архитектуры появилась в Гарвардском универсистеме (в октябре 1937 года), в честь которого и получила свое название. Основными отличительными признаками такой архитектуры являются:

- хранилище инструкций и хранилище данных представлены разными физическими устройствами;
- канал инструкций и канал данных - так же физически разделены.

Фон-неймановская архитектура ЭВМ, показанная на рисунке 1.2, принадлежит Присонскому универсистеу. Ее особенностями являются:

- общая и однотипная для команд и данных память ЭВМ (*принцип однородности* памяти);
- *принцип адресности* - память состоит из пронумерованных ячеек с произвольным доступом;
- *принцип программного управления* — АЛУ (алфавитно-логическое устройство) реализует набор команд, на основе которых пишется программа;
- *принцип двоичного кодирования* — в формате команды выделяются два поля: *поле кода операции* и *поле адресов*, которые можно представить в двоичном виде: последовательностью 0 и 1.

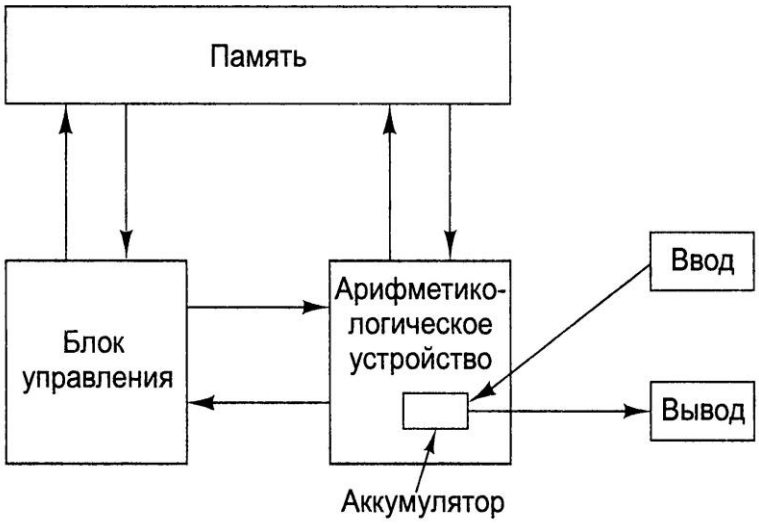

Рисунок 1.2 — Классическая архитектура ЭВМ Фон-Неймана

В конкурсе архитектур победила фон-неймановская идея, которая и была положена в основу последующих реализаций ЭВМ. Эту архитектуру мы и будем подразумевать, обсуждая компьютерные технологии.

Замечание
Гарвардская архитектура не исчезла с горизонта средств вычислительной техники. В настоящее время, она широко реализуется в сигнальных процессорах и ЭВМ специального назначения, типичное прикладное значение которых: быстрое преобразование Фурье и обработка данных в цифровых фильтрах.

В плане обсуждения компьютерных технологий, это поколение ЭВМ также не представляет значительного интереса, поскольку сами технологии только зарождались и, в

основном, находились в стадиях идей и экспериментов. Действительно, технология программирования в данный период находилась на зачаточном уровне, поскольку программы писались в машинных кодах и непосредственно вводились в память ЭВМ. Символьная нотация программ на уровне ассемблера появилась только в начале 50-х годов. Само программное обеспечение фактически не отделялось от аппаратной части компьютера. Соответственно, не разделялось системное и прикладное программное обеспечение.

1.2.3 Второе поколение (1954,1955 -1962,1965)

Второе поколение ЭВМ характеризуется рядом достижений в элементной базе средств вычислительной техники, архитектуре компьютеров и существенном развитии технологии разработки и использования программного обеспечения.

В плане использования элементной базы ЭВМ, осуществлен переход от электронных ламп к полупроводниковым элементам, построенным на диодах и транзисторах. Это позволило существенно повысить надежность аппаратных средств, а также существенно уменьшить время переключения цифровых схем устройств, доведя его до порядка 0,3 мс.

Принципиально важным является переход от устройств памяти на базе ртутных линий задержки к устройствам на магнитных сердечниках. Это не только повысило быстродействие и надежность устройств памяти, но обеспечило произволный доступ к данным, хранимым в ней.

Изменения в структуре ЭВМ связаны с появлением нового класса ЭВМ — мини-компьютеров. Примером такого класса ЭВМ являются вычислительные машины PDP (*Programmed Data Processor*) корпорации DEC (*Digital Equipment Corporation*). Уже, начиная с первого успешного коммерческого проекта (PDP-8, март 1965 года), стала применяться общая шина (*Omnibus*). Соответственно, архитектура такой ЭВМ стала представляться в модульном виде, показанном на рисунке 1.3.

Рисунок 1.3 - Архитектура ЭВМ с общей шиной

Серьезные изменения произошли и в плане программного обеспечения. Появились языки высокого уровня: Fortran (1956 год), Algol (1958 год) и Cobol (1959 год). В результате, программное обеспечение стало отделяться от аппаратных средств ЭВМ. Само программное обеспечение начинает разделятся на системное (супервизорное), инструментальное и прикладное.

1.2.4 Третье поколение (1963,1965 – 1972)

Третье поколение ЭВМ характеризуется резким увеличением вычислительной мощности компьютеров. Как отмечено ранее, корпорация IBM выпустила серию

компьютеров IBM 360, архитектура и программное обеспечение которых стало эталоном для больших универсальных ЭВМ (*mainframes*). Фактически IBM и сформировала первые представления о компьютерных технологиях, представляющих интерес для нашей дисциплины, хотя эти представления и ограничены классом больших машин. Тем не менее, заявив о третьем поколении своих компьютеров, IBM инициировала и интерес к тематике поколений ЭВМ, что существенно продвинуло изыскания в плане стандартизации как текущих, так и будущих технологических решений в области вычислительной техники.

Основные технологические достижения в области аппаратных средств ЭВМ сводятся к следующему:

- переход от дискретных полупроводниковых элементов к интегральным микросхемам;
- начало применения полупроводниковых запоминающих устройств, начинающих вытеснять запоминающие устройтсва (ЗУ) на магнитных сердечниках.

Применительно к архитектуре ЭВМ, проводятся исследования направленные на более эффективное использование возросшей мощности элементной базы, связанное с ее структурной сложностью. В результате появляются:

- *микропроцессоры*, предоставляющие средства конвейеризации и параллельной обработки;
- *микропрограммирование* - как эффективная техника построения устройств управления сложных процессоров.

В области программного обеспечения наблюдается интенсивный поиск новых технологических решений:

- появляются первые операционные системы (ОС);
- реализуются парадигмы разделения времени, обеспечивающие мультипрограммные режимы работы ЭВМ.

В этом плане следует отметить:

- появление первой версии ОС UNIX (1970 год), завершившей многолетние исследования, начатые еще в лаборатории Bell Labs корпорации AT&T;
- создание Кеном Томпсоном языка B, предшественника языка C, ориентированного не только на вычисления, как язык Fortran, но и на написание системного программного обеспечения, такого как ОС.

1.2.5 Четвертое поколение (1972 - 1984)

Четвертое поколение ЭВМ связывается с переходом на интегральные микросхемы большой и сверхбольшой степени интеграции:

- *большая степень интеграции* (large-scale integration, LSI) содержит около 1000 транзисторов на кристалле;
- *сверхбольшая степень интеграции* (very large-scale integration, VLSI) имеет порядка 100000 транзисторов на одном кристалле.

Указанные технические достижения позволили умещать в одной микросхем целый компьютер, включающий центральный процессор (ЦП), основную память и систему ввода/вывода. Как следствие, появились первые персональные компьютеры в виде комплектов. Каждый такой комплект содержал:

- печатную плату;

- набор интегральных схем, обычно включающий схему Intel 8080, несколько кабелей, источник питания и, иногда, 8-дюймовый дисковод;
- программное обеспечение не прилагалось, требовалось писать самому.

Аппаратный технологический прорыв широко открыл двери для прихода программных новинок, что можно назвать началом внедрения современных компьютерных технологий:

- появилась операционная система CP/M (1974 год), написанная Гарри Килдаллом (Gary Kildall) для процессора Intel 8080;
- позднее появились MS-DOC (1981 год), Windows (ноябрь 1985 года), OS/2 (1987 год) и другие;
- в 1981 году правительство Японии объявило о намерениях выделить национальным компаниям 500 миллионов долларов на разработку компьютеров пятого поколения на основе технологий искусственного интеллекта, которые должны были потеснить «тугие на голову» машины четвертого поколения.

Особо следует отметить развитие языков программирования. Именно в это время развиваются языки императивного стиля программирования: C (1974 год), C++ (1983 год). Появляются и развиваются языки сверхвысокого уровня:

- Prolog (Programming in Logic) 1972 год — язык логического программирования, основанный на декларативной парадигме;
- FP (Functional Programming) 1973 год — алгебраический язык программирования, также использующий декларативную парадигму.

Продолжают развитие уже созданные ранее языки Fortran, Cobol и другие.

В целом, этот период очень богат на разные технологические новшества.

1.2.6 Пятое поколение(1984-1990)

Пятое поколение ЭВМ, имеющее достаточно короткий период времени, выделяется масштабом использования процессоров. Появляются компьютеры с сотнями вычислительных устройств, каждый из которых оценивается как отдельный процессор. Это стало побудительным мотивом для прогресса в области параллельных вычислений.

Ранее, параллельные вычисления за счет дополнительных усовершенствований самих процессоров:

- конвейеризации;
- векторной обработки;
- распределения работы между небольшим числом процессоров. Теперь появляются вычислительные системы, обеспечивающие такое распределение задач по множеству процессоров, при котором каждый из процессоров может выполнять задачу отдельного пользователя.

К этому же поколению относят и бурное развитие технологий локальных и глобальных компьютерных сетей. Появляется возможность распределенных вычислений.

1.2.7 Шестое поколение (1990-) – невидимые компьютеры

Шестое поколение ЭВМ завершает рассматриваемую классификацию. Если на ранних стадиях эволюции вычислительных средств смена поколений ассоциировалась с

революционными технологическими прорывами, что и лежит в основе классификации, инициированной корпорацией IBM на уровне третьего поколения, то последующие периоды начинают урачивать этот признак. Действительно:

- каждое из первых четырех поколений имело четко выраженные отличительные признаки и вполне определенные хронологические рамки;
- последующее деление на поколения уже не столь очевидно и может быть понятно лишь при ретроспективном взгляде на развитие вычислительной техники.

Таким образом, пятое и шестое поколения в эволюции ЭВМ — это отражение нового качества, возникшего в результате последовательного накопления частных достижений, главным образом в архитектуре вычислительных систем. С другой стороны, — компьютеры начали стремительно уменьшаться, что побудило научное сообщество выделить последнее (шестое) поколение, не ограничивая его окончательными временными рамками.

Рассмотрим, в качестве демонстрации, ряд достижений, знаменующих начало шестого поколения ЭАМ.

В 1993 году появилась модель Apple Newton, которая наглядно доказала, что компьютер можно уместить в корпусе размером с кассетный плеер.

Основная проблема, которую продемонстрирвало это решение, состояла в усложности «рукописного ввода».

Впоследствии, пользовательский интерфейс подобных машин был изменен, и появилось то, что теперь называется персональными электронными секретарями – PDA.
PDA (***Personal Digital Assistants***) - просто карманный компьютер.

По настоящему революционной технологией настоящего времени, считаются «невидимые» компьютеры, которые встраиваются в бытовую технику, часы, банковские карточки, а также в огромное количество других устройств.

Процессоры этого типа предусматривают широкие функциональные возможности и не менее широкий спектр вариантов применения за весьма умеренную цену.

Вопрос о том, можно ли свести эти микросхемы, появившиеся еще в 1970-х годах, в одно полноценное поколение - остается дискуссионным.

1.3 Известные парадигмы компьютерных технологий

Теория и практика построения ЭВМ и основанных на них технологиях прошла достаточно длительный путь и основана как на удачных практических реализациях, так и на множестве теоретических концепций, в той или иной степени, обосновывающих и направляющих практические решения. Не имея возможности проанализировать все идеи, остановимся на двух концептуальных представлениях, актуальных как в прошлом, так и в настоящем: многоуровневые модели управления и идеи «виртуализации».

1.3.1 Многоуровневые модели управления

Излишне напоминать, что компьютеры являются сложными системами, а, следовательно, они должны иметь тенденцию разбиваться на подсистемы.

В 1959-1961 годы академик В.М. Глушков показал, что в любом устройстве обработки информации можно выделить операционный и управляющий автоматы,

соединенные каналами обратной связи. Это заложило основной теоретический фундамент развития средств вычислительной техники, построенный на математике.

С другой стороны, сформирована парадигма сложной вычислительной машины как иерархии более простых машин, каждая из которых имеет свой язык управления (программирования). Соответственно, результатом программирования является иполняемая программа или интерпретатор.

Таким образом, сформировался принцип модульности, где ЭВМ представляться иерархической многоуровневой системой, состоящей из множества виртуальных машин со своим языком программирования.

Классическая архитектура такой ЭВМ, состоящая из 6 уровней, показана на рисунке 1.4.

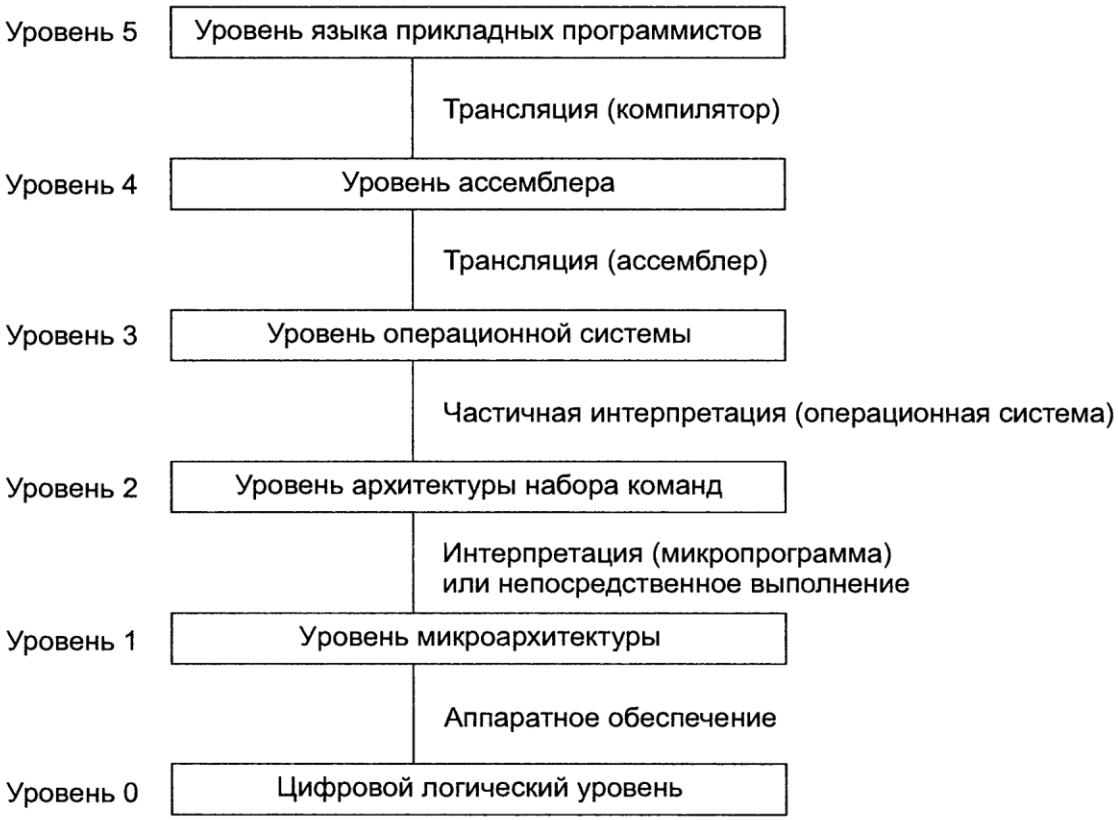

Рисунок 1.4 - Классическая многоуровневая архитектура ЭВМ

Хотя назначение выделенных уровней — достаточно прозрачно, отметим, что три нижних уровня связаны с аппаратной частью ЭВМ и в болшей степени характеризуют процессор:

- Уровень 0 — вентили, преобразующие аналоговые процессы в цифровые, интерпретируемые состояниями 0 и 1 (биты, разряды), а также обеспечивающие операции «И» или «ИЛИ» над ними; соответственно, биты памяти, объединенные в группы, например, 8, 16, 32 или 64, формируют регистры; каждый регистр может содержать одно двоичное число до определенного предела;
- Уровень 1 — микроархитектура процессора;
- Уровень 2 — сам процессор с заданным набором команд.

Верхние три уровня соответствуют программному обеспечению (ПО), в старом классическом его понимании, когда ПО не выделялось из состава ЭВМ.

В современном представлении, верхнии три уровня технологически отделяются от ЭВМ, образуя самостоятельные системы, в которых нижние уровни трактуются как аппаратная платформа. Тем не менее, каждая такая платформа содержит свое программное обеспечение, реализуемое как:

- Firmware — микропрограммы аппаратной платформы;
- BIOS или UEFI — базовое ПО, являющееся составной и неотъемлемой частью ЭВМ.

Продолжая демонстрацию концепции многоуровневых моделей, приведем классическую иерахию ОС UNIX, которая, сверху в низ, представлена уровнями:

- утилиты, системные программы и приложения пользователей;
- интерфейс системных вызовов API;
- менеджеры ресурсов, файловые системы и виртуальная память;
- базовые механизмы ядра; • машинозависимые модули ядра ОС;
- средства поддержки ядра ОС.

Многоуровневая организация ПО стала широко применяться и в сетевых технологиях. Например, хорошо известна 7-уровневая модель взаимодействия открытых систем (ВОС, модель OSI):

- уровень 7 — прикладной;
- уровень 6 — представления;
- уровень 5 — сеансовый;
- уровень 4 — транспортный;
- уровень 3 — сетевой;
- уровень 2 — канальный; • уровень 1 — физический.

В частности, классическая 4-х уровневая модель DoD, которая легла в основу проекта DARPA, также является иерархической и хорошо вписывается в модель OSI.

При внимательном рассмотрении можно было бы еще продолжить список иерархических моделей, которые в той или иной степени соответствуют компьютерным технологиям, например, 3-х уровневая иерархическая модель АСУ содержит:

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

В конечном итоге, можно смело утверждать, что многоуровневое иерархическое представление является общим методологическим приемом при изучении и реализации сложных систем. Следовательно, иерархические схемы и конструкции должны обязательно учитываться в процессе изучения компьютерных технологий.

1.3.2 Идеи «виртуализации»

Другим важным методологическим приемом является «виртуализация» описания и представления средств вычислительной техники применяемой в ЭВМ или СОД. Но, чтобы правильно понимать идею «виртуализации» следует уточнить и осознать контекст применения самого термина.

В общем случае, термин «виртуальный» соответствует термину «модель» или «симулякр», отражая принципиальные ограниченные возможности описания объектов или свойств реального мира.

В более узком смысле, этот термин используется с негативным семантическим значением в противопоставлении к термину «модель». В таком понимании требуется устранение «виртуальности», с целью использования более адекватного представления в виде «модели».

В компьютерных технологиях идея «виртуализации» рассматривается как достаточно общий позитивный прием, направленный на решение конкретных технологических задач. Поясним это рядом конкретных примеров.

Все операционные системы (ОС) используют различные файловые системы (ФС). Все ФС имеют набор общих свойств, например, работа с файлами и директориями, но различную структуру представления на внешних (блочных) устройствах. Доступ к ФС осуществляется через ядро ОС, которое должно обеспечить всем программам все необходимые функции работы с файлами.

Чтобы уменьшить общий размер ядра ОС, используется технология под названием «Виртуальная файловая система», суть которой:

- объединение в ядре общей функциональной части всех ФС;
- реализация оригинального функционала каждой ФС в ПО драйвера.

Другим широко распространенным примером является использование устройства flashUSB в качестве блочного устройства для хранения личных данных.

Хотя физически flashUSB не является блочным устройством, таким как устройство жесткого диска, использование специального ПО (драйверов) позволяет с успехом создавать файловые системы и пользоваться ими.

В современных ОС, вообще стало нормой использование «псевдоустройств», с помощью которых реализуются различные компьютерные технологии. Коме того, сама ОС, в соответсвии с общей концепцией, представляет собой «виртуальную машину», скрывающую все особенности реализации аппаратной части ЭВМ.

1.4 Этапы развития компьютерных технологий

Подводя итог уже изложенному учебному материалу, несложно заметить, что имеются некоторые общие тенденции развития компьютерных технологий, которые можно, а следовательно и нужно, рассматривать в трех проекциях:

1. Исторический аспект изменения технологий, отраженный поколениями развития аппаратных средств ЭВМ;
2. Модульный аспект реализации технологий, обусловленный тенденцией декомпозиции, специализации и стандартизации компонент сложных систем;
3. Интегрирующий аспект развития и модификации технологий, обусловленный целевыми установками их создания и взаимным проникновением частных технологических решений.

В целом, наиболее просто реализуется первая проекция, поскольку как показано в подразделе 1.2, поколения ЭВМ развернуты во времени и достаточно «жестко» ограничивают возможности реализации технологий.

С другой стороны, целью дисциплины является изучение не аппаратных средств ЭВМ, а программного обеспечения, в котором идейная и прагматическая части технологий сгруппированы в собственном ключе, имеющем другие общепринятые названия.

Учитывая сложившиеся традиции, можно выделить ряд направлений, объединяющих компьютерные технологии в следующие группы:

- Вычислительные технологии, — когда компьютер рассматривается как мощный калькулятор, способный обеспечить решение многих расчетных задач.
- Технологии хранения информации, — когда требуется проектирование предметной области приложений и последующее хранение построенных моделей.
- Объектно-ориентированные технологии, объединяющие предыдущие концепции и создающие инструментальную базу для создания более сложных программных систем.
- Технологии автоматизированного управления, охватывающие своими возможностями интересы производственных предприятий.
- Технологии взаимодействия открытых систем, лежащие в основе распределенных вычислительных сетей.
- Офисные технологии, обеспечивающие массовое использование ЭВМ.
- Сервисные технологии, отражающие новое качество компьютерных технологий.
- Интеллектуальные системы, закладывающие потенциал для будущих технологических решений.

В целом, предложенная последовательность технологических достижений достаточно точно отражает исторический аспект их появления. 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. Что касается двух других проекций, то они, по возможности, раскрываются индивидуально в рамках выделенных технологических направлений.

Забегая вперед — отметим, что выделенные технологические направления являются достаточно стабильными, а значит представляющими интерес для их изучения, хотя в разное время они представляют различный научный интерес. Многие из них не оправдали возложенные надежды или поменяли контекст восприятия, но в любом случае не ушли со сцены времени и продолжают интенсивно использоваться.

Указанное свойство стабильности технологических направлений вызвано многими причинами, первая из которых — энерция общественной практики, присущая всем технологическим достижениям человечества. С другой стороны, пока расширяется область применения компьютерных технологий, каждое выделенное технологическое направление будет иметь свою нишу и закрепляться в ней.

Сказанное можно подтвердить множеством примеров, даже на базе уже изученного материала. Например, концепция фон-неймановской архитектуры ЭВМ, победила гарвардскую архитектуру во времена первого поколения ЭВМ, но вполне успешно применяется в сигнальных процессорах, ориентированных на скоростную обработку потоков данных.

Несмотря на в целом позитивный аспект развития выделенных технологических направлений, не следует забывать и о негативных тенденциях, которые следует учитывать в процессе изучения данной дисциплины. Следует учитывать, что позитивное развитие любой технологии, в следствие диалектического характера самого процесса развития, приводит к накоплению негативных качеств, которые в сумме могут иметь значительный эффект. Например, объектно-ориентированные технологии, обеспечивая позитивную возможность программирования приложений на более высоком уровне, порождают, по сравнению с языками третьего уровня, следующие негативные последствия:

- увеличение объема программного обеспечения;
- повышенный расход памяти ЭВМ;

- сокрытие и, как следствие, худшее понимание деталей реализации ПО на низком уровне.

Таким образом, все появившиеся компьютерные технологии практически не исчезают с течением времени и требуют тщательного изучения с целью адекватного их использования.

Это и является основной целью изучения данной дисциплины.

1.5 Контрольные вопросы

1. Какие идейные парадигмы определяют 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»?
7. В чем суть модульного подхода к построению ЭВМ и в чем она проявляется?
8. Какое общепринятое количество поколений ЭВМ принято выделять?
9. Чем выделяется четвертое поколение ЭВМ?
10. Какие технологические направления характеризуют современные ком-пьютерные технологии?
11. Какие два технологических направления выделяет классификация СОД?
12. Чем отличается вычислительный комплекс от вычислительной систе-мы?
13. Чем отличаются системы телеобработки от вычислительных сетей?
14. Чем знаменит проект IBM 360?
15. В чем основное отличие гарвардской архитектуры ЭВМ от архитек-туры фон-Неймана?
16. Чем знаменита корпорация DEC?
17. Каковы временные рамки шестого поколения ЭВМ?
18. В чем историческое значение работ академика Глушкова В.М.?
19. Какое количество составляющих присутствует в классической много- уровневой архитектуре ЭВМ?
20. Сколько уровней имеет классическое ядро ОС UNIX?

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 необходимо сосредоточить свое внимание на следующих вопросах:

- Компьютер как вычислитель.
- Парадигма «программа-массив».
- ОС и системы разработки программного обеспечения.
- Технологии расчетов и моделирования.
- Интегрированные системы научных и инженерных исследований.

- Системы Mathematica, Maple, Mathcad, MATLAB, Simulink.

Чтобы правильно охватить весь объем учебного материала, он разделен на три основные части (подраздела), где, в первой части, сконцентрированы все основные идейные и прикладные достижения, сформировавшиеся в рассматриваемом направлении. Остальные две части содержат краткий обзорный материал конкретных современных достижений, грубо разделенных на два прикладных направления.

2.1 Идейная и базовая части вычислительных технологий

Среди всех направлений, вычислительные технологии в своем интенсивном развитии охватывают наиболее длительный период времени. Можно смело утверждать, что компьютерные технологии порождены этим направлением и продолжают развиваться в настоящее время. Тем не менее, в методическом и организационном планах следует особо выделить период до появления ОС, обсуждаемый в пунктах 2.1.1 и 2.1.2, и современное состояние этого направления, описанное в пункте 2.1.3.

2.1.1 Компьютер как вычислитель

Компьютер задумывался как вычислитель. Компьютер и появился как вычислитель. Опираясь на последовательность поколений ЭВМ, можно смело утверждать, что первые три из развивали именно технологии вычислений.

Нулевое поколение (1492-1945) — механическая эра, включая электромеханические устройства вычислений:

- вычисления появляются как идея учета количественных характеристик свойств реального мира и реализуются в виде простейших вычислителей: аббак и счеты;
- формируется аппарат аналитических вычислений, создаются и совершенствуются механические и электромеханические устройства счета.

Первое поколение (1937,1945 -1953,1955) — электронные лампы и первые ЭВМ создают новую технологическую базу, которая существенно увеличивает скорость вычислений. Совершенствуется научное направление «Численные методы» («Вычислительные методы» как теоретическая часть математического направления «Вычислительная математика»).

Второе поколение (1954,1955 - 1962,1965) — полупроводники и транзисторы существенно расширяют сферу применения ЭВМ и «Вычислительную математику», как выразительницу их целевого назначения.

Сказанное выше можно подтвердить образными тезисными высказываниями, сформулированными специалистами во времена первого и второго поколений ЭВМ:

- первое поколение провозглашало: «Компьютер — это супер дорогие счеты»;
- в более позднее время, с появлением первых микросхем, стали говорить «Компьютер — это большой калькулятор».

Такая направленность мышления вызвана основными потребностями социума, выражаемые в необходимости сложных и затратных вычислений:

- 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».

В целом — в рассматриваемый период времени, общество накопило множество вычислительных задач и создало математический аппарат для их решения. Вычислительные машины стали обеспечивать технологическую основу для их решения. Поставленные задачи стали решаться, с тем или иным переменным успехом, повышая свою социальную значимость. Все это стало определять основную социальную направленность использования вычислительной техники.

Опираясь на указанные потребности, социум стал формировать профессии, связанные исключительно с трудовой деятельностью, ориентированной на применение компьютеров. Эта профессиональная деятельность стала формировать мышление специалистов, создавать теории и парадигмы вычислительных технологий.

Далее, рассмотрим две основные парадигмы.

2.1.2 Парадигма «программа-массив»

Размышляя о компьютере как о калькуляторе, следует учесть, что решение вычислительных задач не сводится к одному математическому действию и не может быть выполненно одной командой. Следовательно, требуется уточнение: «Компьютер — это большой программируемый калькулятор».

Упомянутый выше социальный процесс, ориентированный на использование ЭВМ, стал формировать профессиональную деятельность, каждый представитель которой сейчас называются «Программист».

Программист стал элитой, которая была призвана сформировать ту профессиональную прослойку общества, которая обеспечит потребности социума в плане решения вычислительных задач.

Сама элита стала формироваться из ученых и научно-технических специалистов высокой квалификации, способных не только программировать, но и формулировать нужные теоретические и технологические парадигмы адекватные имеющемуся поколению аппаратных средств ЭВМ.

Основной парадигмой вычислительных технологий стала концепция «Программа-массив», которая оставалась актуальной, вплоть до середины 60х годов XX-го века.

Технологическая суть этой парадигмы — в разбиении самого процесса вычислений на три последовательных этапа:

- подготовка и ввод в ЭВМ исходных данных для обработки;
- обработка исходных данных программной или программным модулем;
- вывод результатов вычислений на внешние носители (хранители) информации ЭВМ и последующая интерпретация результатов работы программы.

Технологическое расширение этой парадигмы - взаимодействие множества программ, обеспечивающих решение сложной целевой задачи на базе имеющихся данных.

Основные технологические характеристики данной парадигмы выражаются следующим образом, через характеристики самих программ:

- программы «плоские», а данные и код находятся в одном адресном пространстве и не разделены между собой;
- программные модули (программы) - слабо связаны и осуществляются только через данные и процесс запуска отдельных программ;

- значимость программы оценивается количеством операторов: кода самой программы;
- направленность алгоритмов — исключительно расчетная;
- имеется сильная насыщенность программ деталями, далекими от прикладной направленности алгоритма вычисления.

Достаточно очевидно, что парадигма «программа-массив» имеет существенные недостатки, главным из которых является несогласованная работа с данными.

Действительно, уделяя основное внимание программе, программист готовит исходные данные в формате необходимом для простейшей реализации вычислительного алгоритма задачи и удобном для ввода в ЭВМ. В результате при взаимодействии, любые две программы требовали написание третьей программы, преобразующей выходные данные одной в формат исходных данных другой. Это стало очень быстро сказываться при усложнении программных систем.

Сам факт длительного периода существования парадигмы «программамассив» связан с еще недостаточно широким распространением компьютеров, концентрацией программистов вокруг вычислительных центров научных и инженерных организаций. С другой стороны, проявляющиеся недостатки технологии стремились устранить дополнительными технологическими решениями, например:

- уменьшить излишнюю информационную загруженность программ;
- перейти от машинного кода к ассемблеру, макросам и 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.

Результатом таких технологических решений является идея разделения программного обеспечения (ПО) по его 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, обеспечения инженерных и научных расчетов.

2.1.3 ОС и системы разработки программного обеспечения

7 апреля 1964 года корпорация IBM объявила о выпуске серии из шести ЭВМ разных моделей, отличающихся по мощности и стоимости. Эта серия была заявлена как IBM System/360 (S/360) и объявлена компьютерами третьего поколения. Считается, что на разработку этой серии корпорация потратила порядка 5 миллиардов долларов.

Эта разработка была столь удачной, что стала образцом для подражания на многие годы для ЭВМ класса mainframes. В СССР аналогом этого направления были компьютеры

серии ЕС ЭВМ (*Единая система электронных вычислительных машин*), выпускаемая странами СЭВ (*Совет экономической взаимопомощи*) куда входили: Болгария, Венгрия, Польша, Румыния, СССР и Чехословакия. 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 вычислительной техники.

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 раз-личных поставщиков ПО;
- *неспособность* или коммерческая нецелесообразность создания и сопро-вождения библиотек ПО для множества меняющихся архитектур ВТ.

Тем не менее, потребности в поддержке сложных математических 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, 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*** . Появившаясь *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: сло-жение, 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 технологий ***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 выше технологии.

Технология CUDA или ***Compute Unified Device Architecture***. 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 также 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***. 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 ***G**eneral-**p**urpose **g**raphics **p**rocessing **u**nits*: «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***. Это - открытая графическая библиотека и графическая API-спецификация, определяющая независимый от языка программирования и платформы программный интерфейс для написания приложений, которые используют двумерную и трехмерную 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 A**udio **L**ibrary*. 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. Он 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 программирования, ***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, 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*** .
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 программиро-вание.
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. 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.

Замечание
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 на 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*). 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 инже-нерные 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, авто-матически 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» - 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 .

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, объектноориентированные возможности и интерфейсы к программам, написанным на других языках программирования.

Программы, написанные на ***MATLAB***, бывают двух типов: функции и 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*** также 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***. 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 другие.

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 Контрольные вопросы

1. В чем состоит историческая значимость парадигмы «программамассив»?

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, появились и ***смешанные технологии***, которые обеспечили *автоматизацию проектирования информационных систем*.

Чтобы подробнее раскрыть эту тему, кратко рассмотрим ряд качественных свойств, которые присущи сложным программно-техническим системам. Такими свойствами являются: ***переносимость*** *и* ***интероперабельность*** *(способность к взаимодействию) 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, во всех необходимых областях максимально строго придерживайтесь международных или общепризнанных фактических стандартов.

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 ***C 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 со-держат 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:

1. *Это позволило* сосредоточить усилия на создании технологий, обеспечивающих массовое и долговременное хранение данных, доступных, в дальнейшем, для решения научных и производственных задач.
2. *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 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 90754:2003, который регулирует эту проблему.

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 технологическо-го направления.

Oracle или *Oracle Corporation:* это - американская корпорация, являющаяся крупнейшим в мире *разработчиком программного обеспечения* для организаций, а также крупным *поставщиком серверного оборудования* .

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 (интерактивная аналитическая обработка) 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 (***Enterprise Resource Planning***, планирование ресурсов) 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***, ***J2EE***, 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, вомногом заимствованными у конкурентов, теоретическими и практическими изысканиями, известными под общим названием ***ADO.NET***:

- ***Microsoft SQL Server*** - система управления реляционными СУБД, разрабо-танная корпорацией Microsoft [4, 5]. Использует основной язык запросов - Transact-SQL, который был создан совместно Microsoft и Sybase.
- ***Transact-SQL***, или 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***.

Замечание
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;
- *содержит* полноценную поддержку новых типов данных, включая XML-спецификации, и является полноценным сервером СУБД;

- *включает* все необходимые компоненты программирования и поддержку национальных алфавитов и ***Unicode***, используется в приложениях при проектировании информационных систем или для самостоятельного изучения;
- *не имеет 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.

Замечание
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 Контрольные вопросы

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 технологии

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.

Замечание

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:

1. Возникла необходимость адаптировать многие теоретические концепции и практические приемы их использования к природной ментальности человека.
2. 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 — метасвойство 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.

Суммируя перечисленные свойства объектно-ориентированной парадигмы, можно утверждать, что:

- ***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 метаописательную 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, что 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 наследование - одно из основных и мощных свойств языков ООП, тем не менее, это свойство порождает проблему неадекватного изменения поведения объектов, при измене-нии (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 (***BM, virtual machine***), это — термин, имеющий следую-щие 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 С*».

Виртуальная машина исполняет некоторый машиннонезависимый код, например, *байт-код, шитый код, p-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 отдельном 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.

Замечание
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:

- *защиты* 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***;
- *внедрения* вредоносного кода для управления инфицированной системой, например, вирус ***PMBS***, обнаруженный **в 1993 году**, а также вирус типа 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;
- *тестирования и отладки* 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 по-мощью 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.

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

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

Замечание
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 MSILпрограммам, 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*** .

Общее классическое представление КОП на структуру приложения показано на рисунке 4.1.

Рисунок 4.1. Структура приложения в классическом представлении КОП.

Замечание
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.
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. Выделяют три отличия ***КОП*** от ***ООП***:

- *компонент* - «независимый модуль программного кода, предназначенный для повторного использования и развертывания»;
- *компонент* может содержать «множество классов»;
- как правило, *компонент* является независим от конкретного языка.

Замечание
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, которая 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 - нерешаема, 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, 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***;
- производный класс может пользоваться только публичным интерфейсом базового класса.

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

Замечание
В 1987 году Вирт, унифицировав язык Оберон, предложил шаблон проектирования (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**.

Плагин (***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 необхо-димости 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 про-игрывателей, 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 диа-пазона.
- Некоторые плагины изменяют технические характеристики звука: глубину и частоту дискретизации.
- Большой популярностью пользуются плагины для почтовых программ: спамфильтры, 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 [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.

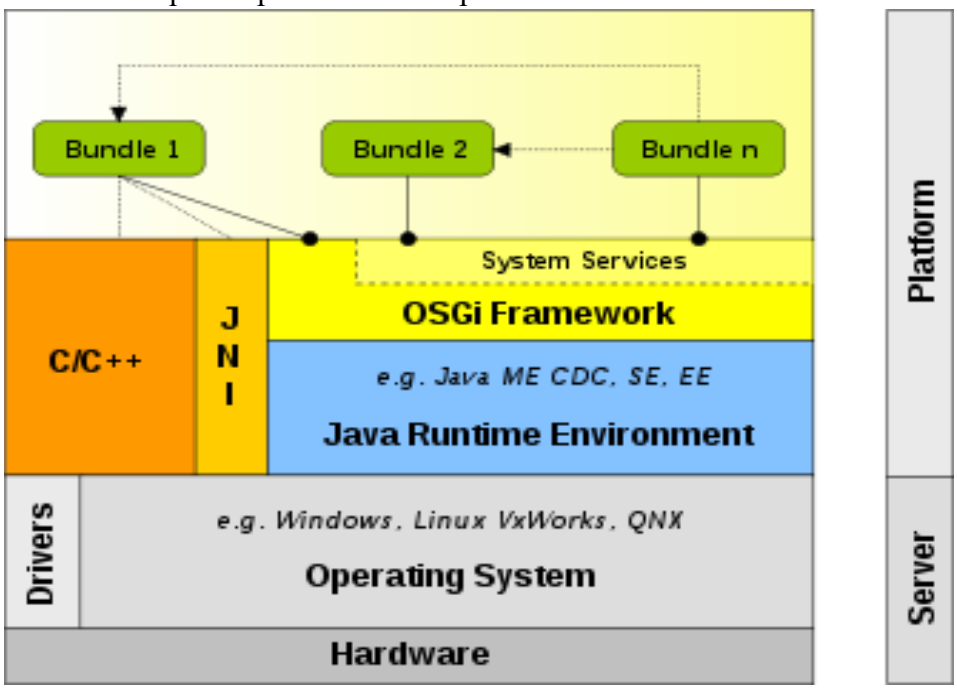

Рисунок 4.2 - Наборы служб (bundles) OSGi [4].

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

Существуют ***четыре реализации стандарта OSGi***, удовлетворяющие требованиям открытого исходного кода:

- ***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.

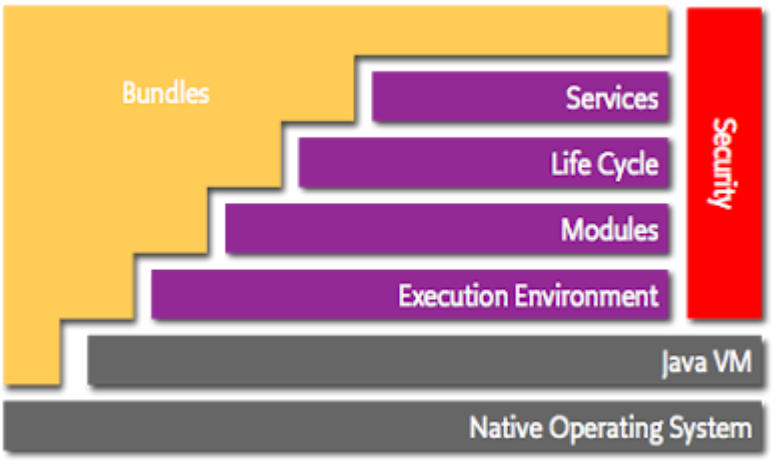

Рисунок 4.3 - Архитектура сервисов OSGi [5].

Замечание
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 рисунком 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.

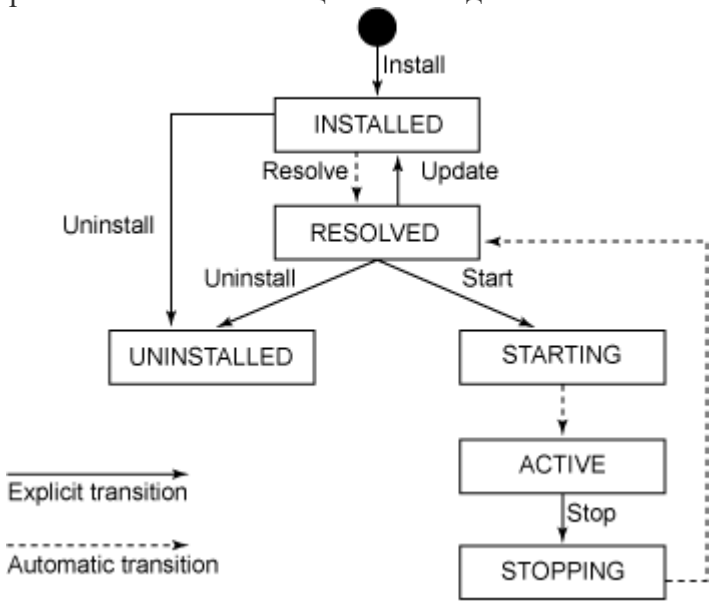

Рисунок 4.4 - Состояния бандла OSGi [6].

В завершение обсуждения этого раздела, отчетим 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.
Замечание
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. Таким образом, технология OSGi представляет собой

сервис-ориентированную платформу с поддержкой модульности для разработки приложений.

4.4 Контрольные вопросы

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.

Далее, 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 имеют ряд общих свойств:

- *распространяются*, как правило, только вместе;
- *имеют* схожий интерфейс;
- *хорошо взаимодействуют* друг с другом.

Как правило, набор офисных пакетов содержит следующийе компоненты:

- *Текстурый 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 инфор-мационно 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 дан-ный момент последней является шестнадцатая версия, известная также, как Microsoft Office 2016;
IBM Lotus Symphony — бесплатный офисный пакет корпорации IBM, ос-нованный 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 сов-местим 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 таблицей 5.1, откуда 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.	1 994
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 идеологии 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. 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* (*СЭД, СЭДО*).

Система автоматизации документооборота, **система электронного документооборота (*СЭД, 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 СЭД даны в документе: «ГОСТ Р 7.0.8-2013 *Делопроизводство и архивное дело. Термины и определения*». Рассмотрим ряд основных базовых определений:
Документооборот - движение документов в организации с момента их создания или получения до завершения исполнения или отправки.

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.

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.

Хотя понимание представленных определений не вызывает серьезных затруднений, тем не менее, проектирование и реализация СЭД сталкивается с проблемами разделения программного обеспечения между компонентами системы автоматизации. Обычно это касается компонент *делопроизводства* и *деловых процедур.*

5.2.1 Делопроизводство и деловые процедуры

На рисунке 5.1 показана схема соотношения между компонентами *делопроизводства* и *деловых процедур*. Их отличительные особенности 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.

Рисунок 5.1 — Схема соотношения делопроизводства и деловых процедур

В приведенном примере деловой процедуры, ***к делопроизводству*** имеют отношение лишь пункты:

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

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.

Замечание
В государственных организациях, 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* отвечают 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.

5.2.2 Западные системы автоматизации делопроизводства

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 существовали серьезные затруднения по внедрению зарубежных СЭД на предприятиях России, что было вызвано:

1. Отсутствием законченных продуктов для автоматизации российского делопроизводства.
2. Отсутствием активного спроса на средства автоматизации деловых процедур.
3. Путаницей в позиционировании продуктов западных производителей для целей автоматизации российского делопроизводства, а также множество других аналогичных проблем.

5.2.3 Три источника и три составные части ДОУ

Сам процесс принятия управленческого решения включает в себя поэтапную работу с информацией:

- получение информации,
- переработка информации;
- анализ, подготовка и принятие решения на основе обработанной информации.

Эти составные части управленческого решения самым тесным образом связаны с документационным обеспечением управления (*ДОУ*), поэтому принято выделять три основные задачи, решаемые в системах делопроизводства:

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, поскольку само управление касается множества компонент, обобщенно показанных на рисунке 5.2.

Рисунок 5.2 - Компоненты управления предприятием

С ростом масштабов предприятия и численности его сотрудников вопрос об эффективности ДОУ становится все более и более актуальным. Основные проблемы, возникающие при этом, выглядят приблизительно 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 других.
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.
7. В итоге, дальнейшее расширение предприятия происходит чисто экстенсивным путем за счет ранее накопленной прибыли или увеличения дефицита бюджета.

Наиболее распространенное решение возникающих проблем - это отдельная *автоматизация рабочих мест* (***АРМ***) предприятия, наподобие:

- 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 программных продуктов для обеспечения автоматизации.

5.3 Интеграция офисных приложений

Как уже было отмечено в подразделе 5.1, офисные пакеты компонуются как набор приложений, обеспечивающих редактирование текста, рисунков, формул, таблиц и других элементов, которые ассоциируются с подготовкой документов или печатных изданий. С другой стороны, потребности производства, рассмотренные в подразделе 5.2, диктуют необходимость:

- формализации понятия документ (электронный документ);
- разработку средств интеграции функциональных возможностей, реализуемых в отдельных компонентах офисных пакетов, применительно некоторому формату электронного документа.

Указанные потребности привели к созданию открытого формата документа для офисных приложений, который стандартизирован в [3], под названием «ГОСТ Р ИСО/МЭК 26300-2010 Информационная технология. Формат Open Document для офисных приложений (OpenDocument) v1.0 (Введение - Раздел 12)». В России, данный стандарт введён в действие с 01.06.2011.

Характерной особенностью данного стандарта использование универсального языка разметки **XML** (***eXtensible Markup Language***), что обеспечивает документы следующими преимуществами:

1. Пользователи, сохраняющие свои данные в открытом формате, избегают опасности быть загнанными в угол единственным поставщиком, поскольку они свободны выбрать другое программное обеспечение.
2. OpenDocument является единственным стандартом для редактируемых офисных документов, утверждённым независимым комитетом по стандартам и реализованным несколькими поставщиками программного обеспечения. OpenDocument может быть использован любым поставщиком ПО, включая поставщиков закрытого программного обеспечения и разработчиков, использующих открытые стандарты.

К недостаткам формата относят:

1. Отсутствие стандарта на цифровые подписи в версиях OpenDocument Format 1.0-1.1, хотя новая версия формата 1.2 уже включает XML-Dsig.
2. Спецификация OpenDocument Format, принятая ISO, не определяет язык формул, хотя OASIS работает над своей стандартизацией для ODF версии 1.2.
3. Спецификации ODF 1.0-1.1 не допускают использования таблиц в презентациях, хотя эти возможности включены в версию 1.2, разработанную OASIS.

В заключение отметим, что развитие офисных технологий находится в интенсивном развитии и будет продолжать естественным образом интегрироватся с другими технологиями.

5.4 Контрольные вопросы

1. В чем состоит основное назначение офисных технологий?

2. Какой набор офисных приложений вам известен?
3. Почему офисные приложения разделены на отдельные части?
4. Какой открытый офисный пакет наиболее популярен?
5. Как называется офисный пакет корпорации Microsoft?
6. Что такое системы документооборота?
7. В чем различие между документооборотом и деловыми процедурами?
8. Какая связь имеется между офисными пакетами и СЭД?
9. Какие зарубежные СЭД вам известны?
10. В чем смысл интеграции офисных приложений?

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 вычислительной техники 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 .

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

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.

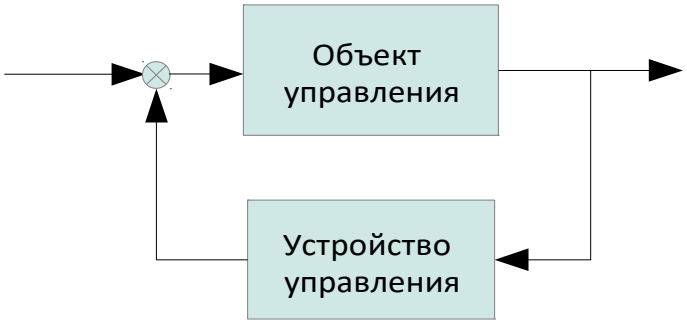

Рисунок 6.1 - Кибернетическая модель объекта управления с обратной связью.

С помощью обратной связи можно организовать два наиболее общих способами воздействия на управляемый объект:

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

Таким образом, буквально начиная с первого поколения средств цифровой вычислительной техники, 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 рисунке 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 на 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 ряд других концепций, 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***).

Внедрение идей АСУ сразу же столкнулось со множеством проблем, главной причиной которых был сам человек:

- человек, за рядом частных случаев, не способен непосредственно наблюдать или воспринимать выходные характеристики объектов;
- человек не способен непосредственно воздействовать на входы многих 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, потребности промышленного производства требовали включать людей в контур управления предприятиями. Соответственно, кибернетическая модель, представленная на рисунком 6.1, стала рассматриваться как базовая теоретическая концепция, на основе которой строились и развивались модели АСУ. Такие модели позволили:

- *переосмыслить* многие взаимодействия между людьми, участвующими в управлении производством;
- *дать* научное объяснение многим, не решенным ранее вопросам;
- *стимулировать* 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 структуры*, 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 и 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* - продукта 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, он 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, что 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;
- 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,
- *правил*, регламентирующих 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, которые несколько шире АСУТП, потому что охватывают:

- системы сбора первичных данных (полевой уровень), находящийся по иерархии управления ниже АСУТП;
- функции диспетчеризации, которые 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 значительно выше, чем участие человека-оператора в АСУТП.

Хотя, задачи диспетчеризации являются специфическими для различных производств, тем не менее, имеются производства, где диспетчеризация техно-логических 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 технологическое 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:

- *наличие* множества удаленных от центра обработки точек сбора измерительной информации;

- *потребность* 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 ГОСТах серии 24 начала 80-х годов . 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 .

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 управля-ющие 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.

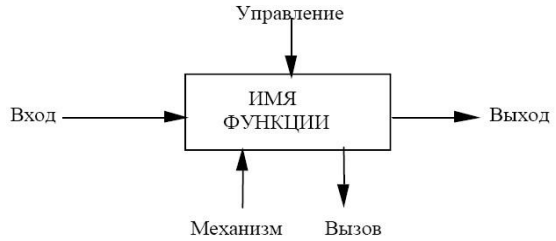

Рисунок 6.2 - Макроопределения модели IDEF0

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 Контрольные вопросы

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.

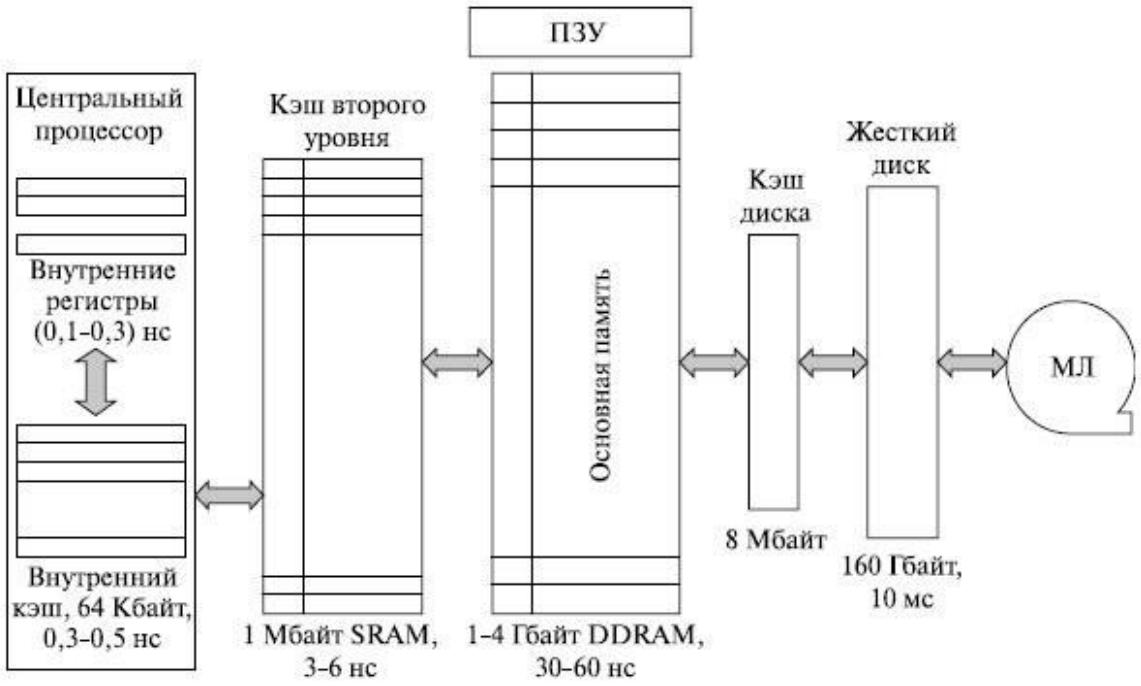

Рисунок 7.1 - Обобщенный вид памяти ЭВМ

Такая, во-многом оригинальная, структура устройств памяти поддерживается средствами операционной системы (ОС). Очевидно, что создание и сопровождение специального ПО, обеспечивающего взаимодействие многих ЭВМ, становится очень сложной и трудоемкой задачей.

Критическая ситуация возникла с появлением персональных компьютеров (ПК), поскольку 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 называться 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***. Эта модель 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 . 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 и 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. Это связано с тем, что модель DoD достаточно длительное время обеспечивала успешное развитие большинства технических решений с использованием Интернет-технологий. Чтобы показать как это происходило и к чему привело, рассмотрим два направления их развития:

- технические решения, обеспечивающие развитие самих сетевых технологий;
- технические решения, основанные на интеграции сетевых технологий и других технологических подходов.

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 рисунке 7.2.

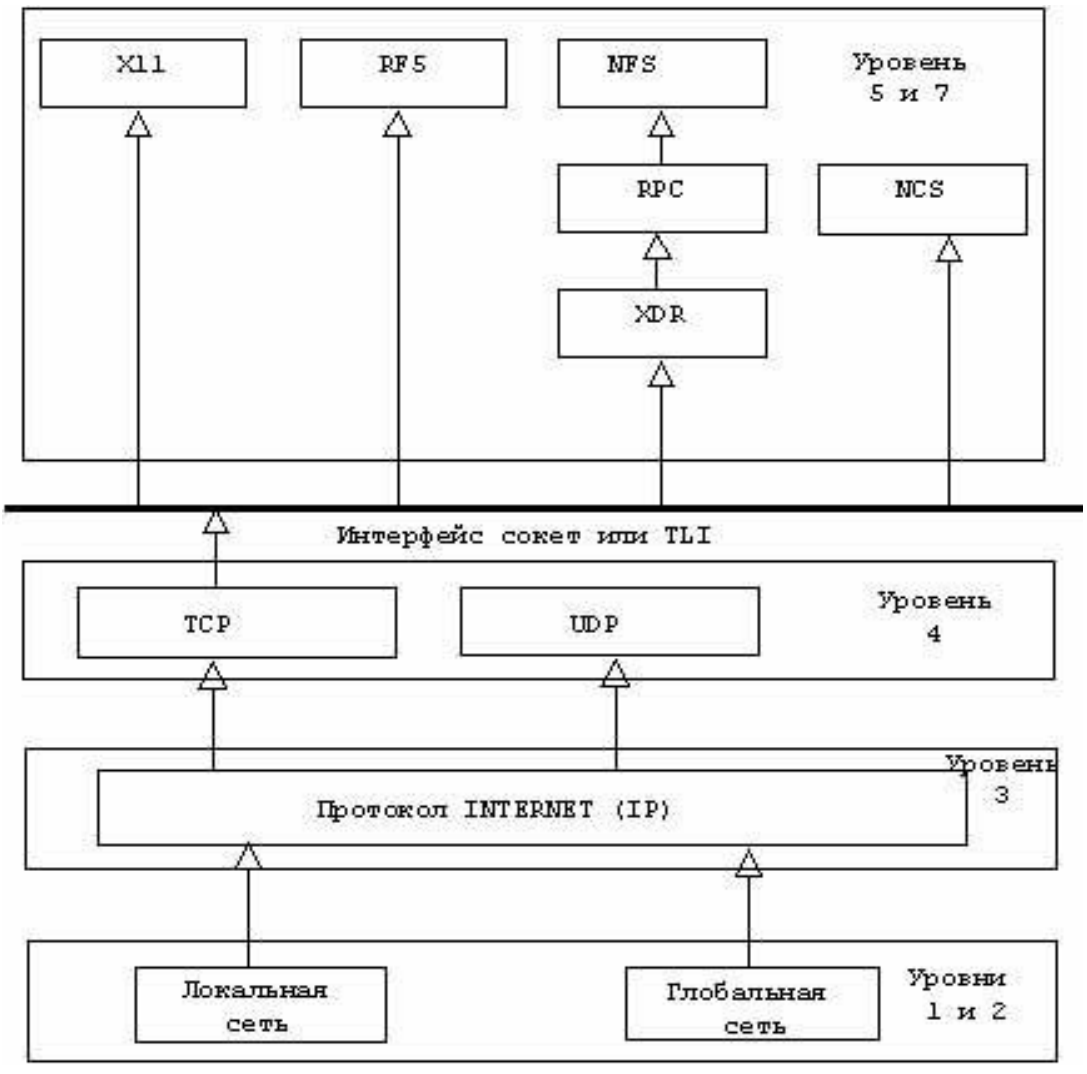

Рисунок 7.2 - Интерпретация базовых сетевых приложений сети Интернет

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 посредством целенаправленного злоумышленного внешнего воздействия (хакерством). Сети стали угрозой уже существующим компьютерным технологиям, поскольку 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 таких 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.

Другим новым теоретическим концептом стала идея объектно-ориентированного подхода (ООП) в программировании.

Суть идеи основана на ***проектировании*** и ***именовании*** сетевых объектов, взамен старым методам ***адресации*** конечных точек взаимодействия приложений в сети. Такой подход значительно упрощает проектирование, создание и эффективность использования столь сложных интегрированных АСУ, подобных системам SCADA.

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 написания 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, именно этот подход предложил идею объектной реализации распределенных систем, как образно показано на рисунке 7.3.

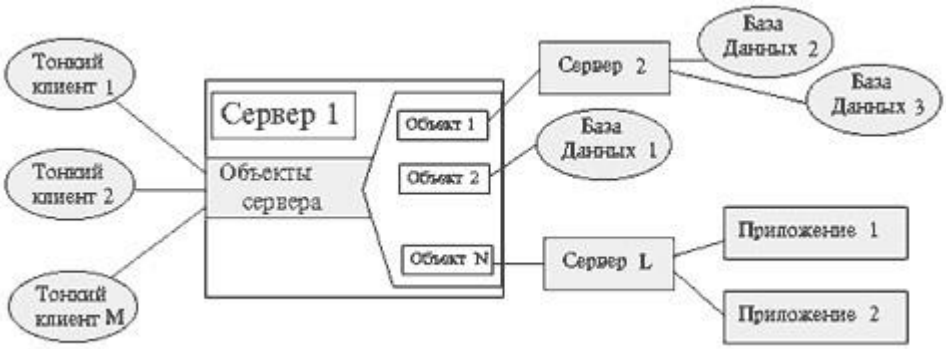

Рисунок 7.3 — Схема реализации сетевой интеграции прикладных объектов

На сегодняшний выделяются три такие технологии, реализующие концепцию распределенных объектных систем. Это - технологии ***RMI***, ***CORBA*** и ***DCOM***.

7.3.1 Технология RMI

Технология RMI обощенно представлена на рисунке 7.4.

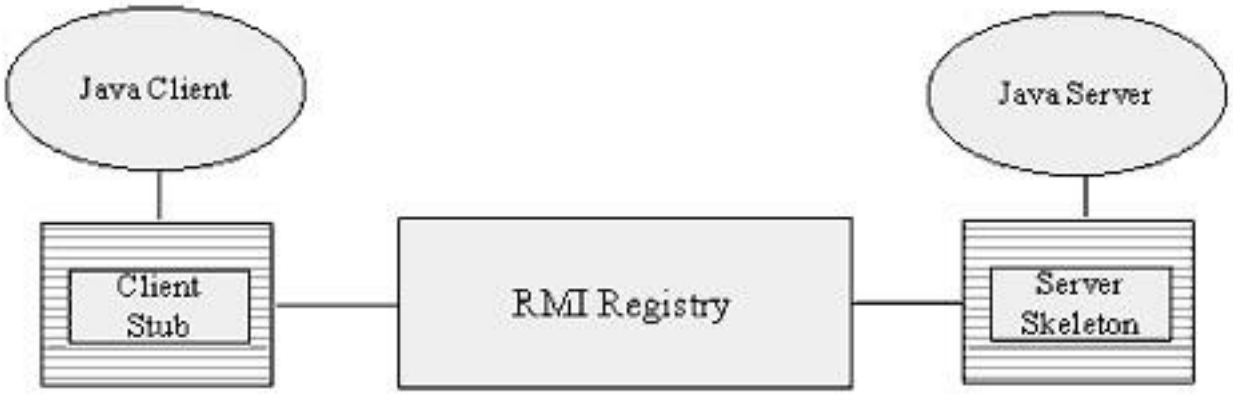

Рисунок 7.4 — Общее представление технологии RMI

Client Stub (переходник для клиента) и ***Server Stub*** (переходник для сервера) порождены от общего интерфейса, но различие своим назначением:

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

RMI (*Remote Method Invocation*) — механизм, который позволяет вызывать метод удалённого объекта.

RMI Registry — серверный сетевой объект, обеспечивающий регистрацию и адресацию взаимодействующих сетевых объектов.

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 технологии ***CORBA***, хотя контролируется теперь уже не корпорацией 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.

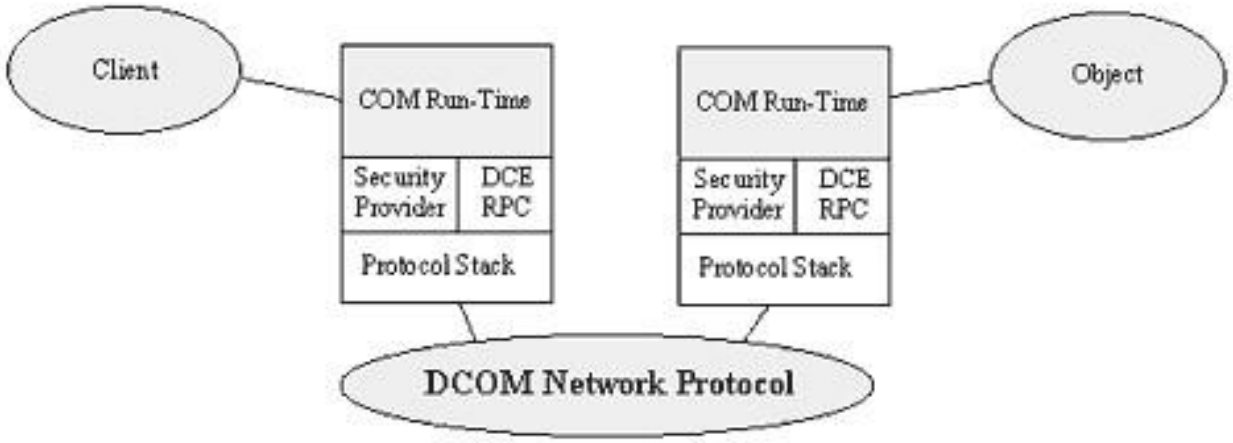

Рисунок 7.5 — Общее представление технологии DCOM

7.3.3 Технология CORBA

Технология CORBA является наиболее обобщенной среди указанных моделей взаимодействия в сети.

Технология CORBA (*Common Object Request Broker Architecture)* — разрабатывается группой ***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 представлена на рисунке 7.6.

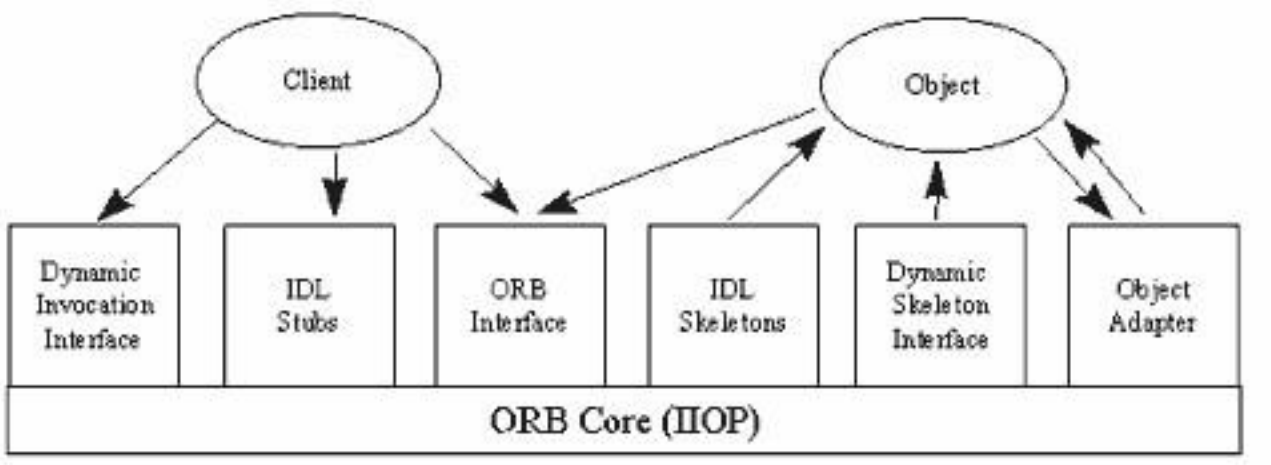

Рисунок 7.6 — Основная структура технологии CORBA

Являясь полностью объектной сетевой архитектурой, технология CORBA содержит следующие основные 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 сер-висы;
- *IDL Skeleton:* обеспечивает статические интерфейсы для объектов опреде-ленного 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 потребности приложений. Новые, более общие представления о приложениях задаются моделью SOA, которая рассматривается в следующей теме.

7.4 Контрольные вопросы

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 темати-ки 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 компьютерная 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. В терминах «Все есть сервис», можно выделить следующие направления:

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, эта концепция — 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 сопутствующего ОС программного обеспечения.

С развитием Интернет-технологий, появился термин - *software ondemand*: *программное обеспечение по требованию*. В дальнейшем, **SoD** стали трактовать как **SaaS** — *software as a service*, а 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 стремление к интеграции различного функционала посредством использования различных технологий и концентрация его в профессиональных вычислительных центрах, обеспечивающих массовое обслуживание сервисов на коммерческой основе;
- превращение клиентских персональных компьютеров в сетевые графические станции, обеспечивающие интерактивное взаимодействие пользователей с центрами сервисного обслуживания.

Указанные тенденции глобализации сервисов порождают обратное явление, связанное с их специализацией и проникновением во все сферы социальной и общественной деятельности. Не вдаваясь в подробности указанных явлений, рассмотрим только аспекты, отражающие концептуальный уровень самих технологий:

- *модель SOA,* - как теоретическую базу модели сервиса;
- *www-технологии,* - как основную технологическую базу ранней и перспективной реализации модели сервиса;
- *облачные технологии,* - как коммерческую площадку для реализации сервисов.

Замечание
В современной терминологии используется термин ИТ-сервисы, трактуя сокращение ИТ как «Информационные технологии».

8.2 WWW-технологии и проект SOA

SOA (*Сервисно-Ориентированная Архитектура*, S*ervice-Oriented Architecture*) — 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 «оберткой», которая в большинстве случаев поставляет ИТ-пользователям «*настоящие сервисы*». Сервисы жизненного цикла отвечают за дизайн, внедрение, управление, а также за 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.

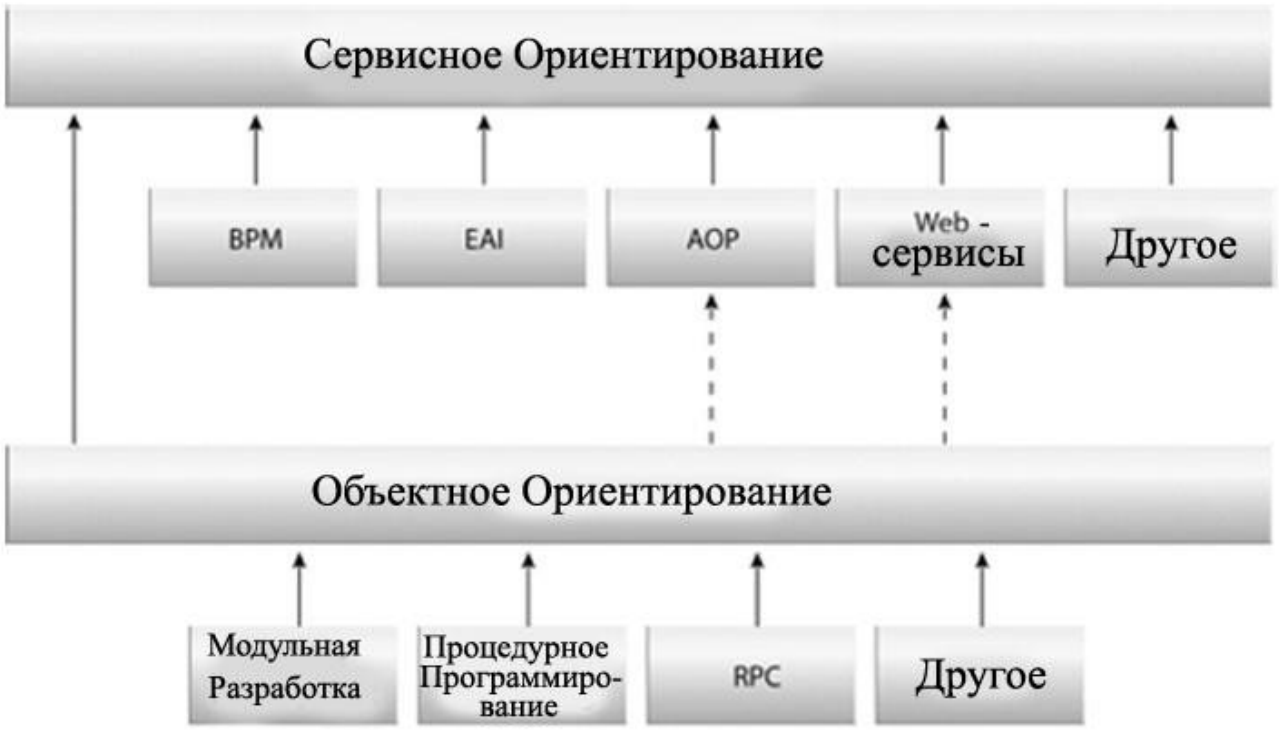

Рисунок 8.2 — Соотношение парадигм SOA и ООП

Появление *сервисно-ориентированного подхода* произвело очередную реформу в теории разработки программного обеспечения, оставив в прошлом концепцию ООП. Действительно, как известно, повторное использование программного кода упрощает разработку больших информационных систем. До недавнего времени, для этой цели традиционно применялся объектно-ориентированный подход, подразумевающий жесткое 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), а также фиксированных механизмов обмена информацией между объектами и модулями

информационной системы, то и в ООП сохраняются все эти зависимости и ограничения. Такой подход — не всегда удобен, поскольку он не позволяет оперативно реагировать на оперативные ситуационные потребности, например, проектировать новомодные системы, которые опираются на концепцию «ресурсы по требованию». Кроме того, для модификации объектных систем нередко приходится переписывать программное обеспечение, реализующее их объекты и методы.

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

Наряду с очевидными преимуществами, модель SOA имеет и свои явные недостатки, что можно выразить как «Проблемы обнаружения и поиска нужного провайдера сервиса». Дополнительно, это формулируется 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 указанных проблем с помощью подхода, сокращенно называемого ***ESB***.

ESB (*Enterprise Service Bus, Сервисная Шина Предприятия*) — связующее аппаратное и программное обеспечение, реализующее централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами на принципах сервис-ориентированной архитектуры.

На момент теоретической проработки, основу реализации моделей SOA составляли www-технологии. Соответственно, для целей передачи сообщений был разработан специальный протокол взаимодействия объектов SOA, названный ***SOAP***.

SOAP (*Simple Object Access Protocol, Простой Протокол Доступа к Объектам*) - протокол обмена структурированными сообщениями в распределенной вычислительной среде.

Для первой реализации SOAP была использована технология, названная «SOAP over HTTP», что подчеркивает важность и значимость www-технологий. Алгоритмическая суть реализации модели SOAP была представлена тремя вариантами методов взаимодействия потребителя и провайдера сервиса:

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

Рассмотрим более подробно каждый из указанных вариантов.

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, сокращенно называемая ***UDDI***.

UDDI (*Universal Description Discovery and Integration*) — универсальный описатель, представляющий собой Web-службу, которая является каталогом для поиска других Web-служб.

Идея заключается в развертывании UDDI-службы по хорошо известному потребителю адресу. Далее, потребитель может использовать UDDI для поиска других интересных ему Web-служб и напрямую взаимодействовать с ними. Само взаимодействие потребителя, UDDI-службы и провайдеров сервиса котировок схематично показано на рисунке 8.3.

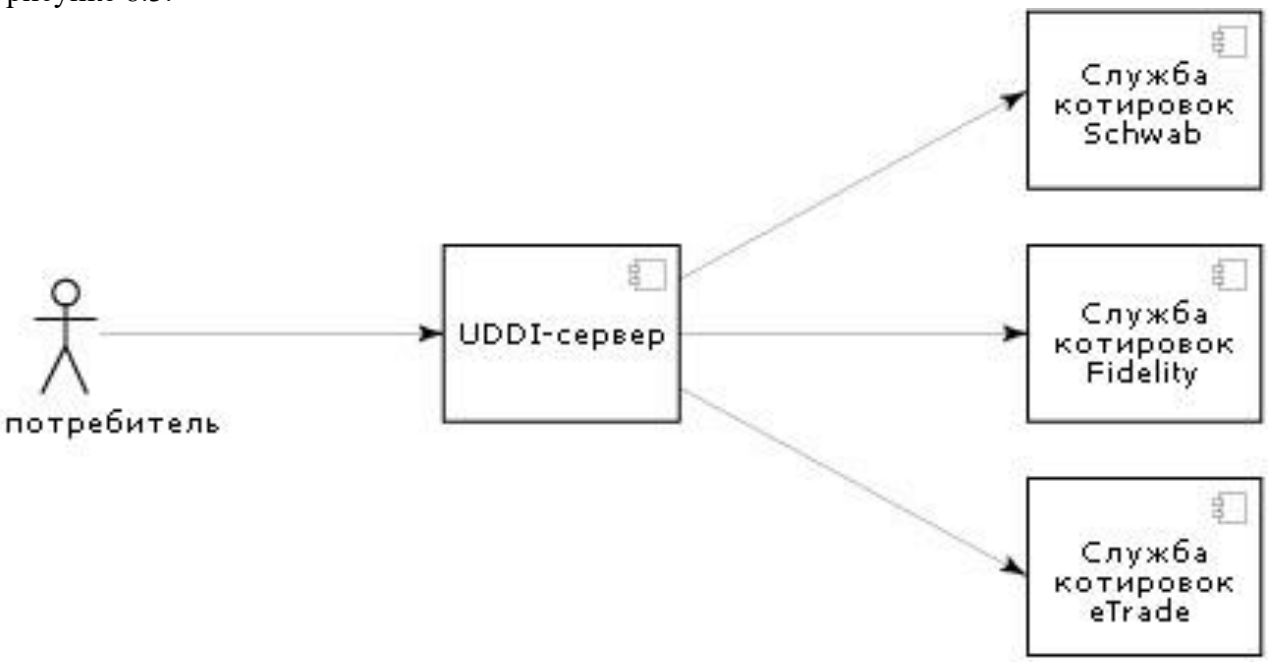

Рисунок 8.3 — Прямой вызов службы через UDDI

В приведенном примере, алгоритм взаимодействия состоит в поиске потребителем ***URI*** некоторой службы котировок.
URI (*Uniform Resource Identifier*) — унифицированный идентификатор ресурса.

Такой подход имеет ряд недостатков, которые хорошо видны из самого алгоритма взаимодействия потребителя и сервисных служб:

- потребитель использует сервис UDDI как каталог для поиска URI оконечных точкек провайдеров (сервисных служб);
- 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; в этом случае, потребитель должен повторно запросить UDDI для получения нового URI.

В сущности, данное описание показывает, что каждый раз, когда потребитель хочет вызвать службу, он должен запросить в UDDI URI оконечных точек и выбрать один из них. Это приводит к дополнительным затратам потребителем усилий, во время операций 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 уменьшения усилий потребителя является введение брокера, показанного на рисунке 8.4, который работает как промежуточное звено, при вызове Web-службы.

Рисунок 8.4 — Синхронный вызов через брокера

Теперь, потребитель вызывает прокси-службу брокера, который, в свою очередь, вызывает службу провайдера. В результате, 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.

Указанный подход также имеет ряд недостатков, обусловленных самим синхронным способом взаимодействия:

- потребитель должен ждать окончания выполнения работы службы, поскольку поток сообщений должен быть заблокирован на время работы самой службы;
- если служба выполняется длительное время, потребитель может прекратить ожидание ответа, например:

➔ 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 будет 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.

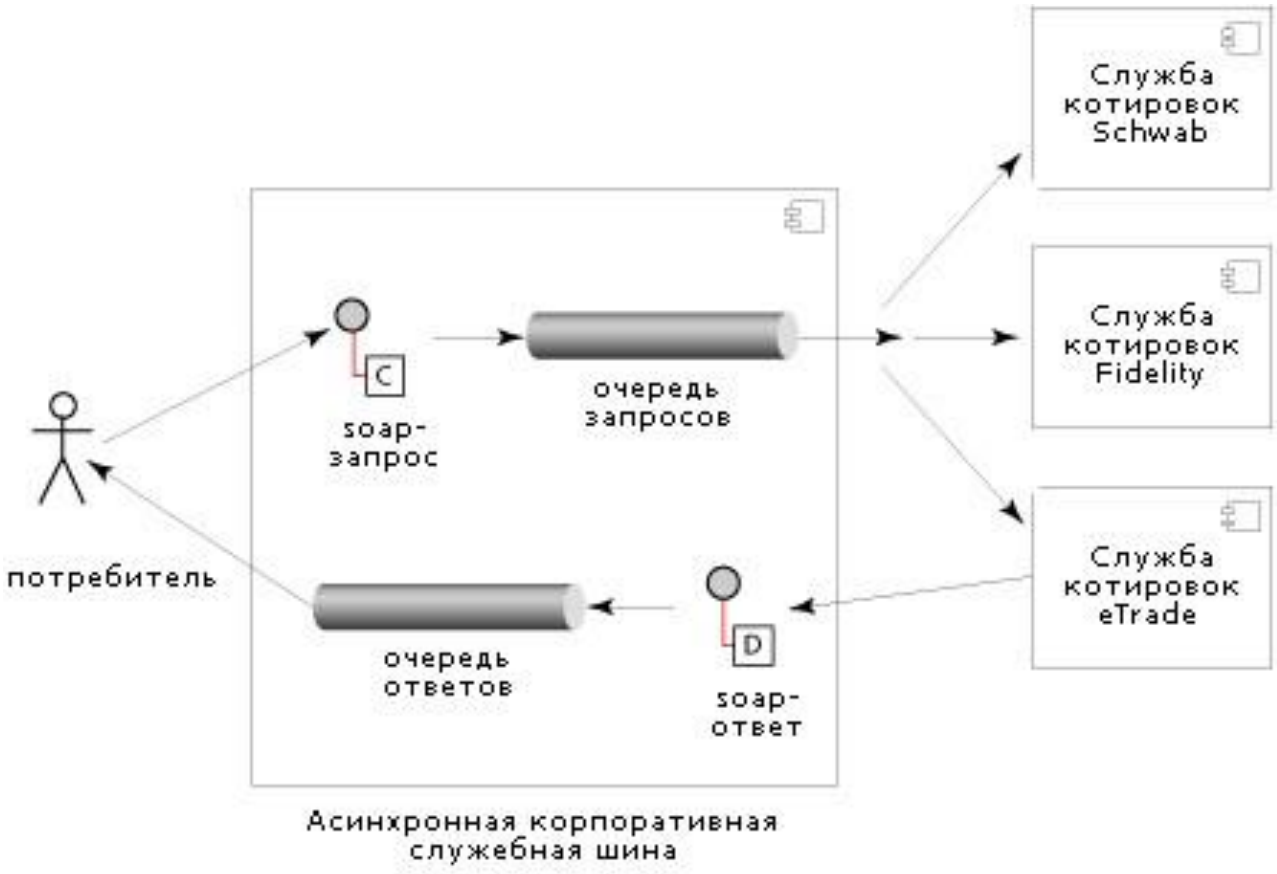

Рисунок 8.5 — Работа через брокера в асинхронном режиме

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 Webслужбу асинхронно, реализуется при помощи системы обмена сообщениями, которая использует очереди сообщений для передачи запроса и получения ответа.
- Аналогично синхронной прокси-службе пара очередей сообщений выступает 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.

Облачные вычисления (*cloud computing*) - технология распределенной обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как Интернет-сервис.

Здесь термин «Облако» используется как метафора, основанная на изображении сложной инфраструктуры, за которой скрываются все технические детали ее реализации. 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 достижения согласованной работы ЭВМ, которые предоставляют услугу облачных вычислений, используется специализированное программное обеспечение, обобщенно называющееся «***middleware control***». Это решает задачи:

- мониторинга состояния оборудования;
- балансировку нагрузки,
- обеспечение ресурсов для решения других задачи.

Существенной особенностью облачных вычислений является предположение о неравномерности потока запроса ресурсов со стороны клиента(ов). Для сглаживания этой неравномерности, между ЭВМ, реально обеспечивающих сервис, и middleware помещается еще один слой, задачей которого является виртуализация серверов. Таким образом, серверы, выполняющие приложения, виртуализируются и балансировка нагрузки осуществляется как средствами программного обеспечения, так и средствами распределения виртуальных серверов по реальным серверам.

Рассмотрим ряд конкретных реализаций облачных технологий.

8.3.1 Частное облако

Частное облако (*private cloud*) - инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей, например, подразделений клиентами или подрядчиками данной организации. Это частное облако может находиться в собственности, управлении и эксплуатации как самой организации, так и третьей стороны или какой-либо их комбинаций. Оно 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.

Таким образом, завершая краткое описание сервисных технологий, можно вполне обоснованно сделать следующие выводы:

- сервисные технологии в максимальной степени интегрируют все технологические достижения технологий предыдущих уровней;
- сервисные технологии проникают во все сферы деятельности современного общества, становясь его неотъемлемой частью;
- сервисные технологии имеют явно коммерческий уклон развития, что является не столько их внутренней сущностью, а сколько отражением современного уклада самого общества.

8.4 Контрольные вопросы

1. В чем состоит концептуальная суть сервисных технологий?
2. Какие направления были определены для сервисных технологий?
3. В чем назначение аутсорсинга?
4. Что такое модель SOA?
5. Какие имеются три аспекта ИТ-сервисов?
6. Что такое ESB?
7. В чем отличие между SOA и SOAP?
8. Что означает абревиатура «SOAP over HTTP»?
9. В чем суть облачных технологий?
10. Какие имеются виды реализации облачных технологий?

9. Интеллектуальные системы и технологии

В 1950 году, английский ученый Алан Тьюринг написал статью под названием «Может ли машина мыслить?». В статье описана процедура, названная позднее «Тест Тьюринга», с помощью которой можно определить момент, когда машина сравняется в плане разумности с человеком.

Как было отмеченно еще в первом разделе, при рассмотрении четвертого этапа развития аппаратных средств ЭВМ, в 1981 году правительство Японии объявило о намерениях выделить национальным компаниям 500 миллионов долларов на разработку компьютеров пятого поколения на основе технологий искусственного интеллекта.

Несмотря на достаточно длительный период развития и возложенные надежды реализации, термин ***искусственный интеллект*** не приобрел своего четкого и понятного содержания. Причин здесь много. Их анализу посвящено множество научных исследований, но единого мнения не достгнуто и сам термин отражает некоторое синтетическое понятие, конкретизация которого дается каждым исследователем, в рамках своей проблематики. Одна из современных трактовок дается участниками «Российской ассоциации искусственного интеллекта», в виде трех определений :

1. «Научное направление, в рамках которого ставятся и решаются задачи аппаратного или программного моделирования тех видов человеческой деятельности, которые традиционно считаются интеллектуальными.
2. Свойство интеллектуальных систем выполнять функции (творческие), которые традиционно считаются прерогативой человека. При этом интеллектуальная система — это техническая или программная система, способная решать задачи, традиционно считающиеся творческими, принадлежащие конкретной предметной области, знания о которой хранятся в памяти такой системы. Структура интеллектуальной системы включает три основных блока — базу знаний, решатель и интеллектуальный интерфейс, позволяющий вести общение с ЭВМ без специальных программ для ввода данных.
3. Наука под названием «Искусственный интеллект» входит в комплекс компьютерных наук, а создаваемые на ее основе технологии к информационным технологиям. Задачей этой науки является воссоздание с помощью вычислительных систем и иных искусственных устройств разумных рассуждений и действий.»

В целом, проблематике искусственного интеллекта посвящено множество серьезных работ, например, в дается обширный обзор исследований для инженеров и ученых. Имеются также учебники и учебные пособия. Постепенно, в сфере компьютерных технологий, термин искусственный интеллект стал заменяться термином ***интеллектуальная система***.

Интеллектуальная система (*ИС, intelligent system*) — это техническая или программная система, способная решать задачи, традиционно считающиеся творческими и принадлежащие конкретной предметной области, знания о которой хранятся в памяти такой системы. Структура интеллектуальной системы включает три основных блока — базу знаний, механизм вывода решений и интеллектуальный интерфейс.

Не пытаясь дать хотя бы поверхностный обзор существующим направлениям развития интеллектуальных систем, кратко упомянем только три из них, наиболее актуальные для тематики нашего обучения:

- *интеллектуальные информационные технологии,* - как развитие информационных систем;
- *системы искуственного интеллекта,* - как модель интеллектуальной деятельности человека;
- *робототехнические системы,* - как одно из широко исследуемых и практических ответвлений интеллектуальных систем.

9.1 Интеллектуальные информационные технологии

В одной из современных монографий дается следующее определение изучаемого понятия:

«**Интеллектуальная информационная система** (*ИИС*) – автоматизированная информационная система, основанная на знаниях, или комплекс программных, лингвистических и логико-математических средств для реализации основной задачи – осуществления поддержки деятельности человека и поиска информации в режиме продвинутого диалога на естественном языке.»

Далее, исследуемый вопрос классифицируется по следующим направлениям:

- системы с коммутативными способностями;
- экспертные системы;
- самообучающиеся системы;
- адаптивные системы.

Полная классификация ИИС представлена на рисунке 9.1, на котором показаны все три уровня декомпозиции.

Даже беглый обзор приведенной классификации показывает, что каждое из выделенных направлений использует свой оригинальный математический аппарат, напрямую неприменимый в других направлениях. Этот факт лишь подчеркивает высказанное ранее мнение, что интеллектуальные системы — это синтетическое направление научных исследований, которое может конструктивно развиваться только в рамках отдельных научных дисциплин.

Рисунок 9.1 - Классификация интеллектуальных информационных систем

Применительно к тематике изучаемой дисциплины, следует зафиксировать следующие выводы:

- значительный компонент интелектуальных информационных систем занимают вычислительные технологии, которые приобретают новый ранее не достигнутый уровень по своему многообразию и аспектам применения;
- непосредственно информационные технологии (в плане технологий хранения информации) составляют целевые объекты развития, преобретающие новые аспекты интерпретации;
- информационная составляющая компьютерных технологий развивается в направлении представления и хранения знаний;
- все другие технологии используются как вспомогательные, значимость которых определяется прикладным аспектом исследования.

9.2 Системы искусственного интеллекта

Учитывая многобразие подходов к понятию систем искусственного интеллекта, ограничим рассмотрение данного вопроса представлениями, изложенным в учебном пособии [8, глава 1].

Конкретизируя объект рассмотрения, выделим наиболее важные понятия прототипа, относительно которого всегда проводятся исследования, даются оценки и делаются выводы. Таким базовым прототипом является человеческий ***интеллект***.

Интеллект (*intellectus)* — качество человеческой психики, которое позволяет ему приспосабливаться к новым ситуациям внешнего мира, обеспечивает ему способность к обучению на основе опыта, пониманию и применению абстрактных концепций, а также к использованию своих знаний для управления окружающей средой.

Интеллект - общая способность к познанию и решению проблем человека, которая объединяет все его познавательные способности: ощущение, восприятие, память, представление, мышление, воображение, внимание, волю и рефлексию.

Даже столь поверхностное описание базового прототипа показывает, что изучаемое понятие является синтетическим набором различных способностей человека, которые проявляются в результате внешних негативных воздействий, стимулирующих их преодоление. И, в таком аспекте, вполне естественно искуственный интеллект трактовать как системные технические решения, обеспечивающие преодоление препятствий аналогичной или более высокой сложности. Но прежде чем осуществлять такую трактовку, следует учесть, что способности человеческого интеллекта имеют свойства разного качества:

- *технически реализуемые* свойства: ощущение, восприятие, память, воображение, представление, внимание и волю;
- *мышление* — свойство, требующее дополнительного рассмотрения;
- *рефлексия* — не является качеством технических систем, включая компьютеры.

Таким образом, начинает вырисовываться более содержательная картина, ограничивающая класс искуственных систем, отсутствием у них такого свойства как ***рефлексия***. Это, в свою очередь, ставит под вопрос наличия, содержания и качества негативных воздействий на искуственные системы, являющиеся неотъемлемым условием проявления интеллекта.

Другая выделенная часть интеллекта, требующая рассмотрения, обозначена как свойство ***мышления***.

Мышление — процесс познания окружающего реального мира, основу которого составляет образование и непрерывное пополнение запаса понятий и представлений, включающих в себя вывод новых суждений (осуществление умозаключений).

В таком аспекте, мышление позволяет получить знание о таких объектах, свойствах и отношениях окружающего мира, которые не могут быть непосредственно восприняты при помощи первой сигнальной системы живых организмов.

Сам процесс познания окружающего мира разделен на четыре стадии мышления: подготовка, созревание, озарение и проверка истинности. А по уровню абстракции выделены:

- *наглядно-действенное* мышление - манипуляция предметной сферой, имеющаяся у детей с рождения;
- *конкретно-предметное* мышление, использующее существующие реальные объекты;
- *наглядно-образное* мышление, когда образы представляются в кратковременной и оперативной памяти;
- *абстрактно-логическое* мышление, осуществляемое категориями, которых нет в природе.

В целом, общие характеристики мышления как психического процесса, представлены на рисунке 9.2, заимствованном из .

Рисунок 9.2 - Общие характеристики мышления [9]

Теперь, проецируя выделенные характеристики процесса мышления на компьютерные технологии, можно смело утверждать, что наиболее подходящими для исследования и реализации в искуственном интеллекте являются свойства из группы мыслительных операций: анализ, синтез, сравнение, абстрагирование, обобщение и конкретизация. К ним можно добавить методы дедукции и индукции.

Обоснованием такого утверждения является тот факт, что компьютер в своей основе остается вычислителем и его возможности проецируются через функциональные преобразования.

Теперь, возвращаясь к вопросу об искусственном интеллекте, остановимся на определении систем искуственного интеллекта (СИИ), которое заимствовано из [8, стр. 12]: «СИИ - это компьютерная, креативная система (многофункциональная, интегрированная, интеллектуальная) со сложной структурой, использующая накопление и корректировку знаний (синтаксической, семантической, прагматической информации) для постановки и достижения цели (целенаправленного поведения), адаптации к изменениям среды и внутреннего состояния путем изменения среды или внутреннего состояния.».

Такое определение наглядно демонстрирует состояние дел в изучаемом вопросе, поскольку в само определение (как знание) дается через скрытую рекурентную зависимость, определенную через наличие знания. В общем случае, такую ситуацию можно рассматривать как неопределенную по отношению к самому определению СИИ, трактуя технологии СИИ как дополнение к уже рассмотренным («классическим») компьютерным технологиям.

Не отрицая научной и практической значимости технологий СИИ, отметим основной негативный аспект, присущий им: значительная избыточность формализации, последующей интерпретации и реализации таких систем.

Одной из причин этого негативного аспекта является изначальная трактовка понятия ***искуственный интеллект***, как альтернативы понятию ***человеческого интеллекта***. Такая направленность исследований требует наделения СИИ описанием «*внешнего мира*», дающееся в абстрактных логических понятиях, которые сами являются элементами «*абстрактных миров*» математики.

Следует также отметить, что явление избыточности описания существует и в более простых технологиях. Например, объектно-ориентированные технологии сталкиваются с избыточностью описания классов, в результате чего на практике — все заявленные объекты представляются лишь минимально необходимыми для решения задач проекциями свойств реальных объектов.

9.3 Робототехника

Завершим краткое рассмотрение интеллектуальных систем и технологий одним из известных направлений, называемых ***робототехника***. Формально эти системы не относятся к интеллектуальным системам или системам искусственного интеллекта, но представляют значительный интерес, как область их прямого применения.

Робототехника (***robotics*** — ***роботика***) [10] — прикладная наука, занимающаяся разработкой автоматизированных технических систем и являющаяся важнейшей технической основой интенсификации производства. Робототехника опирается на такие дисциплины, как электроника, механика, кибернетика, телемеханика, мехатроника, информатика и электротехника. Выделяют строительную, промышленную, бытовую, медицинскую, авиационную и экстремальную (военную, космическую, подводную) робототехнику.

Излишне напоминать, что именно робототехника, в контексте слова «робот», появилась в 20-х годах 19-го века, благодаря чешскому писателю Карелу Чапеку. Таким образом, еще задолго до появления компьютерных технологий стала инициировать внимание к системам искусственного интеллекта.

В целом, робототехника разделяется на два больших класса роботов широкого назначения: ***манипуляционные*** и ***мобильные***.

Манипуляционный робот — автоматическая машина, состоящая из исполнительного устройства в виде ***манипулятора***, имеющего несколько степеней подвижности, и устройства программного управления, которые служат для выполнения в производственном процессе двигательных и управляющих функций.

Мобильный робот — автоматическая машина, у которой имеется ***движущееся шасси*** с автоматически управляемыми приводами. Такие роботы могут быть колёсными, шагающими и гусеничными или другими.

Как прикладная наука, робототехника представляет интерес как область область применения всех рассмотренных нами технологий, которые традиционно отображаются на задачи управления такими системами. По типу управления эти системы подразделяются на :

1. Биотехнические:
   - командные (кнопочное и рычажное управление отдельными звеньями робота);
   - копирующие (повтор движения человека, возможна реализация обратной связи, передающей прилагаемое усилие, экзоскелеты);
   - полуавтоматические (управление одним командным органом, например, рукояткой всей кинематической схемой робота);
2. Автоматические:
   - программные (функционируют по заранее заданной программе, в основном предназначены для решения однообразных задач в неизменных условиях окружения);
   - адаптивные (решают типовые задачи, но адаптируются под условия функционирования);
   - интеллектуальные (наиболее развитые автоматические системы);
3. Интерактивные:
   - автоматизированные (возможно чередование автоматических и биотехнических режимов);
   - супервизорные (автоматические системы, в которых человек выполняет только целеуказательные функции);
   - диалоговые (робот участвует в диалоге с человеком по выбору стратегии поведения, при этом как правило робот оснащается экспертной системой, способной прогнозировать результаты манипуляций и дающей советы по выбору цели).

В последние годы значительно расширились сферы применения роботов. Кроме ***промышленных роботов***, появились :

Медицинские роботы

Прежде всего - различные модели хирургических роботов. Так, *в 1985 году* робот ***Unimation Puma 200*** был использован для позиционирования хирургической иглы при выполнении биопсии головного мозга, проводившейся под управлением компьютера. *В 1992 году*, разработанный в ***Имперском колледже Лондона***, робот ***ProBot*** впервые осуществил операцию на предстательной железе. *В 2000 году*, компания ***Intuitive Surgical*** начала серийный выпуск роботов ***Da Vinci***, предназначенных для лапароскопических операций.

Бытовые роботы

Одним из первых примеров удачной массовой промышленной реализации бытовых роботов стала механическая собачка ***AIBO*** корпорации ***Sony***. В сентябре 2005года, в продажу поступили первые человекообразные роботы производства фирмы Mitsubishi. Робот стоимостью в 15 тысяч долларов способен узнавать лица, понимать некоторые

фразы, давать справки, выполнять некоторые секретарские функции, следить за помещением. *В начале XXI века* получили популярность роботы-уборщики, представляющие собой по сути автоматические пылесосы, способные самостоятельно прибраться в квартире и вернуться на место для подзарядки без участия человека.

Роботы для обеспечения безопасности

Роботы широко применяются полицией, органами государственной безопастности, аварийными спасательными службами и другими ведомствами. *В 2007 году*, в городе Перми прошли первые испытания российского роботаполицейского Р-БОТ 001, разработанного московской компанией «Лаборатория Трёхмерного Зрения».

Боевые роботы

Боевым роботом называют автоматическое устройство, заменяющее человека в боевых ситуациях или при работе в условиях, несовместимых с возможностями человека, в военных целях: разведка, боевые действия, разминирование и другие.

Роботы-учёные

Первые роботы-учёные ***Адам*** и ***Ева*** были созданы в рамках проекта ***Robot Scientist*** университета Аберистуита. Один из них, *в 2009 году*, совершил первое научное открытие, связанное с генной инженерией.

Роботы-учителя

Один из первых образцов робота-учителя был разработан, *в 2016 году*, молодыми учеными Томского политехнического универсистета.

Можно и дальше продолжить рассмотрение подбных примеров, демонстрирующих еще далеко не исчерпанные возможности такого научного направления как робототехника.

Завершая учебный материал даного раздела и всей дисциплины, отметим, что интерпретация достижений современных компьютерных технологий еще полностью не завершена. Это связано с ценностными критериями, которыми оперирует научное сообщество и общество, в целом.

Кризис политических систем, который демонстрируется на протяжении последних десятилетий, показывает, что успехи компьютерных технологий не являются исключительно положительным явлением в современной истории.

9.4 Контрольные вопросы

1. В чем состоит назначение теста Тьюринга?
2. Что такое интеллектуальная система?
3. Что такое интеллектуальная информационная система (ИИС)?
4. Какие основные направления входят в ИИС?
5. Какими способностями обладает человеческий интеллект?
6. Какие уровни абстракции обеспечивает человеческое мышление?
7. Какие мыслительные операции обеспечивает человеческое мышление?
8. Какие известны два класса роботов общего назначения?
9. Какие типы управления известны для робототехнических систем?
10. Какие известны сферы применения роботов?