МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

И.Ю. Коцюба, Чунаев А.В., А.Н. Шиков

# Основы проектирования информационных систем

## Учебное пособие


Санкт-Петербург

2015

Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Основы проектирования информационных систем. Учебное пособие. – СПб: Университет ИТМО, 2015. – 206 с.

Учебное пособие содержит краткий конспект лекций по дисциплине «Основы проектирования информационных систем». Теоретический материал состоит из шести глав, представляющих основные сведения о современных технологиях проектирования информационных систем.

Пособие разработано для студентов направлений подготовки 09.03.02 “Информационные системы и технологии” (бакалавриат), 09.04.02 “Информационные системы и технологии” (магистратура)..

Рекомендовано к печати решением совета Естественнонаучного факультета

(Протокол № 1 от 29 января 2015 года).


**Университет ИТМО** – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

## Оглавление

# Введение

Современные компании и организации функционируют в условиях большого объема постоянно изменяющейся информации, которую необходимо оперативно анализировать и принимать правильные решения. Бурно развивается вычислительная техника и информационные технологии. Трудно найти сейчас компанию, не занимающуюся развитием информационных технологий. Современные руководители фирм полностью отдают себе отчет в том, что в настоящее время успешность и прибыльность компании полностью зависят в том числе, и от уровня развития IT-технологий, скорости и качества обработки информации, обоснованности и взвешенности принимаемых решений.

Требуется постоянная серьезная работа не только IT-специалистов, но и топ-менеджеров по согласованию или точнее – синхронизации всех усилий по стратегическому развитию компании и её информационных систем. Большой ошибкой является позиция руководителей компаний, которые, внедрив однажды информационную систему, перестают ею заниматься. Поэтому процесс проектирования информационных систем в настоящее время становиться обязательным. В данном случае, если этот процесс не впервые осуществляется компанией, то термин проектирование приравнивается к понятию развитие информационной системы. Этим объясняется бурное развитие технологий

проектирования информационных систем (ИС) в последние годы. Прежде всего, создание CASE-технологий, которые во много сокращают сроки проектирования ИС, позволяют организовать одновременную коллективную работу, оперативно вносить изменения и быстро реагировать на изменение обстановки на предприятии.

Особое место занимают современные информационные технологии ведения электронной коммерции, работа с заказчиками и поставщиками. И в этом направлении проектирование и развитие информационных систем не возможно без знания основных методологий и программных средств, позволяющих в кратчайшие сроки и без ошибок управлять этими процессами. Компании получают колоссальные конкурентные преимущества, если уровень развития информационных систем соответствует уровню развития предприятия. Улучшается инвестиционная привлекательность компании, основные бизнес-процесс становятся прозрачными и понятными для контроля и управления, исключаются ошибки, брак и связанные с ними потери и времени, и средств, в конечном счете, увеличивается прибыль компании.

Актуальность и важность дисциплины «Основы проектирование информационных систем» определяется необходимостью изучения теоретических положений, связанных с нормативно-технической документацией на разработку и проектирование ИС, управление жизненным циклом

ИС, архитектурой ИС, внедрением и сопровождением ИС, а так же получения практических навыком разработки основных проектных документов, моделирования и анализа бизнес-процессов, применения современных CASE-средств.

Настоящее учебное пособие содержит полный перечень теоретических сведений по дисциплине «Основы проектирование информационных систем».

# 1.Основные понятия техногии проектирования информационных систем

## 1.1. Основные понятия и определения

Основные понятия в последние годы не претерпели сильных изменений, формулировки стали более точными и лаконичными, исключающими неоднозначность понятий. Наиболее полные определения представлены в Федеральных законах Российской Федерации и стандартах.

***Информация*** – «сведения (сообщения, данные) независимо от формы их представления»[1].

***Информационные технологии*** – «процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов» [1].

***Информационная система*** – «совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств» [1].

***Проектирование информационных систем*** – это упорядоченная совокупность методологий и средств создания или модернизации информационных систем.

***Управление информационными системами*** – «применение методов управления процессами планирования, анализа, дизайна, создания, внедрения и эксплуатации информационной системы организации для достижения ее целей»[4].

***Жизненный цикл информационных системы*** – «развитие рассматриваемой системы во времени, начиная от замысла и кончая списанием» [2].

***Модель жизненного цикла*** – «структурная основа процессов и действий, относящиеся к жизненному циклу, которая служит в качестве общей ссылки для установления связей и взаимопонимания сторон» [2].

***Архитектура информационных систем*** – это концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы.

***Бизнес-процесс*** – это цепочка взаимосвязанных действий, направленных на создание товарной продукции или услуги.

***Регламент бизнес-процесса*** – это четко определенный порядок выполнения бизнес-процесса, определяющий состав и действия участников.

***Модель данных*** – это система организации данных и управления ими.

***Методология проектирования информационных систем*** – это совокупность принципов проектирования (моделирования), выраженная в определенной концепции.

***Средства моделирования*** – это программы описания и моделирования систем.

***Типовое проектное решение (ТПР)*** – это многократно используемое проектное решение.

***Нотации*** – это определенные способы представления элементов информационной системы.

***Реинжиниринг бизнес-процессов*** – это фундаментальная реорганизация бизнес-процессов с целью повышения их эффективности.

***Системный подход*** – процесс рассмотрения любой системы в качестве совокупности взаимосвязанных элементов.

***Процессный подход*** – представление любой системы в качестве совокупности процессов.

***Функциональный подход*** – предусматривает четкое закрепление за каждой структурной единицей набора функций.

***Техническое задание*** – документ, используемый заказчиком в качестве средства для описания и определения задач, выполняемых при реализации договора [3].

## 1.2. Исторические аспекты развития технологий проектирования информационных систем

Середина прошлого столетия ознаменовалась началом активного развития информационных технологий. Прежде всего, военные ведомства и передовые предприятия многих стран понимают важность и ценность создания и развития информационных систем. С появлением вычислительной техники обработка больших объемов информации и автоматизация ос-

новных производственных процессов и органов управления на всех уровнях становятся наиважнейшей задачей для обеспечения военного превосходства наиболее развитых государств и конкурентных преимуществ коммерческих компаний. Разработчики национальных и крупномасштабных информационных систем стали первыми осознавать необходимость создания специальных средств проектирования и моделирования бизнес-процессов, позволяющими сделать их работу более эффективной и сократить не только сроки создания информационных систем, но минимизировать ошибки. Ошибки и неточности встречаются постоянно, чем раньше они диагностируется и локализуется, тем меньше стоимость переделки. Известно, что стоимость выявление и устранение ошибки на стадии проектирования в два раза обходится дороже, на стадии тестирования информационной системы в десять раз, а на стадии эксплуатации в сто раз, чем на стадии анализа бизнес-процессов и разработки технического задания.

При создании сложных информационных систем зачастую очень трудно понять требования персонала заказчика. Они могут быть сформулированы некорректно, а в процессе анализа конкретных бизнес-процессов даже измениться. Поэтому появление методологий современного проектирования и моделирования информационных систем было насущной задачей, над которой работали специалисты разных стран.

Появление автоматизированных систем управления в шестидесятых годах прошлого столетия определялось получением начальных знаний и опыта

их разработки и внедрения. Анализировались все успехи и неудачи создания первых АСУ, но бесспорным было сокращение времени обработки информации, производственных и управленческих затрат и как следствие персонала.

Опыт зарубежных компаний по разработке и внедрению корпоративных информационных систем свидетельствует о появлении программ, в первую очередь связанных с автоматизацией учетных функций бухгалтерий, отдела кадров и складов. И намного позже появляются другие автоматизированные системы управления производством, логистикой, взаимоотношениями с клиентами и поставщиками. На последнем этапе разрабатываются информационные системы управления всей компанией, позволяющие полностью перейти к электронному документообороту и автоматизировать все сферы деятельности фирмы. С появлением персональных компьютеров происходит децентрализация процессов управления, все чаще внедряются модули с распределенными системами обработки информации.

На следующем этапе, в семидесятых годах прошлого столетия пришло понимание, что информация – стратегический ресурс любой компании, который необходимо грамотно использовать. В тоже время главными потребителями информации являются руководители. Идеи использования распределенных систем не находят пока применения из-за отсутствия компактной вычислительной техники, которая появится позднее и перевернет весь мир. В компаниях и организациях создаются информационные отделы и службы, вычислительные центры и лаборатории.

Восьмидесятые годы характеризуются появлением специализированных методологий проектирования информационных систем и CASE-средств. На их основе разрабатываются первые программные средства, а персональные компьютеры позволяют приступить к созданию децентрализованных информационных систем. Различные персональные компьютеры объединяются в локальную сеть. Этот период характеризуется интеграций информационных систем и появлением различных концепций управления ими на единой методологической основе.

Девяностые годы стали триумфом персональных компьютеров. Невысокая стоимость и компактные размеры сделали их чрезвычайно популярными и общедоступными для индивидуализации использования при решении управленческих задач. Разрабатываются корпоративные информационные системы, реализующие принципы распределенной обработки данных. Становится возможным автоматизация всех отделов и служб компаний, а не только бухгалтерии. Появляются системы электронного документооборота, в том числе для предприятий с развитой филиальной сетью в разных городах и регионах. Сокращаются сроки обработки данных, производственных, складских и прочих управленческих отчетов.

Появление и развитие методологий моделирования и проектирования информационных систем не было простым процессом. На всех этапах этого пути были талантливые, энергичные, необычайно трудолюбивые люди, которые вкладывали свои знания, силы, опыт, а порой и денежные

средства для успешной реализации информационных проектов. Вот некоторые из них:

В СССР основателем и теоретиком автоматизированных систем управления был выдающийся ученый академик В.М. Глушков. Под его руководством в 1963-1964 годах в Институте кибернетики Академии наук СССР были начаты работы по созданию автоматизированных систем сбора, учета, обработки данных, оперативно-календарного планирования производства на базе отечественной вычислительной техники. При этом ставилась задача разработки универсальной АСУ, пригодной для внедрения на многих предприятиях страны. Опытная эксплуатация и апробация проходили на самых современных предприятиях государства, таких как Львовский телевизионный завод «Электрон» и Кунцевский радиозавод. Некоторые решения были признаны и за рубежом, так была предложена общая алгоритмическая схема последовательного анализа вариантов, включавшая в себя вычислительные методы динамического программирования (В. С. Михалевича и Н. З. Шора). В.В. Шкурба развил эту схему вместе с методами имитационного моделирования для решения задач упорядочения, в частности в теории расписаний и календарного планировании.

В 1970-1980-х годах информационные системы интегрировались в комплексные АСУ, решающие задачи автоматизированного проектирования новых изделий, технологической подготовки производства и автоматизации организационного управления предприятием. Комплексные АСУ были разработаны и внедрены на Ульяновском авиационном заводе и других предприятиях оборонного комплекса под руководством

А. А.Морозова, В. И. Скурихина. Комплексные АСУ создавались научно-исследовательскими институтами такими как, Всесоюзного объединения "Союзсистемпром" Минприбора СССР: ЦНИИТУ, г. Минск; ГНИПИ ВТ, г. Казань; НИИУМС, г. Пермь и др.

В США Дуглас Т. Росс (SoftTech, Inc) разработал язык АПТ для работы станков с ЧПУ, который в средине 60-х положил начало разработкам графических языков моделирования. А в 1969 году им же предложена методология SADT (IDEF0) для моделирования информационных систем средней сложности, в рамках программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой департаментом Военно-Воздушных Сил США в рамках большого аэрокосмического проекта.

К концу двадцатого века было разработано несколько десятков методов моделирования сложных систем. Они все были разные по функциональным возможностям, но во многом имели схожие подходы к анализу и описанию предметной области. Возникла острая необходимость объединения удачных решений в одну методику, которая устраивала большую часть разработчиков информационных систем. В результате этих процессов был разработан язык UML.

У многих ведущих компаний, таких как Rational Software, Oracle Corporation, IBM, Microsoft, Hewlett-Packard, i-Logix, Texas Instruments и Unisys была четкая уверенность в необходимости создания подобного языка программирования. С этой целью был создан консорциум UML Partners, рабочую группу по семантике UML возглавил Крис Кобрин. Ведущую роль в

создании унифицированного языка моделирования (UML) сыграли Гради Буч, Айвар Джекобсон и Джеймс Рамбо, работающие в компании Rational Software. Ими разработаны следующие методы объектного моделирования сложных информационных систем:

1. Метод объектного моделирования программного обеспечения сложных информационных систем (метод Буча).
2. Метод анализа требований к бизнес-приложениям (метод Джекобсона).
3. Метод анализа обработки данных в сложных информационных системах (метод Рамбо).

Предварительная версия 0.8 унифицированного метода программирования была выпущена в октябре 1995 года. Первая версия UML 0.9 вышла в июне 1996 году и получила мощную поддержку Группы OMG, занимающейся разработкой единых стандартов в сфере web-технологий, включающую в себя несколько сотен компаний, работающих в области IT-технологий и производстве компьютерной техники. Это позволило выпустить в первый года сразу несколько версий. Так, в 1997 году появляется сразу две версии UML 1.0 и UML 1.1. В 1998 году разработчиками представляется версия UML 1.2, в 1999 году версия UML 1.3, в 2001 году выходит версия UML 1.4, а в 2003 году версия UML 1.5. Эта версия и принимается в качестве международного стандарта ISO/IEC 19501-2005.

Сейчас наиболее популярна версия UML 2.4.1, вышедшая в 2011 году, которая тоже оформлена виде международных стандартов ISO/IEC 19505-1 и 19505-2. Для нее разработаны инструментальные средства

поддержки и визуального программирования, осуществляющих прямую генерацию кода из моделей UML, в основном посредством языков программирования C++ и Java. Среди них программы Rational Rose и Visual Paradigm for UML.

В настоящее время методологии и средства моделирования бизнес-процессов, процессно-ориентированных методов анализа и проектирования информационных систем широко представлены как в России, так и в большинстве стран мира.

## 1.3. Процессы и модели жизненного цикла информационных систем

В соответствии с ГОСТ Р ИСО/МЭК 12207-99 процессы жизненного цикла включают себя работы, которые могут выполняться в жизненном цикле программных средств, распределены по пяти основным, восьми вспомогательным и четырем организационным процессам [9].

### *Основные процессы жизненного цикла [9]*

Основные процессы жизненного цикла состоят из пяти процессов, которые реализуются под управлением основных сторон, вовлеченных в жизненный цикл программных средств. Под основной стороной понимают одну из тех организаций, которые инициируют или выполняют разработку, эксплуатацию или сопровождение программных продуктов. Основными сторонами являются заказчик, поставщик, разработчик, оператор и персонал сопровождения программных продуктов. Основными процессами являются:

**1. Процесс заказа.** Определяет работы заказчика, то есть организации, которая приобретает систему, программный продукт или программную услугу.

**2. Процесс поставки**. Определяет работы поставщика, то есть организации, которая поставляет систему, программный продукт или программную услугу заказчику.

**3. Процесс разработки.** Определяет работы разработчика, то есть организации, которая проектирует и разрабатывает программный продукт.

**4. Процесс эксплуатации.** Определяет работы оператора, то есть организации, которая обеспечивает эксплуатационное обслуживание вычислительной системы в заданных условиях в интересах пользователей.

**5. Процесс сопровождения.** Определяет работы персонала сопровождения, то есть организации, которая предоставляет услуги по сопровождению программного продукта, состоящие в контролируемом изменении программного продукта с целью сохранения его исходного состояния и функциональных возможностей. Данный процесс охватывает перенос и снятие с эксплуатации программного продукта.

### *Вспомогательные процессы жизненного цикла [9]*

Вспомогательные процессы жизненного цикла состоят из восьми процессов. Вспомогательный процесс является целенаправленной составной частью другого процесса, обеспечивающей успешную реализацию и качество выполнения

программного проекта. Вспомогательный процесс, при необходимости, инициируется и используется другим процессом. Вспомогательными процессами являются:

1. **Процесс документирования.** Определяет работы по описанию информации, выдаваемой в процессе жизненного цикла.
2. **Процесс управления конфигурацией.** Определяет работы по управлению конфигурацией.
3. **Процесс обеспечения качества**. Определяет работы по объективному обеспечению того, чтобы программные продукты и процессы соответствовали требованиям, установленным для них, и реализовывались в рамках утвержденных планов. Совместные анализы, аудиторские проверки, верификация и аттестация могут использоваться в качестве методов обеспечения качества.
4. **Процесс верификации.** Определяет работы (заказчика, поставщика или независимой стороны) по верификации программных продуктов по мере реализации программного проекта.
5. **Процесс аттестации.** Определяет работы (заказчика, поставщика или независимой стороны) по аттестации программных продуктов программного проекта.
6. **Процесс совместного анализа.** Определяет работы по оценке состояния и результатов какой-либо работы. Данный процесс может использоваться двумя любыми сторонами, когда одна из сторон (проверяющая) проверяет другую сторону (проверяемую) на совместном совещании.

**7. Процесс аудита.** Определяет работы по определению соответствия требованиям, планам и договору. Данный процесс может использоваться двумя сторонами, когда одна из сторон (проверяющая) контролирует программные продукты или работы другой стороны (проверяемой).

**8. Процесс решения проблемы**. Определяет процесс анализа и устранения проблем (включая несоответствия), независимо от их характера и источника, которые были обнаружены во время осуществления разработки, эксплуатации, сопровождения или других процессов.

***Организационные процессы жизненного цикла [9]***

Организационные процессы жизненного цикла состоят из четырех процессов. Они применяются в какой-либо организации для создания и реализации основной структуры, охватывающей взаимосвязанные процессы жизненного цикла и соответствующий персонал, а также для постоянного совершенствования данной структуры и процессов. Эти процессы, как правило, являются типовыми, независимо от области реализации конкретных проектов и договоров; однако уроки, извлеченные из таких проектов и договоров, способствуют совершенствованию организационных вопросов. Организационными процессами являются:

**1. Процесс управления.** Определяет основные работы по управлению, включая управление проектом, при реализации процессов жизненного цикла.

**2. Процесс создания инфраструктуры.** Определяет основные работы по созданию основной структуры процесса жизненного цикла.

**3. Процесс усовершенствования.** Определяет основные работы, которые организация (заказчика, поставщика, разработчика, оператора, персонала сопровождения или администратора другого процесса) выполняет при создании, оценке, контроле и усовершенствовании выбранных процессов жизненного цикла.

**4. Процесс обучения.** Определяет работы по соответствующему обучению персонала.

Данный ГОСТ Р ИСО/МЭК 12207-99 охватывает жизненный цикл программных средств от концепции замыслов через определение и объединение процессов для заказа и поставки программных продуктов и услуг. Кроме того, данная структура предназначена для контроля и модернизации данных процессов.

Процессы, определенные в настоящем стандарте, образуют множество общего назначения. Конкретная организация, в зависимости от своих целей, может выбрать соответствующее подмножество процессов для выполнения своих конкретных задач. Поэтому настоящий стандарт следует адаптировать для конкретной организации, проекта или приложения. Настоящий стандарт предназначен для использования как в случае отдельно поставляемых программных средств, так и для программных средств, встраиваемых или интегрируемых в общую систему [9].

## *Модели жизненного цикла информационной системы*

Модель жизненного цикла информационной системы может включать:

1. Стадии.
2. Основные результаты выполнения работ на каждой стадии.
3. Ключевые события.

Под стадией понимается определенный этап процесса разработки информационной системы.

Жизненный цикл информационной системы характеризуется периодом времени от идеи создания информационной системы и заканчивая моментом вывода ее из эксплуатации и включает в себя следующие стадии:

1. Предпроектное обследование.
2. Проектирование.
3. Создание информационной системы.
4. Ввод в эксплуатацию.
5. Эксплуатация информационной системы.
6. Вывод из эксплуатации.

Процессы жизненногоцикла информационных систем представлены на рис.1.

Процессы жизненного цикла, определенны стандартом ГОСТ Р ИСО/МЭК 15288-2005, могут применяться любой организацией при приобретении и использовании или создании и поставке системы. Они распространяются на любой уровень системной иерархии и на любую стадию жизненного цикла.

Процессы жизненного цикла основываются на принципах модульности (максимальная слаженность

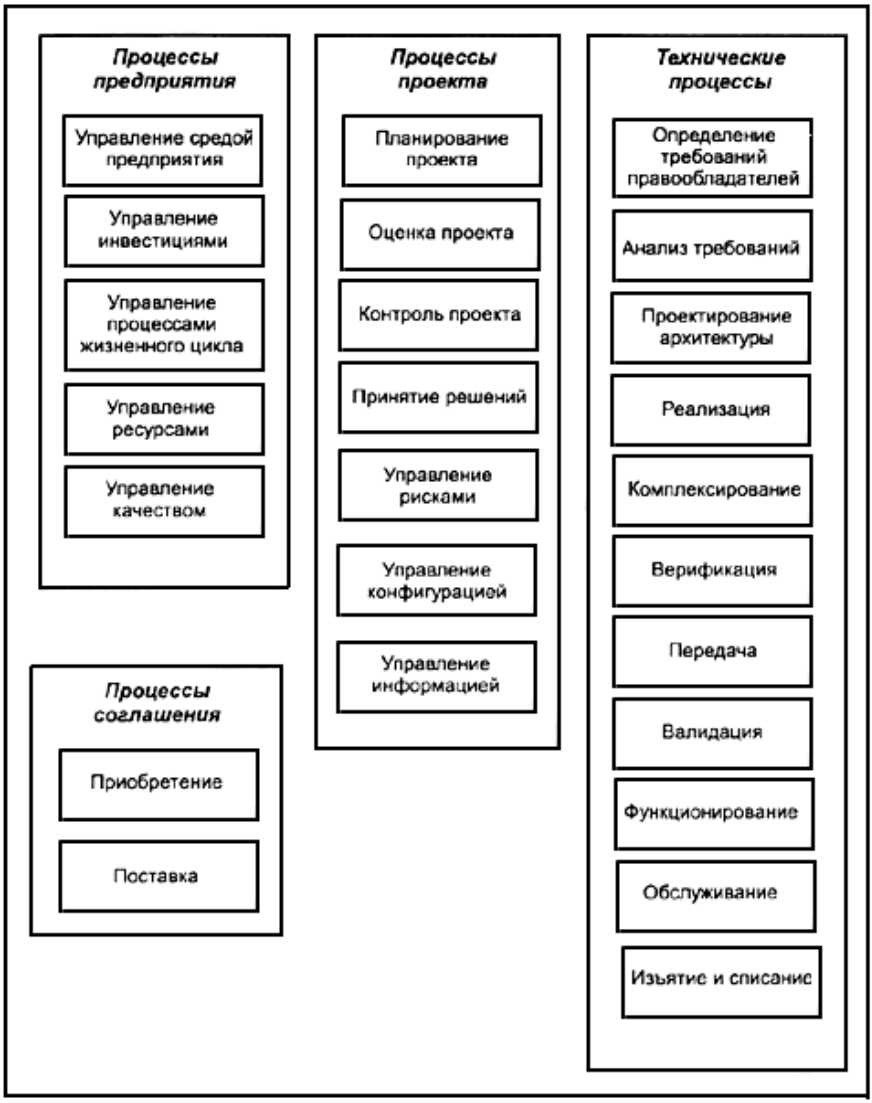



Рисунок 1 - Процессы жизненного цикла информационной системы [11]

функций процесса и минимальная связь между процессами) и собственности (процесс связывается с ответственностью). Функции, которые осуществляются данными процессами, определяются в зависимости от конкретных целей, результатов и набора действий, составляющих данный процесс. Процессы, описанные в настоящем стандарте, не препятствуют и не исключают использование дополнительных процессов, которые организация посчитает необходимыми [11].

***Управление процессами жизненного цикла ИС [11]***

Цель управления процессами жизненного цикла системы заключается в гарантировании доступности эффективных процессов жизненного цикла для использования организацией. Данный процесс обеспечивает процессы жизненного цикла системы, которые согласованы с целями и политикой организации, определены, адаптированы и поддержаны соответствующим образом для учета особенностей отдельных проектов и способны реализовываться с помощью эффективных проверенных методов и инструментальных средств.

В результате эффективного управления процессами жизненного цикла системы:

a) определяются процессы жизненного цикла системы, которые будут использоваться организацией;

b) определяется политика применения процессов жизненного цикла системы;

c) определяется политика адаптации процессов жизненного цикла системы для удовлетворения потребностей отдельных проектов;

d) определяются критерии оценки результатов применения процессов жизненного цикла системы;

e) предпринимаются действия по совершенствованию способов определения и применения процессов жизненного цикла системы.

При реализации процессов управления процессами жизненного цикла системы организация должна осуществлять следующие действия в соответствии с принятой политикой и процедурами:

a) устанавливать стандартные наборы процессов жизненного цикла систем для соответствующих стадий жизненного цикла системы;

b) определять приемлемые политику и процедуры адаптации и требования к их утверждению;

c) определять методы и инструментальные средства, которые поддерживают выполнение процессов жизненного цикла системы;

d) по возможности устанавливать показатели, которые позволяют определять характеристики выполненных стандартных процессов;

e) контролировать выполнение процесса, сохранять и анализировать показатели процесса и определять тенденции по отношению к критериям предприятия;

f) определять возможности для усовершенствования стандартных процессов жизненного цикла систем;

g) совершенствовать имеющиеся процессы, методы и инструментальные средства, используя найденные возможности.

Сравнительные данные стадий жизненногоцикла и этапов ИС представлены в таблице 1.

Таблица 1

| **Классический ЖЦ** | **ИСО / МЭК 12207** | **ГОСТ 34.601-90 и ОРММ ИСЖТ 5.03-00** | | |
|---|---|---|---|---|
| | | **Стадия** | | **Основные этапы (работы)** |
| Системный анализ | Заказ | Формирование требований к ИС | Технико-экономическое обоснование (ТЭО) | 1.Обследование объекта и обоснование необходимости создания ИС.<br>2.Формирование требований Заказчика к ИС.<br>3.Оформление договора между Разработчиком и Заказчиком. |

| | | | | |
|---|---|---|---|---|
| Анализ требований | Разработка | Разработка концепции ИС (для комплексных многоуровневых и интегрированных систем) | ТЭО | 1. Поиск путей удовлетворения требований Заказчика на уровне концепции создаваемой системы (структура, функции, программно-техническая платформа, режимы).<br>2. Рассмотрение альтернативных вариантов концепции системы, их анализ и выбор лучшей концепции. |
| Проектирование | | Техническое задание (ТЗ) | | Разработка, согласование и утверждение ТЗ на создание ИС. |
| | | Эскизный проект (для комплексных многоуровневых и интегрированных систем) | | Разработка предварительных проектных решений по системе и ее частям. |
| | | Пилот-проект (макетирование, прототипирование) (при необходимости) | | 1. Разработка частей проекта для испытаний в реальных, но ограниченных условиях функционирования с целью проверки предварительно принятых решений.<br>2. Проведение испытаний на головном объекте или стенде и анализ результатов испытаний. |
| | | Технический проект | | 1. Разработка проектных решений по системе и ее частям.<br>2. Разработка документации на ИС и ее части.<br>3. Разработка документации на поставку изделий для комплектования ИС и/или технических заданий на их разработку.<br>4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации (строительство, монтаж, наладка и др.). |

| | | | |
|---|---|---|---|
| Кодирование (реализация) | | Рабочая документация | 1. Разработка рабочей документации на систему и ее части.<br>2. Разработка программных и технических средств и/или адаптация приобретаемых.<br>3. Тестирование средств. |
| Тестирование | | Интеграция и тестирование | 1.Загрузка БД типовыми исходными данными и тестами.<br>2. Интеграция программ и тестирование в имитированной среде.<br>3. Интеграция программных средств с аппаратными в реальной операционной и внешней среде.<br>4. Тестирование в реальной среде.<br>5. Разработка комплекта документации для пользователей. |
| Внедрение и сопровождение | Разработка и эксплуатация | Ввод в действие на головном объекте (ввод в эксплуатацию, внедрение) | 1. Подготовка объекта автоматизации к вводу ИС в действие.<br>2. Подготовка персонала.<br>3. Комплектация ИС поставляемыми изделиями.<br>4. Проведение предварительных испытаний и передача ИС для опытной эксплуатации.<br>5. Проведение опытной эксплуатации.<br>6. Проведение приемочных испытаний по сдаче ИС в постоянную эксплуатацию. |
| | | Тиражирование (при внедрении на нескольких объектах) | 1. Передача эталона загрузочных модулей ПО и эксплуатационной документации в группу сопровождения.<br>2.Тиражирование документации.<br>3. Обучение и консультации пользователей.<br>4. Поставка ПО и документации на объекты внедрения. |

| | | | |
|---|---|---|---|
| | Сопровождение и эксплуатация | Сопровождение (авторский надзор) | 1. Выполнение работ в соответствии с гарантийными обязательствами.<br>2. Оказание научно-технических услуг в послегарантийный период.<br>3. Разработка методики оформления отчетов об ошибках и предложениях на изменение версий. |

Каждая система имеет свой жизненный цикл. Жизненный цикл может быть описан с использованием абстрактной функциональной модели, представляющей концептуализацию потребности в системе, ее реализации, применения, развития и ликвидации [11].

Система развивается на протяжении жизненного цикла в результате действий, осуществляемых и управляемых людьми, работающими в организациях и использующими определенные процессы в своей деятельности. Детали модели жизненного цикла выражаются в терминах этих процессов, их результатов, взаимосвязи и возникновения. Настоящий стандарт определяет множество процессов, названных процессами жизненного цикла, при помощи которых может быть смоделирован жизненный цикл системы.

Жизненные циклы различаются по свойствам, целям, использованию системы, а также по преобладающим условиям. Тем не менее, несмотря на очевидное множество различий в жизненных циклах систем, существует базовый

набор стадий жизненного цикла, составляющих полный жизненный цикл любой системы. Каждая стадия имеет определенную цель и вклад в полный жизненный цикл и рассматривается при планировании и выполнении жизненного цикла системы [11].

Стадии представляют собой основные периоды жизненного цикла, связанные с системой и относящиеся к состоянию описания системы или непосредственно к системе. Стадии отображают значимый прогресс и достижение запланированных этапов развития системы на протяжении всего жизненного цикла и дают начало важнейшим решениям относительно своих входов и выходов. Эти решения используются организациями для учета неопределенностей и рисков, непосредственно связанных с затратами, сроками и функциональностью при создании или применении системы. Таким образом, стадии обеспечивают организации структурой работ, в рамках которых управление предприятием обладает высокой способностью для обзора и контроля проекта и технических процессов.

Организации проходят стадии жизненного цикла различными способами, устраняя противоречия между стратегией осуществления бизнеса и стратегией уменьшения рисков. Параллельное прохождение стадий или их прохождение в различном порядке может привести к формам жизненного цикла с совершенно разными характеристиками. Часто в качестве альтернативных вариантов используются

последовательная, инкрементная или эволюционная формы жизненного цикла; в отдельных случаях могут быть разработаны комбинации этих форм. Выбор и разработка организацией конкретных форм жизненного цикла зависят от ряда факторов, включая бизнес-контекст, природу и сложность системы, стабильность требований, технологические возможности, потребность в различных системных возможностях во времени и наличие бюджетных средств и ресурсов [11].

Аналогично тому, как все системные элементы осуществляют вклад в систему как в единое целое, так и каждая стадия жизненного цикла должна учитываться на любой другой ее стадии. Следовательно, участвующие стороны должны координировать свои действия и кооперироваться друг с другом на протяжении всего жизненного цикла. Синергия стадий жизненного цикла и сторон, вкладывающих средства в реализацию функциональностей на этих стадиях, является необходимой для успешного осуществления проектных мероприятий. Тесная связь и, по возможности, единение проектных команд, различных функций и организаций, ответственных за другие стадии жизненного цикла, приводят к логичности и согласованности жизненного цикла [11].

Набольшее распространение получили следующие модели жизненного цикла информационных систем (ИС): каскадная (классическая или водопадная), итерационная и спиральная.

## *Каскадная (классическая, водопадная) модель жизненного цикла информационной системы*

Модель была предложена в 1970 году Уинстоном Ройсом. Переход на следующий этап осуществляется после полного окончания работ по предыдущему этапу, при этом оформляется полный комплект рабочей документации. Все этапы выполняются в строгой последовательности с утвержденными сроками и четкими затратами. Это основные достоинства каскадной модели ЖЦ ИС, которая применялась в условиях полной определенности решаемых задач и совершенно не приемлема когда и разработчики и заказчики не имеют четкого видения всех особенностей проектируемой ИС. Кроме того, невозможно идти дальше, пока не сдан предыдущий этап, а после сдачи нельзя возвращаться к нему для устранения обнаруженных недочетов, что серьезно затрудняет работы по совершенствованию и доработке создаваемой ИС. Эта модель нравиться и заказчикам, и разработчикам по причине жесткой дисциплины финансирования этапов только после их предъявления. Но полностью отсутствует гибкость в работе над созданием ИС. Каскадная модель представлена на рис.2.

На практике, все же приходится возвращаться к предыдущим этапам и в этом случае, в последнее время наиболее востребованной стала итерационная модель ЖЦ ИС.


Рисунок 2 – Каскадная (водопадная, классическая) модель ЖЦ ИС

### *Итерационная модель жизненного цикла ИС*

**Поэтапная модель с промежуточным контролем** — итерационная модель разработки информационной системы. Каждый этап имеет обратные связи в процессе корректировки и создает условия для корректировки ранее созданных этапов. При этом трудоемкость работ и временные затраты существенно сокращаются посравнению с водопадной моделью жизненного цикла. Итерационная модель ЖЦ информационной системы представлена на рис.3.

Создание информационной системы – это организованный процесс построения и последовательного преобразования согласованных моделей на всех этапах жизненного цикла. При этом все разработанные модели находятся в репозитории проекта и доступны всем раз-


Рисунок 3 – Итерационная модель ЖЦ ИС

работчикам, что позволяет эффективно вести одновременную работу над проектированием и созданием информационной системы.

### *Спиральная модель жизненного цикла информационной системы*

**Спиральная модель** предложена Барри Боэм в 1988 году и определяет, в основном стартовые этапы жизненного цикла информационной системы. При этом обосновывается и проверяется возможность реализации спроектированных технических решений. На каждом витке создается прототип проектируемой информационной системы, который на следующих витках спирали ЖЦ ИС совершенствуется, дополняется и доводится до полного внедрения. При этом не

обязательно дожидаться окончания каждого этапа, данная модель позволяет переходить на следующие витки спирали и решать проблемы или недоделки на следующем уровне, что делает работу над проектом более эффективной, гибкой и завершить в более сжатые сроки.

Новый виток спирали соответствует поэтапной модели создания фрагмента информационной системы. При использовании спиральной модели ЖЦ:

- происходит ориентация на модернизацию информационной системы;
- осущесвляется аккумулирование всех решений в процессе проектирования и создания моделей и прототипов информационной системы;
- проводитсяов анализ издержек и всех рисков в процессе проектирования ИС.

Спиральный процесс состоит из следующей повторяющейся последовательности:

1. Определение требований.
2. Анализ.
3. Проектирование.
4. Реализация и тестирование.
5. Интеграция.
6. Внедрение.

Этот многократный цикл, завершающийся созданием новой версии информационной системы представлен на рис. 4.

Для применения спиральнй модели ЖЦ ИС может быть несколько причин, это необходимисть минимизации рисков и возможность представления заказчику прототип или эскизную версию проекта для

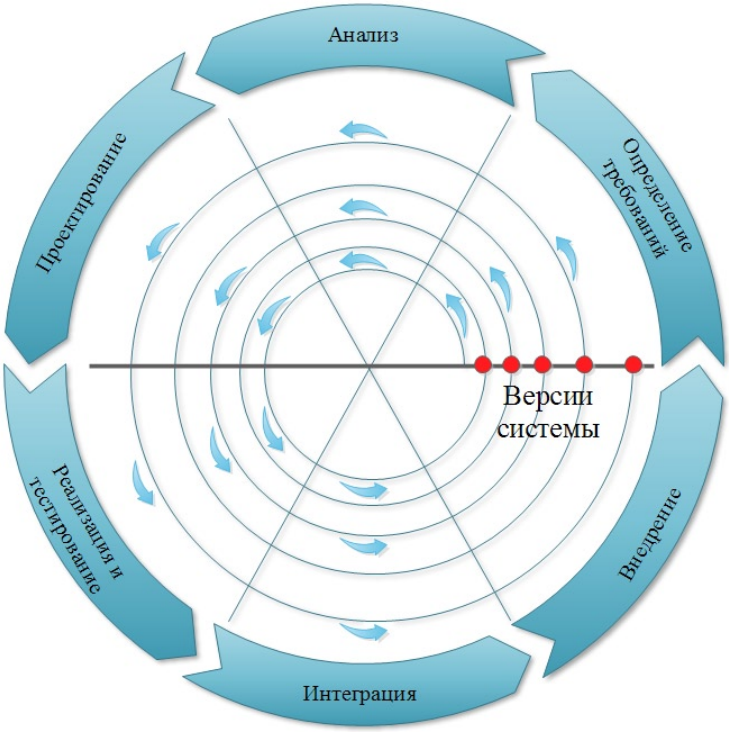



Рисунок 4 – Спиральная модель ЖЦ ИС

конкретизации пожеланий и учета их в следующих циклах. А так же в случае если разрабатываемая информационная система достаточно сложна и существует реальная необходимость создавать промежуточные версии продукта, не откладывая эту работу на финишные этапы, как это предписывает водопадная модель.

Основная задача спиральной модели жизненного цикла информационной системы заключается в том, чтобы на каждой итерации создавать очередную

версию системы, используя разработанный прототип предыдущих этапов. Такая модель позоляет более гибко работать с заказчиком, постоянно учитывать его замечания и предложения, совершенствовать проектируемую систему в процессе каждого нового витка спирали.

## 1.4 Основные методологии современного проектирования информационных систем

### *Методология функционального моделирования работ SADT*

Методология SADT (Structured Analysis and Design Technique - методология структурного анализа и проектирования), разработанная Дугласом Т. Россом в 1969-1973 годах базируется на структурном анализе систем и графическом представление организации в виде системы функций, которые имеют три класса структурных моделей:

1. Функциональная модель.
2. Информационная модель.
3. Динамическая модель.

Процесс моделирования по методологии SADT состоит из следующих этапов:

1. Сбор информации и анализ информации о предметной области.
2. Документирование полученной информации.
3. Моделирование (IDEF0).
4. Корректура модели в процессе итеративного рецензирования.

Методология в настоящее время более известна как нотация IDEF0, использует формализованный процесс моделирования информационных систем и имеет следующие стадии: анализ, проектирование, реализация, объединение, тестирование, установка, функционирование. Проектирование информационных систем по стандарту IDEF0 сводится к декомпозиции основных функций организации на отдельные бизнес-процессы, работы или действия. В результате разрабатывается иерархическая модель анализируемой организации, при этом декомпозицию можно проводить многократно, до четкого и детального описания всех процессов. Диаграммы IDEF0 верхнего уровня принято называть родительскими, а нижнего уровня – дочерними. Пример диаграммы IDEF0 верхнего уровня представлен на рис. 5.

Анализируемый процесс представляется в виде прямоугольника. Слева изображаются входные данные, справа – выходные, сверху управляющие или регламентирующие воздействия, а снизу объекты управления. В диаграмме IDEF0 описываются первоначально все внешние связи исследуемого процесса. После этого осуществляется декомпозиция этого процесса и происходит описание внутренних подпроцессов с обозначением всех связей. При этом ранее обозначенные стрелочками внешние связи не должны потеряться. Они переносятся на диаграмму декомпозиции в соответствующие подпроцессы. Пример декомпозиции диаграммы IDEF0 (дочерней) представлен на рис.6.

Далее каждый подпроцесс тоже можно декомпозировать и подробно описывать все связи до

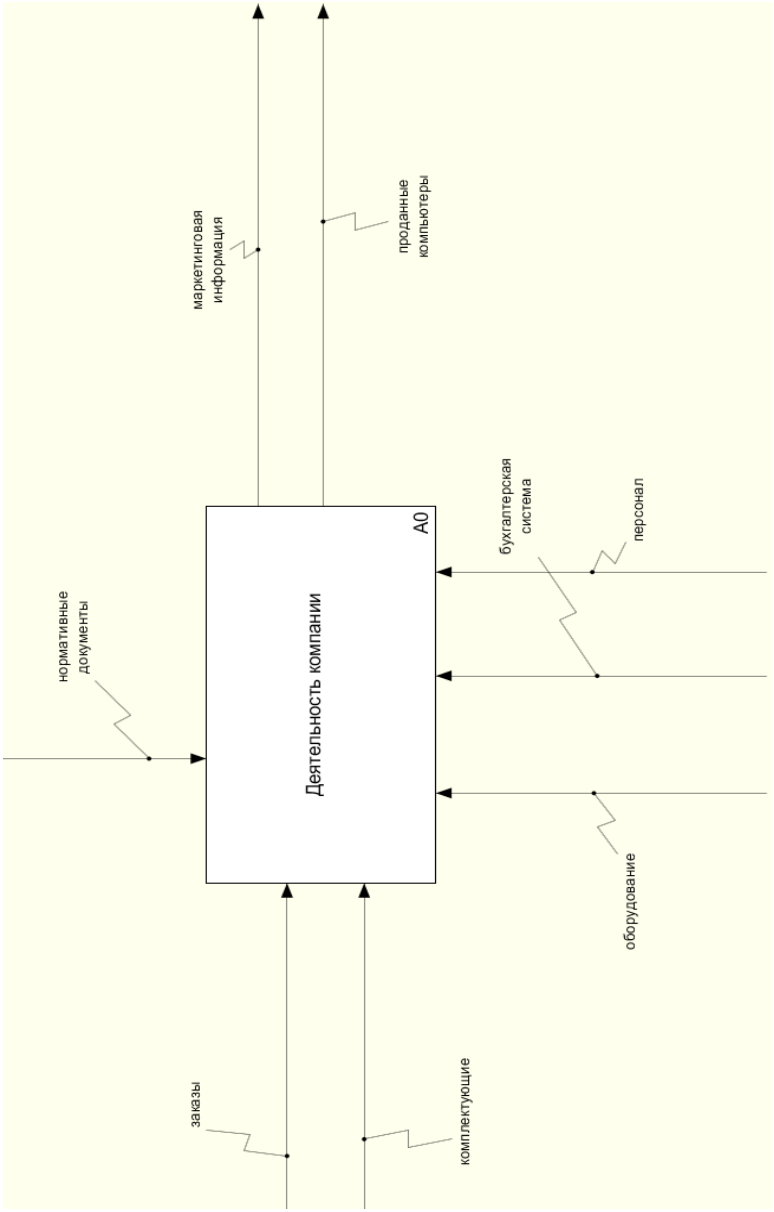



Рисунок 5 - Диаграмма IDEF0 верхнего уровня

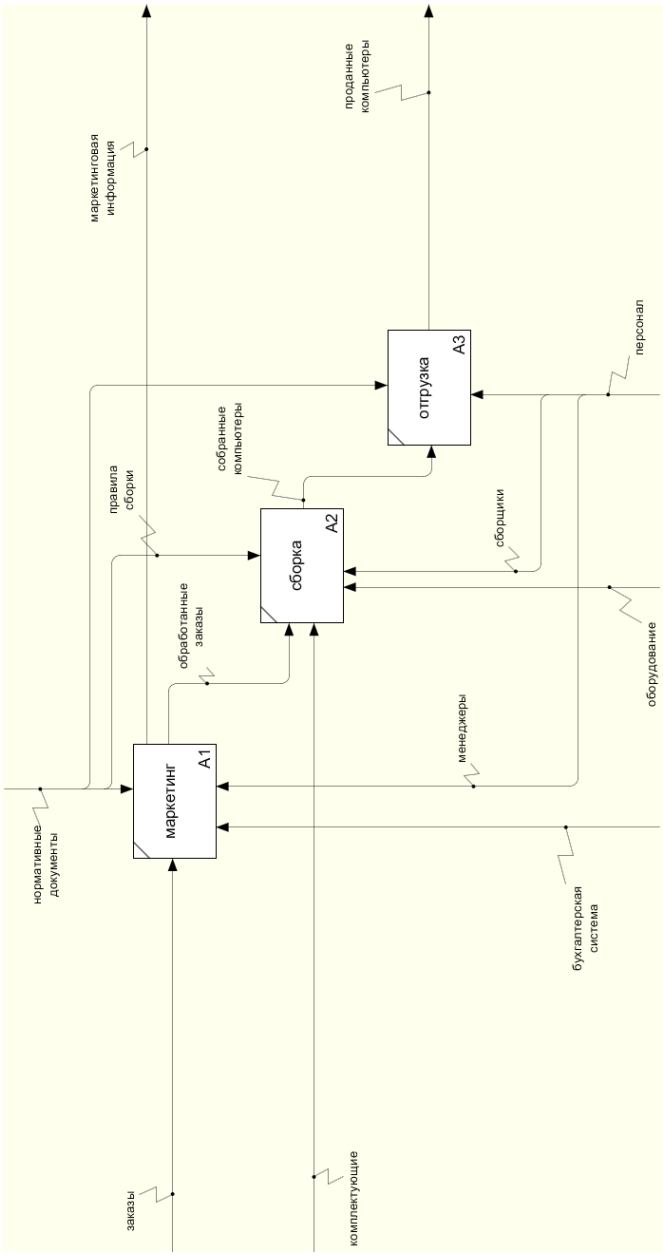



Рисунок 6 – Дочерняя диаграмма IDEF0 (декомпозиция)

необходимого предела. Основным дростоинством этой методологии являются простота и наглядность. В качестве недостатка – невозможность описать реакцию описываемого процесса на изменяющиеся внешние факторы. Для этих целей служат другие методологии.

### *Методология RAD – быстрой разработки приложений*

Принципы RAD сформулированы в 1980 году сотрудником компании IBM Джеймсом Мартином. Они базировались на идеях Скотта Шульца и Барри Бойма при этом методология реализовывалась в кратчайшие сроки небольшой группой разработчиков с использованием инкрементного прототипирования. Это позволяло на ранней стадии проектирования ИС продемонстрировать заказчику действующую интерактивную модель системы-прототипа, уточнить проектные решения, оценить эксплуатационные характеристики.

В настоящее время **методология RAD** стала общепринятой схемой для проектирования и разработки информационных систем. Средства разработки, основанные на RAD, очень популярны за счет использования таких программных сред разработки: IBM Lotus Domino Designer, Borland Delphi, Borland C++ Builder, Microsoft Visual Studio, Macromedia Flash и др.

В методологии RAD быстрая разработка приложений достигается за счет использования компонентно-ориентированного конструирования и применяется если:

- Бюджет проектируемой информационной системы ограничен.
- Нечетко определены требования к информационной системе.
- Требуется реализация проекта информационной системы в минимальные сроки.
- Интерфейс пользователя можно продемонстрировать в прототипе.
- Проект можно разделить на составляющие элементы по функциональному назначению.

Методология RAD имеет следующие стадии:

1. Моделирование информационных потоков между бизнес-функциями.
2. Моделирование данных.
3. Преобразование объектов данных, обеспечивающих реализацию бизнес-функций.
4. Генерация приложений.
5. Тестирование и объединение.

Недостатки методологии RAD:

1. Для больших информационных систем требуются большой коллектив разработчиков.
2. Применима для информационных систем, которые могут декомпозироваться на отдельные модули и в которых производительность не является критической величиной.
3. Не используется в случае применения новых технологий.

### *Методология RUP*

Среди всех фирм-производителей CASE-средств компания IBM Rational Software Corp. (до августа 2003

года – это самостоятельная фирма Rational Software Corp.) одна из первых осознала стратегическую перспективность развития объектно-ориентированных технологий анализа и проектирования программных систем. Эта компания выступила инициатором унификации языка визуального моделирования в рамках консорциума OMG, что привело к появлению первых версий языка UML. Эта же компания первой разработала инструментальное объектно-ориентированное CASE-средство, в котором был реализован язык UML, как базовая нотация визуального моделирования. Графическое представление методологии RUP из Википедии изображено на рис. 7 .

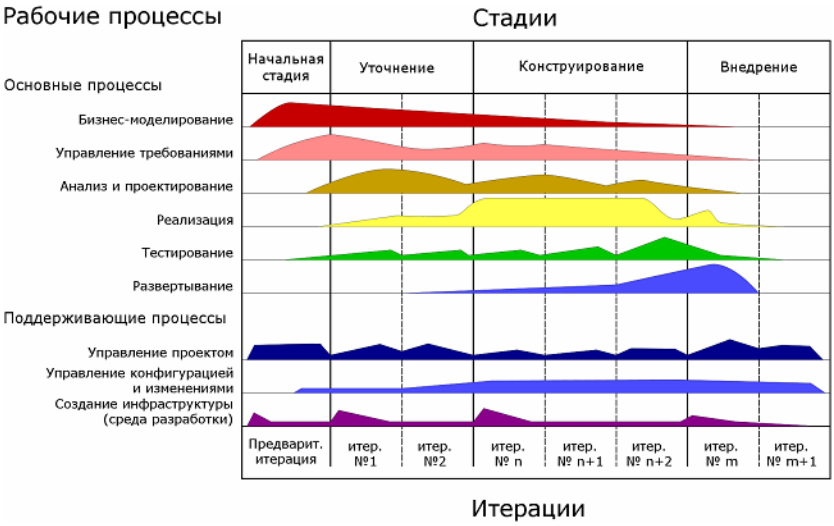



Рисунок 7 – Представление методологии RUP (Википедия)

Одна из самых популярных технологий - Rational Unified Process (RUP). В определенном

плане эта методология становится международным стандартом, разработанный компанией Rational Software, которая в настоящее время входит в состав IBM. Авторами UML считаются сотрудники фирмы Rational Software: Гради Буч, Айвар Якобсон, Джемс Рамбо. RUP полностью соответствует стандартам, определяющим проектные работы в процессе жизненного цикла информационных систем. В методологии RUP реализуются следующие подходы:

1. Итерационный и инкрементный (наращиваемый).
2. Построение системы на базе архитектуры информационной системы.
3. Планирование и управление проектом на основе функциональных требований к информационной системе.

Разработка информационной системы выполняется итерациями. Это отдельные прокты небольшие по объему и содержданию, которые включают свои собственные этапы анализа требований, проектирования, реализации, тестирования, интеграции. Заканчиваются итерации созданием работающей информационной подсистемы.

Итерационный цикл характеризуется периодической обратной связью и может адаптироваться к ядру разрабатываемой системы. Создаваемая информационная система постепенно растет и совершенствуется.

# 2. Организация проектирования информационных систем

## 2.1 Каноническое проектирование информационных систем

Каноническое проектирование ИС направлено на отражение особенностей технологии индивидуального (оригинального) проектирования. Среди основных характерных особенностей канонического проектирования можно выделить такие особенности, как:

- отражение особенностей ручной технологии проектирования;
- ориентация на индивидуальное (оригинальное) проектирование;
- осуществление на уровне исполнителей;
- возможность интеграции выполнения элементарных операций;
- применение, как правило, для сравнительно небольших, локальных ИС;
- использование инструментальных средств универсальной компьютерной поддержки.

Каноническое проектирование направлено на минимальное использование типовых проектных решений. Адаптация проектных решений при каноническом проектировании осуществляется только путем перепрограммирования соответствующих программных модулей.

Организация канонического проектирования ИС основана на использовании каскадной модели жизненного цикла и предусматривает набор

определенных стадий и этапов. Принцип деления процесса проектирования на стадии и этапы направлен на то, чтобы проектировать систему «сверху-вниз» и постепенно разрабатывать - изначально укрупненные, затем детализированные – проектные решения.

Поскольку объекты автоматизации имеют различную сложность и набор задач для создания решения для конкретной ИС, стадии и этапы работ также могут различаться по трудоемкости: существует возможность объединять последовательные этапы, исключать определенные из них на любой стадии проекта, а также до окончания предыдущей стадии начинать выполнение следующей.

Стадии и этапы разработки ИС, которые выполняют организации-участники, оформляются в договорах и технических заданиях на выполнение работ.

**Каноническое проектирование** основано на ряде стандартов, таких, как:

1. ГОСТ 34.003 – термины и определения основных понятий в области автоматизированных систем;
2. ГОСТ 34.201 – виды, комплектность и обозначение документов при создании автоматизированных систем;
3. ГОСТ 34.601 – стадии создания автоматизированных систем;
4. ГОСТ 34.602 – техническое задание на создание ИС;
5. ГОСТ 34.603 – виды испытаний автоматизированных систем;

6. РД 50-34.698 – требования к содержанию документов;

7. ГОСТ 2.105 – общие требования к текстовым документам.

По отношению к проекту разработки ИС можно выделить 3 укрупненные стадии проектирования:

- предпроектную (стадии 1-3);
- проектную (стадии 4-6);
- послепроектную (стадии 7-8).

## 2.2 Стадии и этапы процесса канонического проектирования ИС

Стадии и этапы создания ИС, выполняемые организациями-участниками, фиксируются в договорах и технических заданиях на выполнение работ.

**Предпроектная** стадия направлена на предпроектное обследование и разработку технического задания на ИС. Характерными результатами этого этапа являются: определение целей и задач системы, формирование общих требований к ее созданию, разработка программы проведения обследования, в ходе которого должны быть изучены структура и бизнес-процессы организации, модель управления, задачи, подлежащие автоматизации, технико-экономические характеристики, ориентировочных состав технических средств.

Перечень 8 этапов работ (стадий), в соответствии с ГОСТ 34.601 и дополнительными пояснениями, представлен ниже:

Стадия 1. Формирование требований к ИС.

- обследование объекта и обоснование необходимости создания ИС;

- формирование требований пользователя к ИС;
- оформление отчета о выполненной работе и заявки на разработку ИС (ТТХ).

Стадия 2. Разработка концепции ИС.

- изучение объекта;
- проведение необходимых научно-исследовательских работ;
- разработка вариантов концепции ИС, удовлетворяющих требованиям пользователей;
- оформление отчета о проделанной работе.

Стадия 3. Техническое задание.

- разработка и утверждение технического задания на создание ИС.

Важным документом, фиксирующим результаты определения стратегии внедрения ИС, является технико-экономическое обоснование проекта. В этом документе должно быть четко определены результаты выполнения проекта для заказчика, а также указаны графики выполнения работ и график финансирования на разных этапах выполнения проекта. Дополнительно в таком документе отражаются сроки, время окупаемости проекта, ожидаемые выгода и экономический эффект проекта.

Ориентировочно технико-экономическое обоснование содержит:

- все риски и ограничения, влияющие на успешность проекта;

- условия эксплуатации будущей системы: архитектурные, программные, аппаратные требования, требования к компонентам ПО и СУБД;
- пользователи системы;
- функции, выполняемые системой;
- интерфейсы и распределение функций между человеком и системой;
- сроки завершения этапов, форма приемки/ сдачи работ;
- рамки проекта;
- возможности развития системы.

По результатам обследования формируется **техническое задание на информационную систему.**

В соответствии с ГОСТ 34.602-89, **техническое задание (ТЗ)** – основной документ, определяющий требования и порядок создания (развития или модернизации) автоматизированной системы, в соответствии с которым проводится разработка ИС и ее приемка при вводе в действие.

Разработка технического задания предусматривает описание следующих разделов:

- общие сведения;
- назначение и цели создания (развития) системы;
- характеристика объектов автоматизации;
- требования к системе;
- состав и содержание работ по созданию системы;
- порядок контроля и приемки системы;
- требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;

- требования к документированию;
- источники разработки.

**Проектная** стадия главным образом ориентирована на разработку технического и рабочего проектов. Процесс разработки технического задания включает обследование объекта автоматизации (организации или подразделения) и его систем управления. Для решения задач информационного обеспечения необходимо проанализировать информационные потоки, формы документации, системы кодирования, а также все связанное со структурой баз данных и СУБД, что определяет состав исходных технологических требований.

Стадия 4. Эскизный проект.

- разработка предварительных проектных решений по системе и ее частям;
- разработка эскизной документации на ИС и ее части.

Если для ИС конкретного объекта автоматизации проектные решения выбраны ранее или являются очевидными, стадия эскизного проекта может быть исключена из последовательности работ. Таким образом, эта стадия не является строго обязательной.

На этапе эскизного проекта, в том числе, должны быть определены:

- цели, функции ИС и подсистем;
- состав комплексов задач и отдельных задач;
- концепция и структура информационной базы;

- функции СУБД;
- функции и параметры основных программных средств;
- ожидаемый эффект от ее внедрения.

Документация, содержащая результаты работ по совокупности принятых проектных решений, согласовывается, утверждается и используется в дальнейшем для выполнения работ по созданию ИС.

На основании технического задания (в т.ч., при наличии эскизного проекта) разрабатывается технический проект ИС.

Стадия 5. Технический проект.

- разработка проектных решений по системе и ее частям;
- разработка документации на ИС и ее части;
- разработка и оформление документации на поставку изделий для комплектования ИС и (или) технических требований (технических заданий) на их разработку;
- разработка заданий на проектирование в смежных частях проекта объекта автоматизации.

На этапе технического проекта проводятся работы научно-исследовательского и экспериментального характера для выбора основных проектных решений, а также рассчитывается экономическая эффективность системы.

Важным аспектом разработки **технического проекта** является анализ всей используемой информации на предмет таких характеристик, как

полнота, отсутствие дублирования и избыточности, непротиворечивость и т.д., а также определение форм выходных документов. Документация должна быть оформлена в соответствии с требованиями ГОСТ 34-201 и РД 50-34.698.

Стадия 6. Рабочая документация.

- разработка рабочей документации на систему и ее части;

- разработка или адаптация программ.

Один из основных этапов стадии **рабочего проектирования** – разработка рабочей документации на информационное обеспечение ИС, в состав которой входят: технический проект ИС, описание баз данных, перечень исходных и выходных данных и документов.

Стадия технического проектирования завершается подготовкой и оформлением документации на поставку для комплектования ИС и определением технических требований и составлением ТЗ на разработку ИС.

Стадия «Рабочая документации» предполагает создание, как программного продукта, так и всей сопровождающей документации, которая должна предоставлять все сведения, обеспечивающие выполнение работ на стадиях ввода ИС в действие и эксплуатации ИС, в том числе, сведения для поддержания уровня качества ИС (соблюдения эксплуатационных характеристик).

**Послепроектная** стадия включает в себя реализацию мероприятий по внедрению, подготовку помещений и технических средств,

обучение персонала. Также производится эксплуатация системы с решением конкретных задач, анализируются результаты испытаний, реализуются мероприятия по сопровождению ИС.

Стадия 7. Ввод в действие.

- подготовка объекта автоматизации к вводу ИС в действие;
- подготовка персонала;
- комплектация ИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями);
- строительно-монтажные работы;
- пусконаладочные работы;
- проведение предварительных испытаний;
- проведение опытной эксплуатации;
- проведение приемочных испытаний.

Основными видами испытаний для ИС являются такие, как: предварительные испытания, опытная эксплуатация и приемочные испытания, которые при необходимости могут быть расширены дополнительными испытаниями ИС и ее составляющих частей.

В ходе предварительных испытаний, регламентируемых соответствующей программой и методикой, главным образом проводятся испытания системы на работоспособность и соответствие ТЗ, а также устранение неисправностей и внесение изменений в документацию на ИС.

На следующем этапе происходит процесс проведения опытной эксплуатации, анализируются ее

результаты, и при необходимости проводится доработка ПО и дополнительная наладка технических средств ИС.

В процессе проведения приемочных испытаний реализуются испытания на соответствие ТЗ, анализируются результаты испытания системы, устраняются недостатки, которые были выявлены при испытаниях.

При всех видах испытаний оформляются соответствующие акты о приемке системы в опытную эксплуатацию, ее завершении и о приемке системы в постоянную эксплуатацию.

Стадия 8. Сопровождение ИС.

- выполнение работ в соответствии с гарантийными обязательствами;
- послегарантийное обслуживание.

Основными процессами этой стадии являются осуществление работ по устранению недостатков, выявленных при эксплуатации системы в течение гарантийных сроков, а также анализ функционирования системы, выявление отклонений и их причин, устранение причин отклонений и недостатков, обеспечение стабильности эксплуатационных характеристик.

## 2.3 Типовое проектирование ИС, типовое проектное решение (ТПР)

Методы типового проектирования направлены на выполнение проектирования ИС с использованием типовых проектных решений.

**Типовое проектное решение** – проектное решение, пригодное к многократному использованию (тиражируемое проектное решение).

Применение методов типового проектирования имеет свои особенности. Основным условием для использования таких методов является возможность декомпозиции проектируемой ИС на составляющие компоненты (подсистемы, программные модули, комплексы выполняемых задач и тд), для реализации которых можно выбрать типовые проектные решения, существующие на рынке, которые будут настроены на нужды конкретного предприятия.

Помимо собственно функциональных (программных, аппаратных) элементов, типовое решение подразумевает наличие необходимой документации, в которой дается детальное описание ТПР (в т.ч., процедур настройки), отвечающее требованиям проектируемой системы.

По уровню декомпозиции системы можно выделить такие классы ТПР, как:

- элементные ТПР – ТПР по отдельному элементу (задаче, виду обеспечения);
- подсистемные ТПР – ТПР по отдельным подсистемам;
- объектные ТПР – отраслевые ТПР, включающие весь набор подсистем ИС.

Выделенные классы ТПР имеют свои достоинства и недостатки. Рассмотрим наиболее характерные из них.

К достоинству **элементных ТПР** можно отнести реализацию модульного подхода к проектированию ИС. В то же время это приводит к большим затратам на доработку ТПР конкретных элементов и к затратам на объединение разных элементов вследствие их несовместимости.

**Подсистемные ТПР** также позволяют реализовать модульный подход к проектированию ИС. Кроме того, они позволяют осуществлять параметрическую настройку компонентов на объекты различных уровней управления; взаимосвязанные компоненты и высокая степень интеграции элементов ИС приводят к минимизации затрат на проектирование и программирование. Однако в случае нескольких производителей программного обеспечения появляются проблемы в объединении различных функциональных подсистем; помимо этого, с точки зрения непрерывного реинжиниринга процессов адаптивность ТПР является недостаточной.

**Объектные ТПР** имеют такие преимущества, как:

- масштабируемость (допускаются конфигурации ИС для различного числа рабочих мест);
- методологическое единство компонентов ИС;
- совместимость компонентов ИС;
- открытость архитектуры (возможность развертывания ТПР на платформах разного типа);
- конфигурируемость (возможность использовать необходимое подмножество компонентов системы).

К недостаткам объектных ТПР можно отнести проблемы реализации типового проекта в оригинальном объекте управления, что приводит в определенных ситуациях к необходимости

смены организационной структуры объекта автоматизации.

При реализации типового проектирования применяются такие подходы, как: **параметрически-ориентированное** и **модельно-ориентированное проектирование.**

Этапами **параметрически-ориентированного** проектирования являются:

- постановка задач и определение пригодности пакетов прикладных программ (ППП) для их решения через систему критериев оценки;
- анализ доступных ППП исходя из критериев;
- выбор и приобретение подходящего ППП;
- настройка параметров приобретенного ППП.

Среди критериев оценки ППП выделяют [15] следующие группы:

- назначение и возможности пакета;
- отличительные признаки и свойства пакета;
- требования к техническим и программным средствам;
- документация пакета;
- факторы финансового порядка;
- особенности установки пакета;
- особенности эксплуатации пакета;
- помощь поставщика по внедрению и поддержанию пакета;
- оценка качества пакета и опыт его использования;
- перспективы развития пакета.

Отметим, что каждая из перечисленных групп критериев может быть детализирована на совокупность частных показателей, дающих дополнительную информацию для каждого аспекта анализа выбранного ППП. Значения критериев определяются с использованием методов экспертного оценивания.

Другим подходом реализации типового проектирования является модельно-ориентированное проектирование, сущность которого состоит в адаптации существующих характеристик типовой ИС, исходя из модели объекта автоматизации, построение которой предполагает использование специального программного инструментария.

При таком подходе технология проектирования должна иметь средства как для работы с моделью конкретного предприятия, так и с моделью типовой ИС.

В репозитории типовой ИС содержится модель объекта автоматизации, которая является основой для конфигурирования программного обеспечения. Кроме того, в репозитории содержится базовая (ссылочная) модель ИС и типовая (референтная) модели ее определенных классов.

Базовая модель ИС описывает бизнес-процессы, организационную структуру, бизнес-объекты, бизнес-функции, для поддержки которых предназначены программные модули типовой ИС.

Типовые модели предназначены для описания конфигурации ИС для тех или иных отраслей, типов производства.

Модель конкретного предприятия может быть построена либо в результате выбора фрагментов типовой модели с учетом особенностей объекта автоматизации (BAAN Enterprise Modeler), либо с использованием автоматизированной адаптации этих модулей с учетом мнений экспертов (SAP Business Engineering Workbench). Модель предприятия, на основе которой осуществляется автоматическое конфигурирование и настройка ИС, хранится в репозитории и может быть откорректирована в случае необходимости.

Внедрение типовой ИС начинается с анализа результатов предпроектного обследования предприятия, сформированных в виде требований к конкретной ИС, для оценки которых может быть использована методика оценки ППП. На следующем этапе необходимо построить предварительную модель ИС, которая должна полно отражать особенности реализации ИС для конкретного объекта автоматизации. Предварительная модель – основа для выбора типовой модели системы, а также для формирования перечня компонентов, для реализации которых потребуются другие программные средства или инструментальные средства, имеющиеся в составе типовой ИС.

При реализации типового проекта имеет место выполнение следующих операций [15]:

- установку глобальных параметров системы;
- задание структуры объекта автоматизации;
- определение структуры основных данных;

- задание перечня реализуемых функций и процессов;
- описание интерфейсов;
- описание отчетов;
- настройку авторизации доступа;
- настройку системы архивирования.

Типовое проектирование в настоящее время широко представлено в современных средствах.

# 3. Архитектура информационных систем

## 3.1 Понятие архитектуры информационных систем

Уровень развития современных технологий настолько высок, что позволяет построить информационную систему любого масштаба, сложности и функциональности. Однако, учитывая требования бизнеса, основанные на показателях различных бизнес-оценок, возникают дополнительные сложности, разрешение которых сводится к обеспечению рационального подхода к процессу проектирования, реализации и дальнейшей эксплуатации информационных систем. Исходя из этого, можно однозначно считать выбранную архитектуру одним из основных показателей эффективности создаваемой информационной системы, а, следовательно, и успешности бизнеса.

Изначально термин архитектура применялся в проектировании и постройке различных сооружений. Он определял их структуру, взаимосвязи между составными частями, базовые принципы их организации и дальнейшего развития. С течением времени понимание термина «архитектура», применительно к техническим системам, несколько изменилось. В техническом аспекте можно рассматривать архитектуру, как высоко абстрактную модель, в которой отсутствуют подробности реализации.

Определить понятие «архитектура информационной системы» можно множеством способов. Это связано:

1. С отсутствием общепринятого определения самой информационной системы. Учитывая сложность структуры, достаточным способом описать её возможно только при консолидации нескольких точек зрения, что в каждом конкретном случае может приводить к различным результатам.
2. С многообразием трактовок самого термина «архитектура».

В результате, архитектуру информационной системы можно описать как «концепцию, определяющую модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы» [28].

Процедура выбора архитектуры для проектируемой информационной системы, в рыночных условиях, сводится к определению стоимости владения ею. Стоимость владения информационной системой складывается из плановых затрат и стоимости рисков.

Плановые затраты включают в себя стоимость технического обслуживания, модернизации, зарплату обслуживающего персонала и т.д.

Совокупная стоимость рисков определяется из стоимости всех типов рисков, их вероятностей и матрицей соответствия между ними. Сама же матрица соответствия определяется выбранной архитектурой информационной системы.

Можно выделить наиболее важные типы рисков:

- проектные риски (риски при создании системы);
- риски разработки (ошибки, недостаточная оптимизация);
- технические риски (простои, отказы, утрата данных);
- бизнес-риски (возникают из-за технических рисков и связаны с эксплуатацией системы);
- неопределённости (связаны с вариативностью бизнес-процессов и складываются из необходимости внесения изменений в систему и неоптимальной процедурой функционирования);
- операционные (подразумевают невыполнение набора операций, могут возникать из-за технических рисков и являться инициаторами бизнес-рисков).

Концепция архитектуры информационной системы должна формироваться ещё на этапе технико-экономического обоснования и выбираться такой, чтобы стоимость владения ею была минимальной.

Для того чтобы конструктивно определить архитектуру, необходимо ответить на ряд вопросов:

1. Что делает система?
2. На какие составные часть она разделена?
3. Каким образом происходит взаимодействие этих частей?
4. Как и где эти части размещены?

Таким образом, можно считать архитектуру информационной системы моделью, определяющей стоимость владения через имеющуюся в данной системе инфраструктуру.

## 3.2. Типы архитектур

Рассматривая архитектуру крупных организаций, принято использовать понятие «корпоративная архитектура». Её можно представить в виде совокупности нескольких типов архитектур:

- бизнес архитектура (Business architecture);
- ИТ-архитектура (Information Technology architecture);
- архитектура данных (Data architecture);
- программная архитектура (Software architecture);
- техническая архитектура (Hardware architecture).

Модель корпоративной архитектуры представлена на рис. 8.


Рисунок 8 - Модель корпоративной архитектуры

**Техническая архитектура** является первым уровнем архитектуры информационной системы. Она описывает все аппаратные средства, использующиеся при выполнении заявленного набора функций, а также включает средства обеспечения сетевого взаимодействия и надёжности. В технической архитектуре указываются периферийные устройства, сетевые коммутаторы и маршрутизаторы, жёсткие диски, оперативная память, процессоры, соединительные кабели, источники бесперебойного питания и т.п.

**Программная архитектура** представляет собой совокупность компьютерных программ, предназначенных для решения конкретных задач. Данный тип архитектуры необходим для описания приложений, входящих в состав информационной системы. На данном уровне описывают программные интерфейсы, компоненты и поведение.

**Архитектура данных** объединяет в себе как физические хранилища данных, так и средства управления данными. Кроме того, в неё входят логические хранилища данных, а при ориентированности рассматриваемой компании на работу со знаниями, может быть выделен отдельный уровень – **архитектура знаний** (Knowledge architecture). На этом уровне описываются логические и физические модели данных, определяются правила целостности, составляются ограничения для данных.

Следует особенно выделить уровень **ИТ-архитектуры,** поскольку он является связующим.

На нём формируется базовый набор сервисов, которые используются как на уровне программной архитектуры, так и на уровне архитектуры данных. Если какая-либо особенность функционирования для этих двух уровней не была предусмотрена, то сильно возрастает вероятность сбоев в работе, а, следовательно, потерь для бизнеса. В некоторых случаях невозможно разделить ИТ-архитектуру и архитектуру отдельного приложения. Такое возможно при высокой степени интеграции приложений. Примером ИТ-архитектуры может служить SharePoint от компании Microsoft. Этот продукт предоставляет сервисы для совместной работы и хранения информации, что является очень важным аспектом функционирования любой компании. Его базовые системные модули относятся к ИТ-архитектуре, а пользовательские – к программной. Основной функцией ИТ-архитектуры является обеспечение функционирования важных бизнес-приложений для достижения обозначенных бизнес-целей. Если некоторая функция требуется сразу в нескольких приложениях, то её следует перенести на уровень ИТ-архитектуры, тем самым повысив интеграцию системы и снизить сложность архитектуры приложений.

Последним в иерархии является уровень **бизнес-архитектуры** или **архитектуры бизнес-процессов**. На этом уровне определяются стратегии ведения бизнеса, способы управления, принципы организации и ключевые процессы, представляющие для бизнеса огромную важность.

## 3.3. Микроархитектура и макроархитектура

Термины микроархитектура и макроархитектура в большей степени применяются для описания программных систем. В соответствие с рассмотренной моделью уровней архитектур корпоративных информационных систем, микроархитектуру можно отнести к уровням программной архитектуры и архитектуры данных, а макроархитектуру – к уровню ИТ-архитектуры.

Микроархитектура описывает внутреннее устройство конкретного компонента или подсистемы, а макроархитектура описывает устройство всей ИС, как совокупности её компонент или подсистем.

Сложность программных систем постоянно увеличивается. Это обусловлено ростом объёма передаваемой и обрабатываемой информации, усложнению самих задач по обработке информации и увеличению количества таких задач. Без применения какого-либо архитектурного подхода при построении сложных систем, их создание, обслуживание и модификация, в конце концов, станут нерентабельными для бизнеса.

Для решения данной проблемы можно использовать методы абстракции, декомпозиции, инкапсуляции. Так, при разработке программной системы, например входящей в состав инфраструктуры крупной организации, она представляется в виде множества модулей, каждый из которых выполняет определённую функцию, а все вместе они выполняют функции самой системы. В данном случае, организация каждого модуля будет являться микроархитектурой, а спосо-

бы взаимодействия между этими модулями в рамках системы – макроархитектурой.

Уменьшение сложности реализации системы будет происходить за счёт разбиения сложных задач на несколько более простых. В результате, это может привести к появлению большого набора простых задач, на основании которого можно будет реализовать любую, даже очень сложную. В этом заключается принцип объектно-ориентированного программирования – создание конкретных классов объектов для решения конкретных задач.

К сожалению, не всегда сформированная система классов сможет упростить процессы управления программной системой. К примеру, подсистема выполняет слишком большой спектр действий необходимых для различных процессов в организации и выход её из строя нарушает функционирование сразу нескольких их них.

Существуют два принципа, позволяющие оценить взаимное влияние компонентов системы друг на друга:

- low coupling (слабая связанность);
- high cohesion (сильное зацепление).

Принцип Low Coupling способствует распределению функций между компонентами системы таким образом, чтобы степень связанности между ними оставалась низкой. Степень связанности (coupling) — это мера взаимозависимости подсистем. Данный принцип связан с одним из основных принципов системного подхода, который требует минимизации информационных потоков между подсистемами.

Подсистема с низкой степенью связанности (или слабым связыванием) имеет следующие свойства:

- малое число зависимостей между подсистемами;
- слабая зависимость одной подсистемы от изменений в другой;
- высокая степень повторного использования подсистем.

В свою очередь, принцип High Cohesion задаёт свойство сильного зацепления внутри подсистемы. В результате подсистемы получаются сфокусированными, управляемыми и понятными.

Зацепление (функциональное зацепление) — это мера связанности и сфокусированности функций подсистемы. Подсистема обладает высокой степенью зацепления, если её функции тесно связаны между собой, и она не выполняет больших объемов работы.

Подсистема с низкой степенью зацепления выполняет множество различных функций никак не связанных между собой. Такие подсистемы создавать нежелательно, поскольку они приводят к возникновению следующих проблем:

- трудность понимания.
- сложность при повторном использовании.
- сложность поддержки.
- ненадежность, постоянная подверженность изменениям.

Подсистемы с низкой степенью зацепления не имеют чёткого функционального назначения и выполняют слишком разноплановые функции, которые можно легко распределить между другими подсистемами.

Следует заметить, что связанность является характеристикой системы целиком, а связность характеризует отдельно взятую подсистему.

Связанность (coupling) и связность (cohesion) являются общесистемными характеристиками и могут применяться при проектировании любых систем.

## 3.4. Архитектурный подход к проектированию ИС

Процесс проектирования информационной системы тесным образом связан с её архитектурным описанием, что отражено в некоторых определениях термина «архитектура».

Можно выделить пять различных подходов к проектированию:

- Календарный подход.
- Подход, за основу которого взят процесс управления требованиями.
- Подход, основанный на процессе разработки документации.
- Подход, в основе которого лежит система управления качеством.
- Архитектурный подход.

**Календарный подход** подразумевает составление графика предстоящих работ с их поэтапным выполнением. Следует отметить, что ключевые решения принимаются на основании локальных задач и целей каждого конкретного этапа разработки. Также при данном подходе практически не уделяется времени на разработку документации, формированию

архитектур и процессов по внесению различных изменений. В долгосрочной перспективе из-за этого возрастает стоимость владения, разработанной таким образом системы. Такой стиль считается морально устаревшим, однако в некоторых компаниях до сих пор применяется.

Подход, за основу которого взят процесс **управления требованиями**, большую часть времени всего процесса разработки выделяет на функциональные характеристики системы, а нефункциональные, такие как масштабируемость, например, практически не рассматриваются. Все решения в ходе проекта формируются исходя из локальных целей по реализации конкретного функционала. Такой подход может быть эффективен, если требования к разрабатываемой системе определены заранее и не изменяются в процессе проектирования. В качестве недостатков можно выделить несоответствие стандарту качества ISO 9126 и нестабильность разрабатываемых архитектур, поскольку каждая реализуемая функция связывается с одним или несколькими компонентами. В связи с этим, трудоёмкость при добавлении к подобным системам дополнительных функций возрастает. Применение этого подхода в долгосрочной перспективе является нерациональным, однако позволит успешно управлять требованиями в рамках требуемой функциональности.

При применении подхода, основанного на процессе **разработки документации**, неоправданно большое количество времени тратится на формирования пакета документов, которые часто не ис-

пользуются ни заказчиком, ни пользователем. Кроме того, из-за нехватки времени страдает качество самой разрабатываемой системы. Данный подход используется в правительственных организациях и крупных компаниях.

Процесс проектирования, в основе которого лежит **система управления качеством**, включает в себя большое количество разноплановых мер для отслеживания наиболее значимых для функционирования системы параметров. Выбранные параметры наблюдаются на всех стадиях разработки системы, причём, в некоторых случаях, в ущерб другим. В некоторых случаях набор такого рода параметров может быть составлен не правильно, и оптимизация системы будет проведена ошибочно. В этом заключается основной недостаток подобного подхода. Кроме этого, при появлении новых требований весьма трудно изменять функциональность, а значит дальнейшее развитие системы, в том числе и из-за архитектурных просчётов, может быть невозможно. Такой подход считается консервативным, а его применение целесообразно при необходимости создать систему с экстремальными характеристиками.

**Архитектурный подход** к проектированию информационных систем можно считать наиболее зрелым. Его ключевым аспектом является создание фреймворка, то есть каркаса, адаптация которого под нужды конкретной системы будет легко осуществима. В соответствии с этим, задача проектирования разбивается на две: разработка многократно используемого каркаса и создание

системы на его основе. Следует отметить, что данные подзадачи могут решаться различными группами специалистов. При использовании каркасов появляется возможность довольно быстро изменять функциональность системы за счёт итеративности процесса проектирования. Архитектурный подход призван ликвидировать недостатки, возникающие в процессе проектирования, основанном на управлении требованиями.

Из всех рассмотренных стилей проектирования не возможно однозначно определить лучший, поскольку каждая конкретная проектная группа, каждая проектируемая система имеет свои особенности, на основании которых следует выбрать тот подход, применение которого позволит решить поставленные задачи в установленные сроки и в рамках выделенного бюджета.

## 3.5. Значение программного обеспечения в информационных системах. Характеристики качества программного обеспечения

Пользователи современных информационных систем практически всегда взаимодействуют с ними при помощи специальных программных модулей, от показателей качества которых зависит уровень качества всей информационной системы целиком.

Существует огромное количество стандартов для создания правильной и надёжной архитектуры, а также для разработки и интеграции программных систем. Практически каждая технология, существующая в мире, имеет свой

стандарт, описанный в соответствующих нормативных документах. Применение этих стандартов существенно увеличит шансы на успешное создание системы и её дальнейшее безотказное функционирование, однако рациональность их применения должна определяться до момента начала работ, поскольку сложность системы при их интеграции может существенно возрасти.

Качеством программного обеспечения можно считать совокупность его характеристик, относящихся к возможности удовлетворять высказанные или подразумеваемые потребности всех заинтересованных лиц. Данное определение включено в стандарт ISO 9126, в котором также определены и сами характеристики.

Можно выделить три аспекта качества:

1. Внутреннее качество (характеристики самого программного обеспечения).
2. Внешнее качество (поведенческие характеристики программного обеспечения).
3. Контекстное качество (ощущения пользователей при различных контекстах использования).

Руководствуясь этими аспектами, стандарт ISO 9126 выделяет шесть характеристик качества программного обеспечения:

1. Функциональность.
2. Надёжность.
3. Производительность.
4. Удобство использования.
5. Удобство сопровождения.
6. Переносимость.

**Функциональность** подразумевает способность ПО решать задачи в определённых условиях и подразделяется на следующие подхарактеристики:

- функциональная пригодность (suitability) – способность решать нужный набор задач;
- точность (accuracy) – способность получать требуемые результаты;
- способность к взаимодействию (interoperability) – способность взаимодействия с требуемым набором иных систем;
- защищённость (security) – способность предотвращать неавторизованный доступ к данным и программам;
- соответствие стандартам и правилам (compliance) – соответствие программного обеспечения различным регламентирующим нормам.

**Надёжность** (reliability) характеризуется способностью программного обеспечения удерживать функциональность в заданных рамках при определённых условиях и подразделяется на следующие подхарактеристики:

- зрелость (maturity) – величина, обратная частоте отказов программного обеспечения;
- устойчивость к отказам (fault tolerance) – способность удерживать определённый уровень работоспособности при различных отказах и нарушениях правил взаимодействия с окружением;
- способность к восстановлению (recoverability) – способность восстанавливать требуемый уровень работоспособности после отказа;

- соответствие стандартам надёжности (reliability compliance).

**Производительность** (efficiency) определяется способностью программного обеспечения при определённых условиях гарантировать требуемую работоспособность по отношению к выделяемым для этого ресурсам. Можно также определить, как отношение получаемых результатов к затраченным ресурсам. Данная характеристика подразделяется на следующие подхарактеристики:

- временная эффективность (time behavior) – способность программного обеспечения получать требуемые результаты и обеспечивать передачу необходимого объёма данных за определённое время;
- эффективность использования ресурсов (resource utilization) – способность программного обеспечение решать требуемые задачи и использованием заданных объёмов определённых видов ресурсов;
- соответствие стандартам производительности (efficiency compliance).

**Удобство использования** (usability) характеризуется привлекательностью для пользователей, удобством в обучении и использовании программного обеспечения. В своём составе также имеет ряд подхарактеристик:

- понятность (understandability) – величина обратная усилиям, затраченным пользователями, по осознанию применимости программного обеспечения для решения требуемых задач;
- удобство работы (operability) – величина обратная усилиям, затраченным пользователями, для

решения требуемых задач при помощи программного обеспечения;

- удобство обучения (learnability) – величина обратная усилиям, затраченным пользователями, на процесс обучения работе с программным обеспечением;
- привлекательность (attractiveness) – способность программного обеспечения быть привлекательным для пользователей;
- соответствие стандартам удобства использования (usability compliance).

**Удобство сопровождения** (maintainability) характеризуется удобством сопровождения программного обеспечения. Данная характеристика также включает ряд подхарактеристик:

- анализируемость (analyzability) характеризуется удобством проведения анализа ошибок, дефектов, недостатков, необходимостей внесения изменений и их возможных последствий;
- удобство внесения изменений (changeability) – величина обратная трудозатратам на выполнение требуемых изменений;
- стабильность (stability) – величина обратная риску появления непредусмотренных последствий при внесении требуемых изменений;
- удобство проверки (testability) – величина обратная требуемым трудозатратам на тестирование и другие виды проверок достижения предусмотренных результатов при внесении изменений;
- соответствие стандартам удобства сопровождения (maintainability compliance).

**Переносимость** (portability) характеризуется способностью программного обеспечения сохранять работоспособность при изменении организационных, аппаратных и программных аспектов окружения. Для этой характеристики выделяются следующий подхарактеристики:

- адаптируемость (adaptability) – способность программного обеспечения без совершения непредусмотренных действий приспосабливаться к изменениям окружения;
- удобство установки (installability) – способность программного обеспечения устанавливаться в заранее определённое окружение;
- способность к сосуществованию (coexistence) – способность программного обеспечения функционировать в общем окружении с другими программами, разделяя с ними ресурсы;
- удобство замены (replaceability) – возможность применения программного обеспечения вместо уже используемого для решения тех же задач, в том же окружении;
- соответствие стандартам переносимости (portability compliance).

Все указанные характеристики описывают внутреннее и внешнее качество программного обеспечения. Для описания контекстного качества существует другой, уменьшенный набор характеристик:

- эффективность (effectiveness) – способность программного обеспечения решать пользовательские задачи с заданной точностью и в заданном контексте;

• продуктивность(productivity)–способность программного обеспечения получать требуемые результаты при использовании заранее определённого количества ресурсов;

• безопасность (safety) – способность программного обеспечения поддерживать требуемый низкий уровень риска нанесения ущерба людям, бизнесу и окружающей среде;

• удовлетворённость пользователей (satisfaction) – способность программного обеспечения при использовании в определённом контексте приносить удовлетворение пользователям.

Руководствуясь рассмотренными показателями можно значительным образом увеличить качество программных модулей, а, следовательно, и всей информационной системы в целом.

## 3.6. Функциональные компоненты информационной системы

Учитывая принцип декомпозиции, принято проектировать информационные системы с разделением функционального назначения их компонентов, то есть создавать многоуровневое представление.

Можно выделить три основные функциональные группы, предназначенные для решения различных по смыслу задач:

1. Взаимодействие с пользователями.
2. Бизнес-логика.
3. Управление ресурсами.

Реализация такого функционала происходит при помощи создания соответствующей программной системы. Такая система также имеет многоуровневое представление компонентов (рис. 9).

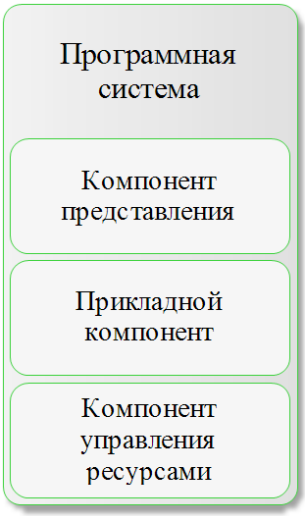



Рисунок 9 - Компоненты программной системы

**Компонент представления** служит для обеспечения взаимодействия пользователей с программой, то есть обрабатывает нажатия клавиш, движения различных контроллеров, осуществляет вывод информации – предоставляет пользовательский интерфейс.

**Прикладной компонент** представляет собой набор правил и алгоритмов реализации функций системы, реакций на действия пользователей или внутренних события, обработки данных.

**Компонент управления** ресурсами отвечает за хранение, модификацию, выборку и удаление данных, связанных с решаемой прикладной задачей.

Одним из важнейших этапов проектирования архитектуры информационной системы является распределение этих функциональных компонентов по выбранной платформенной архитектуре.

## 3.7. Платформенные архитектуры информационных систем

Можно выделить три направления развития платформенных архитектур:

1. Автономные.
2. Централизованные.
3. Распределённые.

**Автономная архитектура** подразумевает наличие всех функциональных компонентов системы на одном физическом устройстве, например, компьютере и не должна иметь связей с внешней средой. Примером таких систем могут служить системные утилиты, текстовые редакторы и достаточно простые корпоративные программы. Следует отметить, что в процессе построения корпоративной информационной системы, как правило, не должно формироваться не связанных узлов или модулей. Их появление может быть обусловлено определёнными требованиями к безопасности или надёжности.

**Централизованная архитектура** подразумевает выполнение всех требуемых задач на специально отведённом узле, мощности которого достаточно, чтобы удовлетворить потребности всех пользователей. Такой тип архитектуры был популярен на заре компьютерной техники

(70-е годы 20 века), однако, и в настоящий момент является востребованным. Компоненты системы в данном случае распределяются между вычислительным узлом, который называется мейнфрейм (mainframe), и терминальной станцией, за которой работает пользователь. Терминал содержит компонент представления, а мейнфрейм – прикладной компонент и компонент управления ресурсами. Следует отметить, что терминал выступает исключительно в виде устройства ввода-вывода и не имеет иных функциональных возможностей. Представление централизованной архитектуры показано на рис.10.

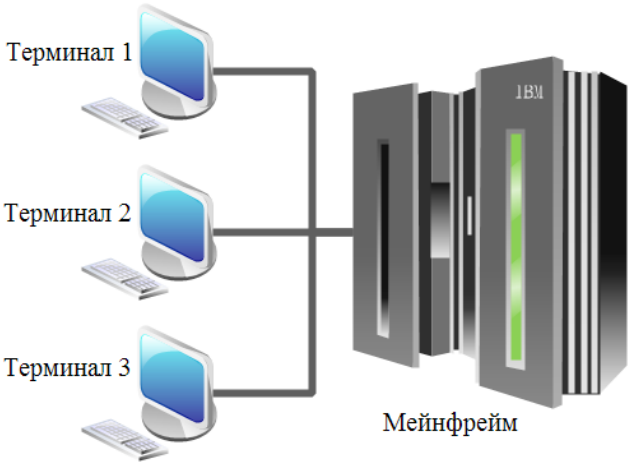



Рисунок 10 - Представление
централизованной архитектуры

К достоинствам такой архитектуры можно отнести:

- отсутствие необходимости администрирования рабочих мест;

- лёгкость обслуживания и эксплуатации системы, поскольку все ресурсы сосредоточены в одном месте.

Недостатками подобной архитектуры являются:

- функционирование всей системы полностью зависит от главного узла (мейнфрейма);
- все ресурсы и программные средства являются коллективными и не могут быть изменены под нужны конкретных пользователей.

Чтобы избавиться от последнего недостатка, в современных информационных системах применяются технологии виртуализации, благодаря которым становится возможным выделить каждому пользователю необходимое количество ресурсов и установить требуемое программное обеспечение.

Следует заметить, что при помощи технологий виртуализации можно создать практически любую архитектуру, используя при этом только ресурсы мейнфрейма.

Появление и развитие **распределённых архитектур** связано с интенсивным развитием технических и программных средств. В данном типе архитектуры функциональные компоненты информационной системы распределяются по имеющимся узлам в зависимости от поставленных целей и задач. Можно выделить шесть основных характеристик архитектуры распределённых систем:

- совместное использование ресурсов (как аппаратных, так и программных);

- открытость – возможность увеличения типов и количества ресурсов;
- параллельность – возможность выполнения нескольких процессов на различных узлах системы (при этом они могут взаимодействовать);
- масштабируемость – возможность добавлять новые свойства и методы;
- отказоустойчивость – способность системы поддерживать частичную функциональность за счёт возможности дублирования информации, аппаратной и программной составляющей.

К недостаткам распределённых систем следует отнести:

- структурная сложность;
- сложно обеспечить достаточный уровень безопасности;
- на управление системой требуется большое количество усилий;
- непредсказуемая реакция на изменения.

Все из них связаны в первую очередь со сложной структурой, разноплановым оборудованием и сложной системой распределения прав доступа. Необходимо учитывать все из них, иначе разработанная информационная система не сможет функционировать в рамках ожидаемых параметров.

Существуют следующие виды распределённых архитектур:

- архитектура «файл-сервер»;
- архитектура «клиент-сервер»;
- архитектура Web-приложений.

**Файл-серверная архитектура** подразумевает наличие выделенного сетевого ресурса для хранения данных. Такой ресурс называется «файловым сервером». При такой архитектуре все функциональные компоненты системы расположены на пользовательском компьютере, который называется «клиентом», а сами данные находятся на сервере.

Такая организация системы имеет следующие достоинства:

- многопользовательский режим работы с данными, хранящимися на сервере;
- централизованное управление правами доступа к общим данным;
- малая стоимость разработки;
- высокая скорость разработки.

Недостатки файл-серверной архитектуры:

- последовательный доступ к общим данным и отсутствие гарантии их целостности;
- производительность (зависит от производительности сети, клиента и сервера);

Классическое представление файл-серверной архитектуры представлено на рис.11.

**Архитектура «клиент-сервер»** представляет собой сетевую инфраструктуру, в которой серверы являются поставщиками определённых сервисов (услуг), а клиентские компьютеры выступают их потребителями.

Классическое представление клиент-серверной архитектуры подразумевает наличие в сети сервера и нескольких подключённых к нему клиентов. В таких системах сервер, в основном, играет роль поставщика услуг по использованию базы данных.

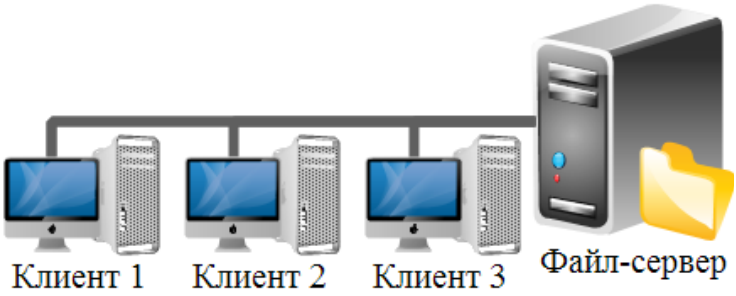



Рисунок 11 - Представление файл-серверной архитектуры

Эта архитектурная модель называется двухзвенной или двухуровневой (two-tier architecture). Двухзвенная архитектура представлена на рис.12.

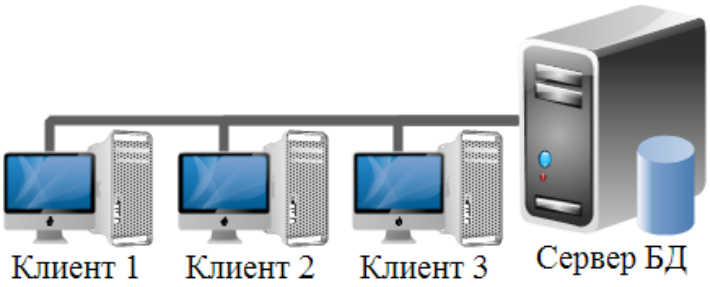



Рисунок 12 - Двухзвенная клиент-серверная архитектура

Преимущества данной архитектуры:

- поддержка многопользовательской работы;
- гарантия целостности данных;
- наличие механизмов управление правами доступа к ресурсам сервера;

- возможность распределения функций между узлами сети.

Недостатки:

- выход из строя сервера может повлечь неработоспособность всей системы;
- требуется высокий уровень технического персонала;
- высокая стоимость оборудования.

При увеличении масштабов системы может потребоваться замена аппаратной части сервера и клиентских машин. Однако, при увеличении числа пользователей возникает необходимость синхронизации версий большого количества приложений. Для решения этой проблемы используют многозвенные архитектуры (три и более уровней). Часть общих приложений переносится на специально выделенный сервер, тем самым снижаются требования к производительности клиентских машин. Клиенты с низкой вычислительной мощностью называют «тонкими клиентами», а с высокой производительностью – «толстыми клиентами». При многозвенной архитектуре с выделенным сервером приложений существует возможность использования портативных устройств. Многозвенная архитектура показана на рис.13.

Использование такого типа архитектуры обусловлено высокими требованиями приложения к ресурсам. В таком случае существует вынести его на отдельный сервер и, тем самым, понизить требования к производительности рабочих станций.

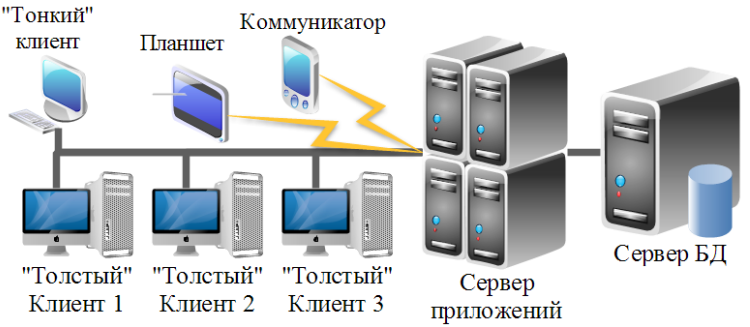



Рисунок 13 - Многозвенная клиент-серверная архитектура

Грамотный подбор характеристик сервера приложений, сервера баз данных и клиентских рабочих станций позволит создать информационную систему с приемлемой стоимостью владения.

Следует заметить, что распределение функциональных компонентов системы при использовании клиент-серверной архитектуры может производиться несколькими способами (рис.14).

**Архитектура Web-приложений** или архитектура Web-сервисов подразумевает предоставление некоторого сервиса, доступного в сети Internet, через специальное приложение.

Основой для предоставления таких услуг служат открытые стандарты и протоколы SOAP, UDDI и WSDL. SOAP (Simple Object Access Protocol) определяет формат запросов к Web-сервисам. Данные между клиентом и сервисом передаются в SOAP-конвертах (envelops). WSDL (Web Service Description Language) служит для описания интерфейса предоставляемого сервиса. Перед развёртыванием web-приложения требуется составить

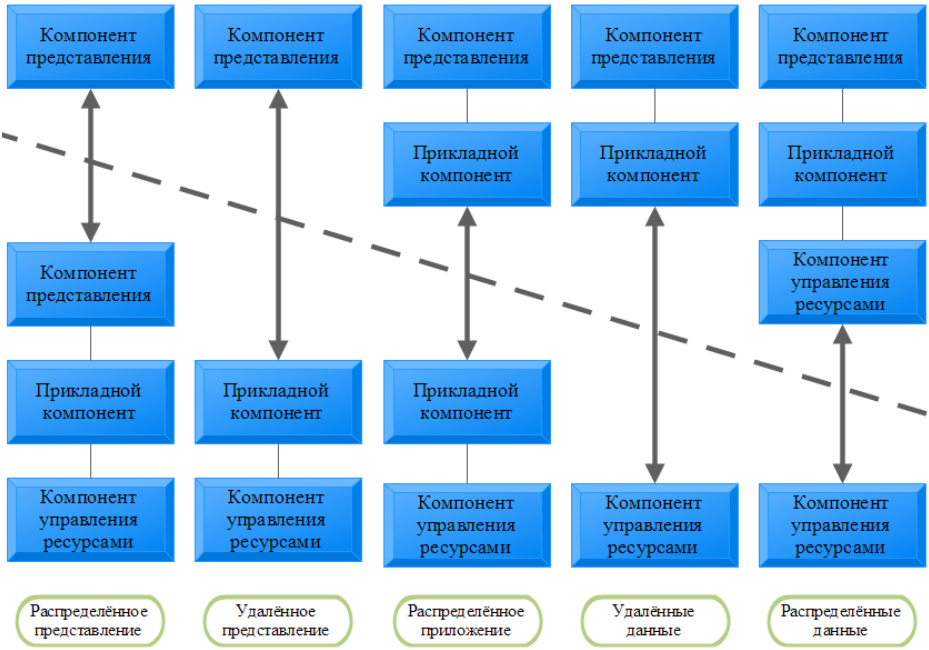



Рисунок 14 - Примеры распределения функциональных компонентов системы

его описание, указать адрес, список поддерживаемых протоколов, перечень допустимых операций, а так же форматы запросов и ответов. UDDI (Universal Description, Discovery and Integration) представляет собой протокол поиска Web-сервисов в сети Internet. Поиск осуществляется по их описаниям, которые расположены в специальном реестре.

Архитектура таких сервисов схожа по концепции с многозвенной клиент-серверной, однако, сервера приложений и баз данных располагаются в сети Internet.

Можно выделить три технологии, которые возможно использовать для построения распределённой архитектуры Web-сервиса:

1. EJB (Enterprise JavaBeans).
2. DCOM (Distributed Component Object Model).
3. CORBA (The Common Object Request Broker Architecture).

Идеей для создания **EJB** было желание создать такую инфраструктуру, в которой было бы легко добавлять и удалять компоненты, при этом изменяя функциональность сервера. EJB позволяет разработчикам составлять собственные приложения из заранее созданных модулей. При этом возможно их изменение, что делает процесс разработки гибким и гораздо более быстрым. Данная технология совместима с CORBA и Java API.

Взаимодействие между клиентов и сервером в данном случае представляется как взаимодействие EJB-объекта, генерируемого специальным генератором, и EJB-компонента, написанного разработчиком. При необходимости вызвать метод у EJB-компонента, находящегося на сервере, вызывается одноимённый метод EJB-объекта, расположенного на стороне клиента, который связывается с требуемой компонентой и вызывает необходимый метод.

**Достоинства EJB:**

- простое и быстрое создание;
- Java-оптимизация;
- кроссплатформенность;
- встроенная безопасность.

**Недостатки EJB:**

- сложность интегрирования с приложениями;
- плохая масштабируемость;
- низкая производительность;
- отсутствие международной стандартизации.

**DCOM** представляет собой распределённую программную архитектуру от компании Microsoft. С её помощью программный компонент одного компьютера может передавать сообщения программному компоненту другого компьютера, причём соединение устанавливается автоматически. Для надёжной работы требуется обеспечить защищённое соединение между связанными компонентами, а также создать систему перенаправления трафика.

**Достоинства DCOM:**

- независимость от языка;
- динамическое нахождение объектов;
- масштабируемость;
- открытый стандарт;

**Недостатки DCOM:**

- сложность реализации;
- зависимость от платформы;
- поиск через службу Active Directory;
- отсутствие именования сервисов через URL.

Технология **CORBA** рассматривает все приложения в распределённой системе как набор объектов. Объекты могут одновременно выступать

в роли клиента и сервера, вызывая методы других объектов и отвечая на их вызовы. Применение данной технологии позволяет строить системы, превосходящие по сложности и гибкости системы с архитектурой клиент-сервер (как двухуровневой, так и трёхуровневой).

**Достоинства CORBA:**

- независимость от платформы;
- независимость от языка;
- динамические вызовы;
- динамическое обнаружение объектов;
- масштабируемость;
- индустриальная поддержка.

**Недостатки:**

- отсутствие именования по URL;
- практически полное отсутствие реализации CORBA-сервисов;

При грамотном подходе к построению архитектуры информационной системы может потребоваться использование сразу нескольких из рассмотренных технологий. Каждая из них будет реализовывать определённый функционал, который может потребовать также нескольких типов платформенных архитектур. В одной системе могут работать несколько файл-серверов, несколько серверов приложений и несколько серверов баз данных. Таким образом можно распределять нагрузку в системе или группировать наборы сервисов по выполняемым функциям.

## 3.8 Понятие и классификация архитектурных стилей

Большая часть процессов по проектированию информационных систем подразумевает использование опыта реализации похожих проектов. Сложно представить систему, для реализации которой нельзя было бы применить уже готовые решения или опыт, полученный при их создании. Архитектурный стиль можно охарактеризовать как сходство в подходах к реализации поставленных задач, обусловленное опытом. Он определяет перечень компонентов системы, способы и условия их взаимодействия. К сожалению, вопреки множеству попыток, не существует стандартных языков описания архитектур.

Архитектурные стили подразделяются на пять групп (рис.15.):

1. Потоки данных (Data Flow Systems).
2. Вызов с возвратом (Call-and-Return Systems).
3. Независимые компоненты (Independent Component Systems).
4. Централизованные данные (Data-Centric Systems).
5. Виртуальные машины (Virtual machines).

Системы потоков данных, в свою очередь, подразделяются на:

- системы пакетно-последовательной обработки (Batch Sequential Systems);
- системы типа конвейеры и фильтры (Pipe and Filter Architecture).

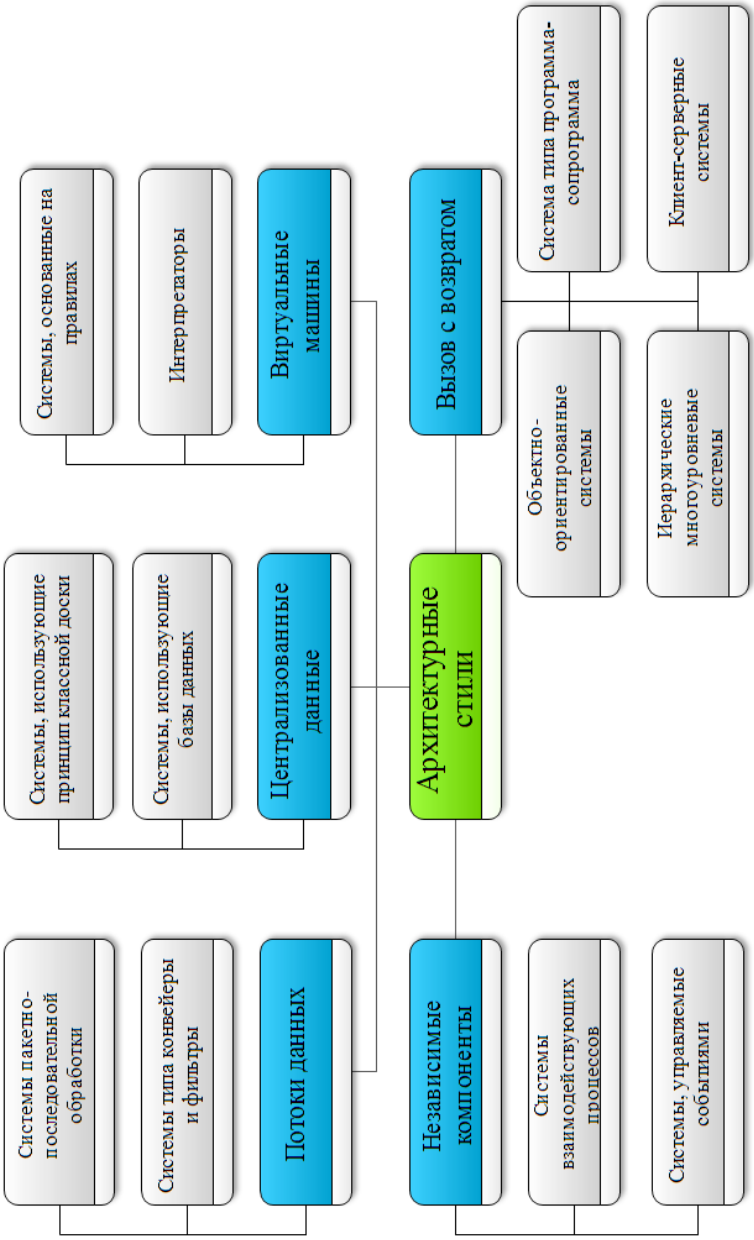



Рисунок 15 - Классификация архитектурных стилей

В системах пакетно-последовательной обработки решаемая задача делится на совокупность подзадач, механизм решения которых будет реализован в отдельных программных модулях, объединённых в линейную структуру. Выходные данный одной подзадачи являются входными данными для другой.

Стиль «конвейеры и фильтры» может считаться обобщением пакетно-последовательной обработки. Его структура состоит из множества модулей, каждый из которых выполняет один или несколько процессов. Результаты выполнения одного процесса могут передаваться как одному, так и нескольким модулям, причём различными способами. Такие системы реализуют принцип конвейера, в котором могут присутствовать обратные связи.

Хорошим примером такого подхода может служить компилятор, который последовательно выполняет лексический анализ, семантический анализ, оптимизацию и генерацию кода.

Системы, функционирующие при помощи **вызовов с возвратами** являются синхронными программными архитектурами, клиентская часть которых приостанавливает функционирование на время обслуживания собственного запроса сервером. Такие архитектуры могут включать произвольное количество уровней вложенности. Существует несколько типов подобного рода систем:

- программа-сопрограмма (Main Programm and Subroutines);
- объектно-ориентированные системы (Object-Oriented Systems);

- клиент-серверные системы (Client-Server Systems);
- иерархические многоуровневые системы (Hierarchically Layered Systems).

Стиль «программа-сопрограмма» является реализацией идей структурного программирования и подразумевает наличие главной управляющей программы (контроллера), отвечающей за процесс функционирования, и множества сопрограмм, реализующих функциональность. Разновидностью данного подхода считается архитектура «ведущий-ведомый» (Master-Slave Architecture), в которой основная программа и сопрограммы работают одновременно (параллельно). Контроллер выполняет функции диспетчера процесса, в то время, как сопрограммы выполняют задания, по завершении которых запрашивают у него новые.

Объектно-ориентированные системы являются частным случаем систем «программа-сопрограмма». Общение между объектами, включающими (инкапсулирующими) в себя код и данные, осуществляется либо с помощью вызовов процедур, либо при помощи сообщений. Следует заметить, что вызывающий объект должен знать, где находится вызываемый, кроме того, ему необходимо знать набор интерфейсов, которые он может использовать. В результате инкапсуляции скрываются данные о реализации какого-либо объекта, в результате чего становится возможным вносить в него изменения без уведомления об этом конечных пользователей, что является неоспоримым преимуществом данного типа архитектуры. Также к

достоинства можно отнести естественную поддержку распараллеливания процессов.

Клиент-серверные системы также можно считать частным случаем стиля «программа-сопрограмма», с той лишь разницей, что контроллер и сопрограммы могут располагаться на различных узлах сети.

Для крупномасштабных систем применяют иерархически многоуровневый стиль, в котором каждый из имеющихся слоёв можно рассматривать как набор сервисов для вышележащего слоя. Соответственно, вышележащий слой является клиентом, а нижележащий — сервером. Такой стиль оправданно применяется для создания стеков протоколов или операционных систем. Главным его достоинством является возможность ведения разработки каждого из слоёв независимо. Стоит заметить, что не все алгоритмы можно реализовать виде многоуровневой структуры, поэтому её применение не всегда оправдано.

Системы, функционирующие по принципу **независимых компонентов**, используют механизм неявного вызова операторов, то есть взаимодействующие операторы могут работать независимо и располагаться на разных хостах сети. Выделяют два типа подобных систем:

- системы взаимодействующих процессов (Communicating Sequential Processes);
- системы, управляемые событиями (Event-Based Systems).

Основным принципом функционирования систем взаимодействия процессов является обмен сообщениями между независимыми процессами.

В системах, управляемых событиями, процессы запускаются только в момент появления определённого события, однако получатель сообщения о событии может не знать об отправителе, а отправитель – о получателе. Похожие принципы функционирования у систем с прерываниями.

При наличии в системе общедоступного централизованного хранилища информации, её относят к стилю **централизованных данных** (репозитория). При использовании данного подхода данные вводятся в системы однократно и при необходимости дополняются. Это обеспечивает общий доступ нескольких приложений к данным, возможность обмена данными, исключает дублирование и упрощает масштабирование. Существует две разновидности подобных систем:

- системы, основанные на использовании централизованной базы данных (Database Systems) – чаще всего подразумевают наличие реляционной базы данных;
- системы, использующие принцип классной доски (Blackboard Systems).

Системы, построенные по принципу классной доски, характеризуются наличием общей разделяемой памяти (базы данных), хранящей результаты выполняемых процессами действий. В таких системах существует возможность оповещения заинтересованных процессов об изменениях в требуемой им информации.

**Виртуальные машины** представляют собой специальные эмуляторы, обеспечивающие программный интерфейс отличный от используемого. Виртуальные машины могут напрямую работать с аппаратной платформой или являться надстройками

операционной системы. При рассмотрении информационной системы в виде многослойной структуры, виртуальная машина будет являться верхним слоем, обеспечивающий взаимодействие с пользовательскими приложениями.

- интерпретаторы (Interpreters);
- системы, основанные на правилах (Rule-Based Systems).

Интерпретаторы предназначены для обеспечения работоспособности различного рода программ, изначально созданных для различных платформ. Например, запуск и отладка Linux-приложений в среде Windows.

Системы, основанные на правилах, представляют все данные и логику в виде набора специализированных правил. В таких системах для каждой задачи существует набор фактов и набор правил. Для решения задачи к фактам применяются правила до тех пор, пока не будет получен результат. Примером таких систем может являться CLIPS.

Целесообразность использования различных архитектурных стилей приведена в таблице 2.

Таблица 2.

**Целесообразность применения стилей**

| **Название стиля** | **Целесообразность применения** |
|---|---|
| Пакетно-последовательный | Решаемую задачу можно разделить на подзадачи, использующие единственную операцию ввода-вывода. |

| | |
|---|---|
| Конвейеры и фильтры | Процесс решения задачи можно представить в виде последовательности повторяющихся операций над независимыми однотипными данными. |
| Программа-сопрограмма | Фиксированный порядок операций, простаивание из-за ожидания ответов от компонентов. |
| Объектно-ориентированный | Возможность использования механизмов наследования, расположение объектов на разных хостах. |
| Клиент-серверный | Возможность представить решаемую задачу в виде набора запросов и ответов от клиентов к серверу. |
| Иерархический многоуровневый | Возможность представить задачу, как совокупность слоёв с определёнными интерфейсами, необходимость в различных вариантах реализации бизнес-логики, портируемость, использование имеющихся реализаций. |
| Взаимодействующие процессы | Механизм взаимодействия между процессами – обмен сообщениями, объём долговременных централизованных данных небольшой. |
| Управляемые события | Асинхронный процесс функционирования системы, возможно представление системы в виде независимых процессов. |

| | |
|---|---|
| Централизованная база данных | Доступна СУБД, задачи разделяются на производителей и потребителей данных, очерёдность исполнения компонентов определяется последовательностью входных запросов. |
| Классная доска | Большое количество клиентов общающихся между собой. |
| Интерепретатор | Необходимо нивелировать специфику платформ, предоставить специфическую среду работы. |
| Основанный на правилах | Решение задачи можно представить в виде набора правил и условий их применения. |

## 3.9. Фреймворки (каркасы)

Термин фреймворк можно определить как общепринятые архитектурно-структурные решения и подходы к проектированию. Можно сказать, что фреймворк представляет собой общее решение сложной задачи.

Классификация фреймворков показана на рис.16.

Инфраструктурные фреймворки (System Infrastructure Frameworks) упрощают процесс разработки инфраструктурных элементов, применяются внутри организации и не продаются.

Фреймворки уровня промежуточного программного обеспечения (Middleware Frameworks) применяются для встраивания приложений и компонентов.

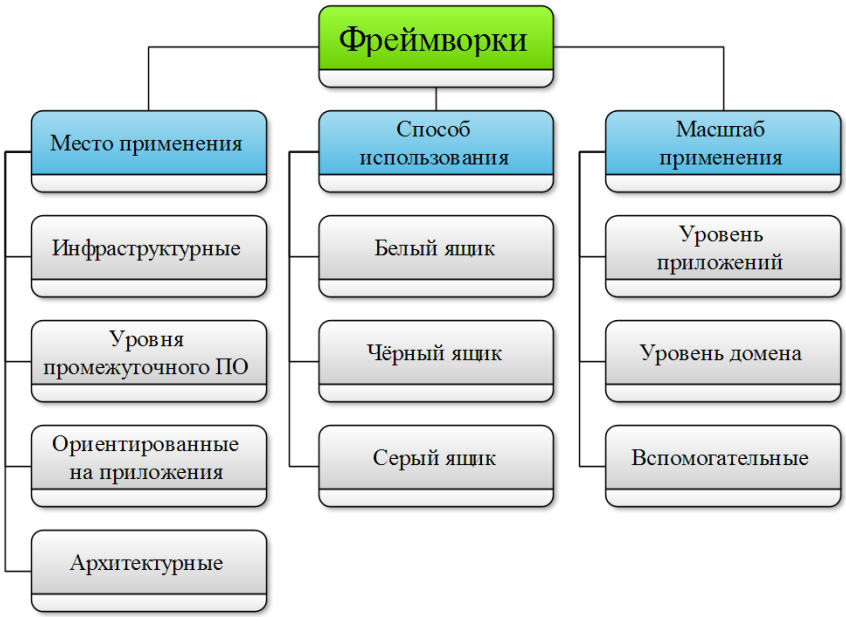



Рисунок 16 - Классификация фреймворков

Фреймворки, ориентированные на приложения, используются для поддержки систем, ориентированных на работу с конечными пользователями в конкретной предметной области.

В соответствии со стандартом ISO/IEC 42010 архитектурный фреймворк определяется как «совокупность соглашений, принципов и практик, используемых для описания архитектур и принятых применительно к некоторому предметному домену и (или) в сообществе специалистов (заинтересованных лиц)». Архитектурный фреймворк включает в себя описание заинтересованных лиц, типовые проблемы

предметной области, архитектурные точки зрения и методы их интеграции.

Фреймворки, используемые по принципу белого ящика (Architecture-driveng framework), применяют методы наследования и динамического связывания для формирования основных элементов приложения. Такие фреймворки определяются через интерфейсы объектов, добавляемых в систему. Для работы с ними необходима подробная информация о классах, расширение которых необходимо.

Фреймворки, функционирующие по принципу чёрного ящика, также называют фреймворками, управляемыми данными. Основными механизмами формирования приложений, в данном случае, выступают композиция и параметризация, при этом функциональность обеспечивается добавлением дополнительных компонентов. Следует отметить, что процесс использования фреймворков, работающих по принципу чёрного ящика проще, чем работающих по принципу белого ящика, однако их разработка сложнее.

На практике применяют подход серого ящика (grey box), являющийся комбинацией обоих подходов.

Фреймворки уровня приложений (application frameworks) предоставляют функционал по реализации типовых приложений (GUI, базы данных и т.д).

Фреймворки уровня домена (Domain Frameworks) применяются для создания приложений в определённой предметной области. Их классификация представлена на рис.17.

Мягкие фреймворки подразумевают возможность собственной настройки для решения конкретной задачи, в то время как жёсткие – нет.

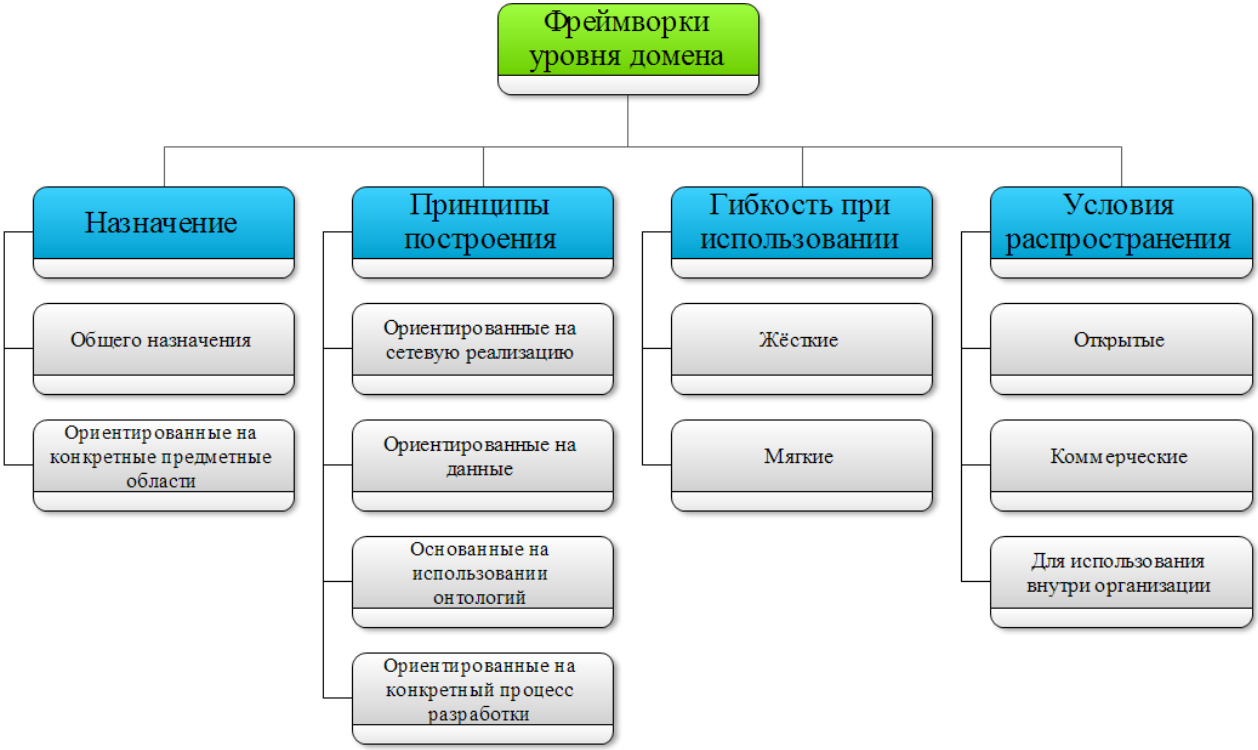



Рисунок 17 - Классификация фреймворков уровня домена

Вспомогательные фреймворки (Support Frameworks) применяются для решения частных задач.

В мире существует огромное количество различных фреймворков. В качестве примеров можно выделить пять наиболее известных:

1. Фреймворк Захмана.
2. TOGAF.
3. DoDAF.
4. FEA.
5. Gartner.

**Фреймворк Захмана** является одним из самых старых архитектурных фреймворков. Он был создан сотрудником компании IBM Джоном Захманом (John Zachman). Захман заложил в основу своего фреймворка классификацию (таксономию) артефактов системы. Среди них можно выделить данные, функциональность, модели, спецификации и документы. В результате, можно считать этот фреймворк онтологией верхнего уровня, описывающей конкретную систему (таблица3).

Для построения таксономии Захманом предложено ответить на шесть вопросов о функционировании организации: что, как, где, кто, почему. Данные вопросы относятся к следующим аспектам системы:

- используемые данные (что?);
- процессы и функции (как?);
- места выполнения процессов (где?);
- организации и персоналии (кто?);
- управляющие события (когда?);
- цели и ограничения, определяющие работу системы (почему?).

Таблица 3.

## Матрица Захмана

| | **Используемые данные (что?)** | **Процессы и функции (как?)** | **Места выполнения процессов (где?)** | **Организации и персоналии (кто?)** | **Управляющие события (когда?)** | **Цели и ограничения** | |
|---|---|---|---|---|---|---|---|
| **Контекст** | Список основных сущностей | Основные бизнес-процессы | Территориальное размещение организации | Важные внешние организации | Список событий | Бизнес-стратегия | Аналитики |
| **Бизнес-модель** | Отношения между сущностями | Подробное описание бизнес-процессов | Система логистики | Модель потоков данных | Базовый график работ | Дерево целей, Бизнес-план | Топ-менеджеры |
| **Системная модель** | Концептуальные модели данных | Архитектура приложений | Архитектура распределённой системы | Интерфейсы пользователя | Модель работы с событиями | Бизнес-правила | Архитекторы |
| **Технологическая модель** | Физическая модель данных | Программно-аппаратная архитектура | Технологическая архитектура | Архитектура представления | Алгоритмы обработки событий | Правила обработки событий | Разработчки |
| **Детальное описание** | Спецификации форматов данных | Исполняемый код | Архитектура сети | Роли и права пользователей | Обработка событий с помощью прерываний | Алгоритмы работы системы | Администраторы |
| **Функционирующая организация** | Данные | Реализуемая функциональность | Функционирующая сетевая инфраструктура | Организационная структура организации | История функционирования системы | Реализуемые стратегии | Пользователи |

Отвечать на эти вопросы необходимо с различной степенью детализации. Описано шесть уровней:

1. уровень контекста;
2. уровень бизнес-описаний;
3. системный уровень;
4. технологический уровень;
5. технический уровень;
6. уровень реальной системы.

Применительно к каждому уровню детализации существует своё заинтересованное лицо (stakeholders), то есть точка зрения:

1. аналитики (уровень контекста);
2. топ-менеджеры (уровень бизнес-описаний);
3. архитекторы (системный уровень);
4. разработчики (технологический уровень);
5. администраторы (технический уровень);
6. пользователи (уровень реальной системы).

По результатам проделанных действий сформировывается матрица размером 6 на 6, в каждой ячейке которой располагаются артефакты. Для заполнения ячеек введены следующие правила:

1. Колонки можно менять местами, но нельзя добавлять и удалять.
2. Каждой колонке соответствует собственная модель.
3. Каждая из моделей, соответствующих столбцов, должная быть уникальна.
4. Каждый уровень (строка) представляет собой описание системы с точки зрения группы пользователей (представляет отдельный вид).

5. Каждая из ячеек уникальна.

6. Каждая ячейка содержит описание аспекта реализации системы в виде модели и текстового документа.

7. Заполнение ячеек должно производиться последовательно сверху вниз.

Первая строка матрицы определяет контекст всех остальных и представляет собой общий взгляд на организацию. Вторая строка описывает функционирование организации в бизнес-терминах. Третья строка описывает бизнес-процессы в терминах информационных систем. Четвёртая строка позволяется распределить данные и выполняемые над ними операции по конкретным аппаратным и программным платформам. Пятая строка описывает конкретные модели оборудования, сетевые топологии и программный код. Шестая строка описывает готовую систему в виде руководств пользователей, справочных баз данных и т.д.

Столбец используемые данные, в соответствии с уровнями, содержит в своих ячейках следующие артефакты: 1 уровень – список основных сущностей; 2 уровень – семантическая модель; 3 уровень – нормализованная модель; 4 уровень – физическая модель данных или иерархия классов; 5 уровень – описание модели на языке управления данными; 6 уровень – фактические наборы данных, статистики и т.д.

Столбец процессы и функции описывает порядок действий для детализации процесса перехода от миссии организации к описанию конкретных операций: 1 уровень – перечисление ключевых бизнес-процессов; 2 уровень – описание бизнес-процессов; 3 уровень – описание операций над данными и архитектуры приложений; 4 уровень –

методы классов; 5 уровень – программный код; 6 уровень – исполняемые модули. С четвёртого уровня рассмотрение ведётся в рамках отдельных приложений.

Столбец места выполнения процессов определяет расположение компонент системы и сетевую инфраструктуру: 1 уровень – расположение основных объектов; 2 уровень – модель взаимодействия объектов; 3 – распределение компонентов информационной системы по узлам сети; 4 уровень – физическая реализация на аппаратных и программных платформах; 5 уровень – используемые протоколы и спецификации каналов связи; 6 – описание функционирования сети.

Столбец организации и персоналии определяет участников процесса функционирования: 1 уровень – список партнёров, подразделений организации, выполняемые функции; 2 уровень – полная организационная диаграмма (могут присутствовать требования к информационной безопасности); 3 уровень – участники бизнес-процессов и их роли; 4 уровень – требования к пользовательским интерфейсам; 5 уровень – правила доступа к объектам; 6 уровень – физическая реализация в коде.

Столбец управляющее событие определяет временные параметры системы и бизнес-процессов: 1 уровень – список значимых для системы событий; 2 уровень – базовый график работ; 3 уровень – модели работы с событиями; 4 уровень – алгоритмы обработки событий; 5 уровень – программная реализация; 6 уровень – история функционирования системы.

Столбец цели и ограничения указывает последовательность действий для перехода от задач бизне-

са к требованиям для элементов системы: 1 уровень – бизнес-стратегия; 2 уровень – дерево целей и бизнес-план; 3 уровень – правила и ограничения для бизнес-процессов; 4 уровень – приложения, включаемые в состав системы; 5 уровень – алгоритмы работы приложений; 6 уровень – физическая реализация в коде.

При помощи фреймворка Захмана нельзя описать динамику поведения системы (развитие), кроме того, в нём не существует механизма контроля за изменениями. Данный фреймворк распространяется на коммерческой основе.

Фреймворк TOGAF (The Open Group Architecture Framework) представляет собой набор средств для разработки архитектур различного назначения. С его помощью информационная система описывается как совокупность модулей.

В рамках TOGAF даётся особенное определение архитектуры – «формальное описание системы, или детальный план системы на уровне компонентов и методологии их реализации». Общепринятое же определение архитектуры (в соответствии со стандартом ANSI/IEEE 1471-2000) определяется как «описание организации системы в терминах компонентов, их взаимосвязей между собой и с окружающей средой и принципы управления их разработкой и развитием».

TOGAF состоит из четырёх архитектурных доменов:

- бизнес-архитектура (описывает ключевые бизнес-процессы, стратегию развития бизнеса и принципы управления);

- архитектура уровня приложений (описывает интерфейсы приложений и способы их применения в терминах бизнес-сервисов);
- архитектура уровня данных (определяет логическую и физическую структуру данных в организации);
- технологическая архитектура (определяет программную, аппаратную и сетевую инфраструктуры).

Главными составными частями TOGAF являются:

- ADM-методика (Architecture Development Method), описывающая процесс разработки архитектуры;
- руководства и методики проектирования для ADM;
- фреймворк архитектурного описания (Architecture Content Framework), являющийся детально проработанной моделью результатов разработки;
- архитектурный континуум организации (Enterprise Continuum), в виде репозитория архитектурных артефактов и реализаций;
- эталонные модели TOGAF (TOGAF Reference Models):
  - TRM (Technical Reference Model) – техническая эталонная модель;
  - III-RM (The Integrated Information Infrastructure Model) – интегрированная модель информационной инфраструктуры.

• фреймворк, описывающий структуру организации, её персонал, требуемые роли и уровни ответственности (Architecture Capability Framework).

В соответствии с методикой ADM архитектурный процесс можно разбить на девять фаз. ADM представляет из себя итерационный процесс происходящий на двух уровнях. На верхнем уровне каждой итерации повторяются общие для каждой из фаз действия. Нижний уровень описывает итерации внутри каждой фазы. Решения принимаются на основании существующих требований бизнеса и существующих решений.

Схема фаз TOGAF представлена на рис.18.

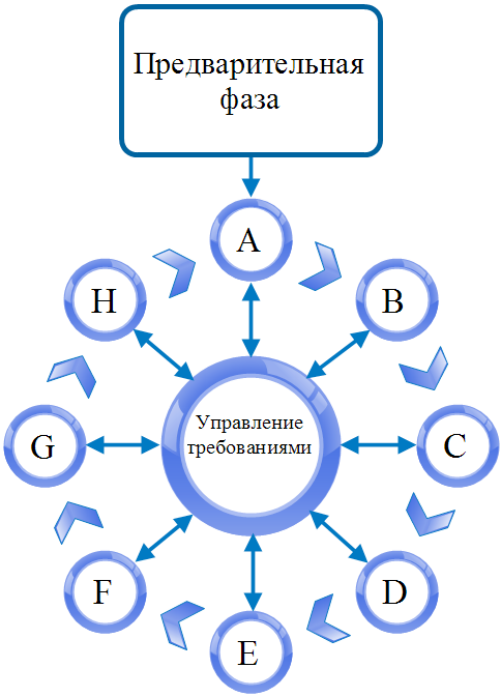



Рисунок 18 - Стадии TOGAF

Описание каждой конкретной фазы указано в таблице 4.

Таблица 4.

Описание фаз TOGAF

| **Стадия процесса** | **Описание** |
|---|---|
| Предварительная фаза (Preliminary Phase) | Определение способов управления, границ разработки, принципов реализации, уточнение модели относительно специфики. |
| Фаза A, разработка общего представления (Architecture Vision) | Создание общего представления об архитектуре, определение границ проекта, утверждение плана работ и заданий для исполнителей |
| Фаза B, разработка бизнес-архитектуры (Business Architecture) | Выявление принципов функционирования организации, принципов управления проектом |
| Фаза C, разработка информационной архитектуры (Information Systems Architecture) | Разработка архитектуры данных и приложений |
| Фаза D, разработка технологической архитектуры (Technology Architecture) | Подбор аппаратных средств, средств сетевой инфраструктуры, механизмов их взаимодействия |

| | |
|---|---|
| Фаза E, возможности и решения (Opportunities and Solutions) | Реализация разработанной архитектуры (купить готовое решение или сделать самим) |
| Фаза F, планирование перехода к новой архитектуре (Migration Planning) | Оценка рисков, анализ деталей перехода |
| Фаза G, формирование системы управления реализацией (Implementation Governance) | Мониторинг процесса внедрения и спецификация возникающих проблем |
| Фаза H, управление изменением архитектуры (Architecture Change Management) | Наладка процесса управления изменениями разработанной архитектуры |

Следует отметить, что TOGAF может использоваться не только как единственный фреймворк при разработке, но и в совокупности с другими фреймворками. В частности, в него входит специальный документ, определяющий соответствия между понятиями TOGAF и фреймворком Захмана.

Фреймворк миистерства обороны США **DoDAF** (Department of Defense Architecture Framework) состоит из трёх основных элементов: модели (models), виды (views) и точки зрения (viewpoints). Этот набор позволяется отражать взгляды всех заинтересованных

сторон. Несмотря на военное назначение, DoDAF является свободно распространяемым. На его базе были сформированы фремворки НАТО (NATO Architecture Framework – NAF), фреймворк Министерства Обороны Великобритании (Ministry of Defense Architecture Framework – MODAF) и множество других.

Главная особенность DoDAF – ориентация на данные, что подразумевает повышенную важность сохранности данных и повторного их использования. Основным классом систем, которые проектируются при помощи этого фреймворка, являются системы сбора, хранения и анализа данных для поддержки принятия решений (СППР).

DoDAF определяет модели, как шаблоны для сбора данных и подразделяет их на классы:

- таблицы;
- графические изображения структурных аспектов архитектурного решения;
- графические изображения поведенческих аспектов архитектурного решения;
- отображения, определяющие взаимосвязь между типами информации;
- онтологии;
- картинки в свободном формате;
- временные диаграммы.

Виды определяются способами представления для пользователя связанного набора данных. К видам можно отнести документы, таблицы, графики, диаграммы и т.д.

Точки зрения представляют собой упорядоченное множество видов.

Вторая версия DoDAF содержит восемь точек зрения:

- обобщенная (All Viewpoint): интегрирует все точки зрения для создания архитектурного контекста;
- определяющая потенциальные возможности (Capability Viewpoint): обучение персонала, сроки поставок и т.д.;
- определяющая данные и информацию (Data and Information Viewpoint): определяет способы представления и структуры данных;
- операционная (Operational Viewpoint): рассматривает сценарии работы и активности системы;
- проектная (Project Viewpoint): рассматривает требуемые характеристики и возможности системы;
- сервисная (Service Viewpoint): рассматривает совокупность сервисов;
- учитывающая стандарты (Standards Viewpoint): рассматривает технические стандарты, ограничения, методики, руководства и т.д.;
- системная (System Viewpoint): рассматривает совокупность взаимодействующих систем и их взаимодействие.

DoDAF использует мета-модель данных (Data Meta-Model – DM2), которая является онтологией, составленной из уровней, отражающих особенности представления информации для конкретных групп пользователей. DM2 может быть расширена. В ней определены три уровня:

1. Концептуальная модель данных (Conceptual Data Model) – описывает архитектуру в нетехнические терминах.

2. Логическая модель данных (Logical Data Model) – расширение концептуальной модели путём добавления атрибутов.

3. Спецификация обмена данными на физическом уровне (Physical Exchange Specification) – средство, обеспечивающее обмен информации между моделями.

Существует восемь базовых принципов, руководствуясь которыми, можно успешно применять DoDAF [Советов]:

1. Архитектурное описание должно быть чётко ориентировано на провозглашённые цели.

2. Архитектурное описание должно быть по возможности простым и понятным, но не упрощённым.

3. Архитектурное описание должно облегчать, а не затруднять процесс принятия решений.

4. Архитектурное описание должно быть составлено таким образом, чтобы его можно было использовать для сравнения различных архитектур.

5. При составлении архитектурного описания должны в максимальной степени использоваться стандартные типы данных, определяемые в DM2.

6. Архитектурное описание должно выполняться в терминах самих данных, а не инструментальных средств работы с данными.

7. Архитектурные данные должны быть организованы в виде, удобном для групповой работы.

8. Архитектурное описание должно быть построено таким образом, чтобы его можно было использовать в сетевой среде.

DoDAF предназначен для составления архитектурного описания системы. Результатом его применения будет являться набор документов, а не информационная система.

Архитектура федеральной организации (**Federal Enterprise Architecture** - FEA) призвана привести множество агентств к единой архитектуре. Фреймворк FEA можно считать наиболее полной методологией из всех рассматриваемых. Он включает и проработанную таксономию (модель Захмана), и архитектурный процесс (фреймворк TOGAF).

FEA включает набор из пяти эталонных моделей:

- модель бизнеса;
- модель обслуживания;
- модель компонентов;
- технологическая модель;
- модель данных.

Полное описание FEA состоит из:

- точки зрения, на архитектуры;
- эталонных моделей, описывающих различные точки зрения;
- процесса создания архитектуры;
- процесса перехода от старой архитектуры к новой;
- таксономия для классификации активов;
- методика оценки успешности применения.

В соответствии с FEA, архитектура состоит из сегментов, представляющих главные аспекты бизнеса. Сегменты бывают двух типов:

- базовый – аспект деятельности в границах политико-административного деления (здоровье является базовым сегментом для Министерства здравоохранения США);
- служебный – сегмент, являющийся фундаментальным, для большинства организаций (управление финансами является служебным сегментом для всех федеральных агентств).

Ещё одним типом активов являются службы. Служба — это четко определенная функция в границах политико-административного деления. К примеру, управление безопасностью — это служба, единообразно реализованная по всему предприятию.

Трудно различать службы предприятия и сегменты, особенно служебные. Отличия заключаются в области действия: служебные сегменты распространяется только на одну политическую организацию, а службы предприятия распространяется на все предприятие.

Глобальное определение сегментов, упрощает их повторное использование. Использование сегментов можно спланировать в границах политико-административного деления, а затем, при помощи этого плана, искать возможности для повторного использования разработанной архитектуры.

Пять эталонных моделей FEA созданы, чтобы упростить совместную работу и обмен данными в федеральном правительстве:

- эталонная модель бизнеса (BRM) дает бизнес-представление различных функций федерального правительства.
- эталонная модель компонентов (CRM) дает ИТ-представление систем, поддерживающих бизнес.
- техническая эталонная модель (TRM) определяет различные технологии и стандарты, используемые при построении ИТ-систем.
- эталонная модель данных (DRM) определяет стандартные способы описания данных.
- эталонная модель производительности (PRM) определяет стандартные способы описания полезности, обеспечиваемой архитектурами предприятий.

Процесс проектирования в FEA сфокусирован на создание архитектуры сегмента для общей архитектуры. Процесс разработки архитектуры сегмента состоит из четырёх этапов:

1. Анализ архитектуры: формирование представления сегмента, учитывая план организации.
2. Архитектурное определение: указание требуемого состояния сегмента, документирование показателей производительности, рассмотрение альтернатив и разработка архитектуры предприятия для сегмента (бизнеса, данных, служб, технологической).
3. Стратегия инвестиций и финансирования: анализ способов финансирования проекта.
4. План управления программой и реализация проектов: создание плана управления проектом, его реализации (контрольные точки, показатели производительности).

Уровень соответствия федеральных агентств критериям FEA оценивается по трем категориям:

- завершенность архитектуры — уровень готовности архитектуры;
- использование архитектуры — эффективность использования архитектуры при принятии решений;
- результаты использования архитектуры — преимущества, достигнутые благодаря использованию архитектуры.

Каждому агентству присваивается рейтинг успеха на основе оценок по каждой категории и совокупному показателю:

- зелёный — агентство показало хорошие результаты по всем трём критериям;
- жёлтый — агентство показало хорошие результаты в области завершенности и либо в области использования, либо в области результатов;
- красный — агентство либо не завершило разработку архитектуры, либо неэффективно использует разработанную архитектуру.

Эта структура применима и за пределами государственного сектора экономики. Рейтинги по категориям можно эффективно адаптировать ко многим предприятиям для оценки степени готовности их архитектуры.

Методология **Gartner** представляет собой набор рекомендаций по построению архитектуры предприятия компании Gartner.

Согласно Gartner, архитектура представляет собой непрерывный процесс создания, сопровождения и использования инфраструктуры предприятия.

Архитектура предприятия призвана объединить три группы профессионалов: владельцев бизнеса, ИТ-специалистов и специалистов по внедрению технологий. Успех оценивается прагматичными методами, например, по доходности бизнеса, а не по количеству отмеченных элементов в матрице процесса.

Архитектура предприятия должна начинаться с постановки целей, которые организация желает достичь, а не с анализа текущего положения дел.

Gartner рекомендует начать работу с написания рассказа о стратегическом направлении развития организации и бизнес-факторах, на которые необходимо реагировать. Рассказ должен быть написан простым языком, без использования аббревиатур, специальной терминологии и технических рассуждений. Рассказ должен быть всем понятен и направлен на формирование у всех единого представления.

После того как в организации будет сформировано единое представление о будущем, можно будет рассмотреть влияние этого представления на архитектуру бизнеса, технологическую архитектуру, информационную архитектуру и архитектуру решений. Общее представление о будущем определяет изменения, которые необходимо внести во все перечисленные выше архитектуры, приоритеты этих изменений и привязку этих изменений к ценности бизнеса. Архитектура предприятия, согласно представлению Gartner, связана со стратегией, а не с технической реализацией.

По мнению Gartner, существуют два самых важных вопроса: «куда организация стремится?», «как

она туда попадет?». Любое действие, не связанное с этими вопросами, считается неуместным.

Процесс проектирования архитектуры в методологии Gartner можно представить в виде следующих этапов:

1. Изложить стратегическое представление о будущем компании на языке бизнеса без упоминания технологий.
2. Составить перечень необходимых к решению задач.
3. Выбрать для реализации задачу из сформированного перечня.
4. Составить список требований к решению выбранной задачи.
5. Сформировать рабочую команду.
6. Разработать целевую архитектуру бизнеса, для выбранной задачи.
7. Создать спецификации будущей системы.
8. Изучить существующую архитектуру и определить, какие изменения необходимо внести.
9. Разработают технологическую архитектуру для ИТ-систем, которые будут поддерживать новую архитектуру бизнеса.
10. Изучить существующие архитектуры на предмет возможности повторного использования имеющихся технологических активов.

Компания Gartner не занимается реализацией архитектур, поэтому в её методологии нет каких-либо рекомендаций на этот счёт. Её подход заключается в создании высокоуровневой архитектуры, ориентированной на бизнес; детали рассматриваются

только тогда, когда это необходимо. Роль методологии Gartner заключается не в создании архитектуры предприятия, а в создании процесса, позволяющего развивать архитектуру предприятия в соответствии с бизнес-стратегией.

## 3.10. Интеграция информационных систем

Термин интеграция имеет широкое значение. Под ним можно понимать объединение информационных систем, приложений, различных компаний или людей.

Интеграция подразделяется на внешнюю и внутреннюю. Внутренняя интеграция подразумевает объединение корпоративных приложений в одной организации (Enterprise Application Integration), а внешняя – интеграцию информационных систем организаций (Business-to-Business Application).

Существуют четыре основных типовых интеграционных подхода:

- интеграция на уровне данных;
- интеграция на уровне бизнес-функций и бизнес-объектов;
- интеграция на уровне бизнес-процессов;
- порталы.

Интеграция на уровне данных (Information-Oriented Integration) подразумевает наличие в системах баз данных, для работы с которыми необходимо разработать единый программный интерфейс.

К основным технологическим решениям данного подхода относятся:

- системы репликации данных;
- федеративные базы данных;

- использование API для доступа к EPR-системам.

Репликация является процессов синхронизации данных между различными источниками. Необходимость в этом возникает в момент изменения блока информации в распределённых системах хранения, чтобы гарантировать корректность и непротиворечивость данных используемых во всех модулях или приложениях информационной системы. Обычно функции репликации возлагают на промежуточное программное обеспечение.

Федеративные базы данных (Federated Database Systems) предоставляют единый интерфейс к распределённым данным. Это обеспечивает интеграцию множества автономных данных, которые могут быть физически расположены на различных устройствах в сети. Такие базы данных принято называть виртуальными.

Использование API для доступа к ERP-системам призвано упростить механизмы обмена информацией между пользовательскими приложениями и программным обеспечением, предназначенным для управления функционированием производственных информационных систем (ERP).

Интеграция на уровне бизнес-функций и бизнес объектов подразумевает реализацию совместно используемых служб (сервисов). Служба может являться набором функций, используемом в нескольких приложениях. Набор служб и будет являться бизнес-функциями.

При использовании сервисно-ориентированной архитектуры, бизнес-функции можно рассматривать

как бизнес-сервисы, а при компонентном подходе – бизнес-объектами (бизнес-компонентами).

Интеграция на уровне бизнес-процессов различается в зависимости от уровня интеграции. При внутренней интеграции взаимодействует большое количество сервисов, а при внешней интеграции, в основном, два. Сами бизнес-процессы функционируют над выделенными службами, для управления которыми существует специальный интерпретируемый язык.

Порталы можно считать графическими интерфейсами бизнес-процессов, поскольку они предназначены для персонализированного доступа к информации и консолидации данных из нескольких источников.

Главное назначение процесса интеграции – объединение функций приложений или модулей для предоставления новой функциональности.

При интеграции приложений можно выделить два основных типа задач:

- задачи интеграции корпоративных приложений;
- задачи интеграции приложений из различных информационных систем.

Для решения задач первого типа применяются системы EAI, которые иногда называются A2A (Application-to-Application Integration), а для решения задач второго типа применяются системы B2B (Business-to-Business Integration).

В некоторых ситуациях очень сложно определить разницу между интеграцией A2A и B2B, поскольку сложность некоторых решений внутри

информационных систем может превышать сложность решений для их совместного функционирования.

Существуют три альтернативных топологии интеграции:

1. Точка-точка (Point-to-Point);
2. Шлюз (hub-and-spoke);
3. Шина (Bus).

В топологии «точка-точка» все объекты имеют прямые связи друг с другом (рисунок 19, а). Следует отметить, что каждая связь может быть реализована каким угодно способом. Варианты реализации зависят от требований и характеристик взаимодействия между объектами. К недостаткам топологии можно отнести:

- недостаточная гибкость;
- сложность поддержки многочисленных соединений «точка-точка»;
- изменения одного объекта влияют на оставшиеся;
- логика маршрутизации часто программируется в коде объектов;
- отсутствие общей модели безопасности;
- использование различных API;
- низкая надёжность;
- сложность создания фреймворков;
- сложность поддержки асинхронного взаимодействия;

Для сокращения числа используемых интерфейсов следует использовать топологию с общим шлюзом (рисунок 19, б) или топологию с общей шиной (рисунок 19, в). Такие модели интеграции реализуются на уровне промежуточного программного обеспечения.

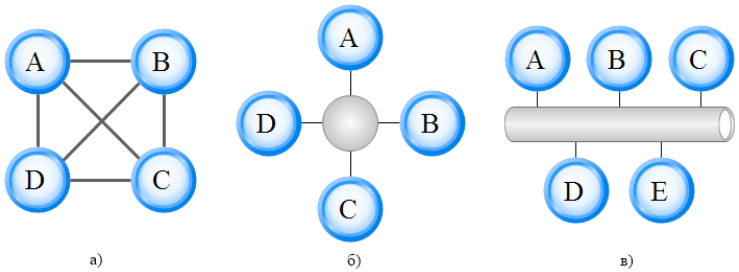



Рисунок 19 - Топологии интеграции

Следующим шагом в разработке интеграционных архитектур можно считать появление корпоративной сервисной шины (Enterprise Service Bus - ESB). Ряд авторов рассматривают системы ESB, как следующую ступень развития EAI. Однако есть несколько различаев:

1. EAI – централизованная архитектура, с обменом информации через хаб (брокер), а ESB – шинная архитектура, которая может быть реализована в виде нескольких распределённых систем;
2. В отличие от EAI, ESB ориентирована на использование открытых стандартов.

Эти два различия наглядно демонстрируют возможность использования ESB как интеграционной платформы позволяющей использовать различные механизмы интеграции. ESB позволяет проводить как внутреннюю, так и внешнюю интеграции, и представляет собой шину (backbone), работающую как слабосвязная система, управляемая событиями.

Концепции сервисно-ориентированных архитектур (СОА) и ESB очень сильно связаны. ESB подде-

рживает принцип реализации СОА: разделение представления службы и её реализации.

Функции ESB:

- предоставление интерфейсов взаимодействия;
- отправка и маршрутизация сообщений;
- преобразование данных;
- реакция на события;
- управление политиками;
- виртуализация.

На основании функций ESB, можно сформировать типовой список требований, предъявляемых со стороны пользователей:

- большая пропускная способность;
- поддержка нескольких стилей интеграции;
- обеспечение возможности приложениям работать с сервисами как напрямую, так и через адаптеры.

ESB является, по сути, логическим компонентом архитектуры, приводящим интеграционную инфраструктуру в соответствие принципа СОА. Архитектурами, построенными по принципу ESB, сложнее управлять, но они более гибкие и масштабируемые, более того, внедрение СОА не потребует изменений во всех элементах системы, в результате чего сможет происходить поэтапно.

Можно представить ESB в виде пятиуровневой структуры:

- уровень сопряжения (адаптеры и интерфейсы);
- транспортная подсистема;
- уровень реализации бизнес-логики;

- уровень управления бизнес-процессами;
- уровень бизнес-управления.

**Уровень сопряжения** призван решать проблему использования различных интерфейсов. На этом уровне функционируют адаптеры, отслеживающие события в приложениях и в интеграционной подсистеме, и обеспечивают преобразование передаваемых объектов при взаимодействии с транспортной подсистемой. Существует возможность, помимо заранее созданных адаптеров интеграционной платформы, использовать созданные самостоятельно.

**Адаптеры** можно разделить на две категории: технологические (применяются для интеграции технологических компонент, при отсутствии у них API), адаптеры для приложений (применяются для интеграции с конкретным приложением).

**Транспортная подсистема** предоставляет возможность асинхронного взаимодействия интегрируемым приложениям. Данный уровень также отвечает за управление и безопасность информации, может выполнять маршрутизацию сообщений и их обработку.

**Уровень реализации** бизнес-логики предоставляет функции для трансформации и маршрутизации сообщений. На этом уровне функционируют брокеры сообщений, обменивающиеся сообщениями через транспортную подсистему.

Брокер сообщений может выполнять следующие функции:

1. Принятие сообщений и их отправка по указанным адресам.

2. Преобразование форматов сообщений.
3. Агрегирование и фрагментации сообщений.
4. Взаимодействие с репозиториями.
5. Выборка данных через вызовы Web-служб.
6. Обработка ошибок и событий.
7. Маршрутизация сообщений по адресу, содержимому, теме.

Управления бизнес-процессами на одноимённом уровне осуществляется при помощи BPEL (Business Process Execution Language) посредством Web-сервисов.

Уровень бизнес-управления представляет из себя надстройку на предыдущим уровнем и предназначен для управления бизнес-процессами в терминах соответствующей предметной области.

Подход ESB имеет множество преимуществ и позволяет строить интеграционные архитектуры практически любой сложности. Типовая структура интеграционной системы представлена на рис.20.

## 3.11. Сервисно-ориентированная архитектура

Сервисно-ориентированная архитектура СОА (service-oriented architecture, SOA) представляет собой подход к реализации информационных систем, в которых основное внимание уделяется созданию и использованию служб (service). СОА является преимущественно интеграционной архитектурой, предоставляющей множество механизмов для гибкой интеграции как элементов одной

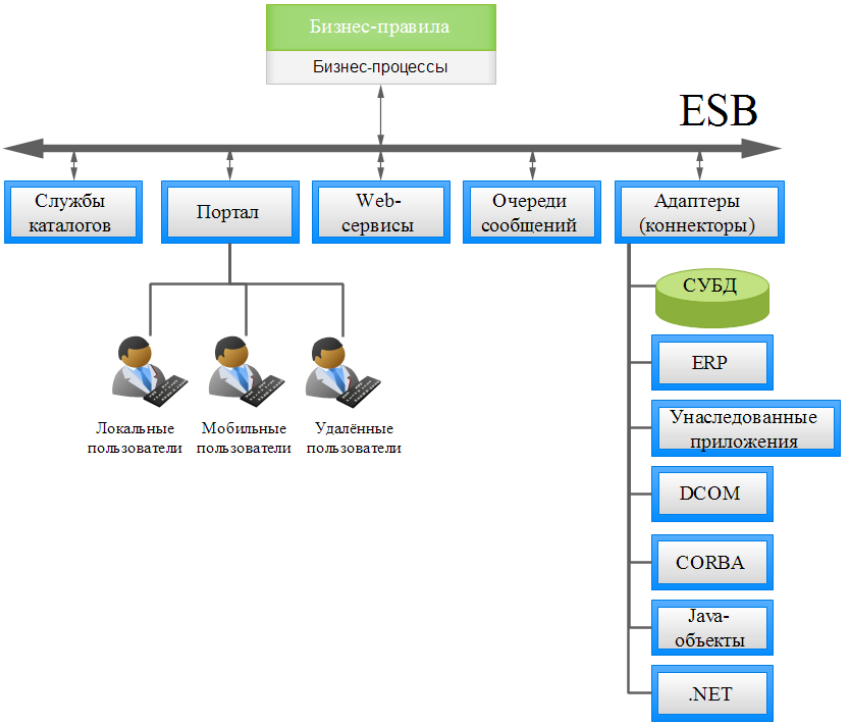



Рисунок 20 - Типовая структура интеграционной системы

системы, так и различных информационных систем. Принцип функционирования заключается в вызове одними приложениями сервисов, входящих в состав других приложений. Созданные службы могут быть объединены для создания бизнес-процессов, которые будут, в дальнейшем, использовать как сервисы.

SOA позволяет решать следующие задачи:

• уменьшение сроков внедрение новых элементов систем

• быстрое создание новых систем на основе существующих решений;

• уменьшение стоимости владения информационной системой;

• уменьшение стоимости интеграции;

• увеличение срока жизни системы;

• реализация бизнес-процессов абстрагировано от приложений и платформ.

Входящие в состав СОА службы должны удовлетворять следующим свойствам:

• представлять собой многократно используемые бизнес-функции;

• определяться с помощью формальных интерфейсов;

• обладать протоколами связи, независящими от языка и платформы реализации, и обеспечивающими доступность информации о местонахождении.

Каждый сервис располагается в определённом месте и удалённо вызывается клиентами. Следовательно, при необходимости внесения изменений, их нужно производить в единственном месте.

Основная концепция СОА заключается в описании сервиса независимо от существующих протоколов. Служба определяется единожды и должна иметь несколько реализаций для работы с различными протоколами.

Для оценки степени эффективности и развитости СОА применяются модели зрелости. Первоначально, при упоминании о зрелости информационных систем, имелась в виду модель CMMI (Capability Maturity Model Integration), однако теперь существуют модели специально разработанные для СОА.

Модель зрелости COA, предложенная группой компаний в главе с Sonic Software Corporation в 2005 году состоит из пяти уровней:

1. Эпизодическое применение COA (Initial Services).
2. Архитектурный уровень (Architected Services).
3. Бизнес-сервисы и службы взаимодействия (Business Services And Collaboration Services).
4. Снятие метрик бизнес-процессов (Measured Business Services).
5. Оптимизация бизнес-процессов (Optimized Business Services).

Уровень эпизодического применения является низшим в модели и предназначен для определения целесообразности применения COA в организации. На этом уровне происходит внедрение основных стандартов XML, WSDL, SOAP, UDDI. Задача первого уровня – ознакомление и накопление опыта при работе с COA. Следует отметить, что уже на первом уровне могут интегрироваться решения на базе ESB. Система соответствует первому уровню зрелости до тех пор, пока не её архитектура не будет задокументирована. При этом не важна сложность самой системы.

Пример системы, соответствующей первому уровню зрелости показан на рис.21

Архитектурный уровень подразумевает переход к плановой интеграции и введению стандартов. Архитектура в организации должна быть задокументирована, план её дальнейшего развития

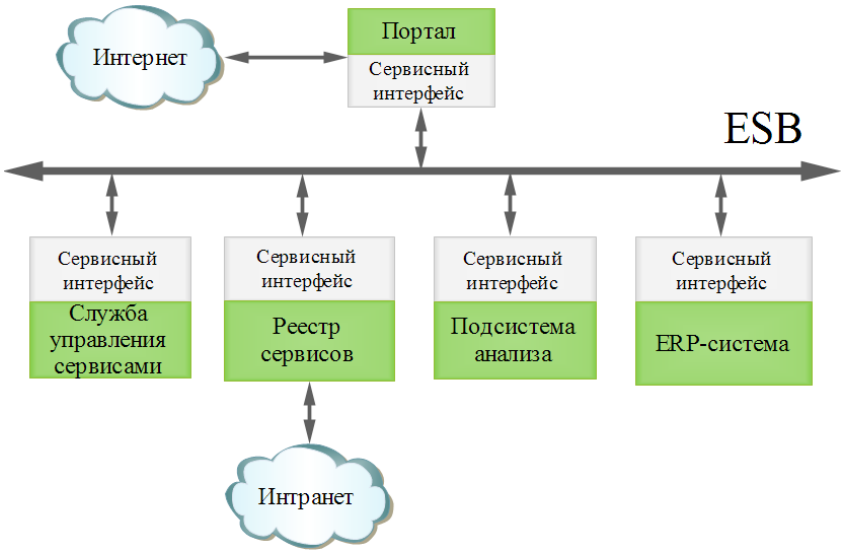



Рисунок 21 - Система первого уровня зрелости СОА

полностью описан, составлены политики безопасности и повторного использования компонентов. Пример подобной системы показан на рис.22.

В такой системе регистр сервисов расширяется до репозитория сервисов и политик, содержащего расширенное описание служб. Для диагностики, обнаружения и реагирования на особые ситуации включается подсистема управления особыми ситуациями. Для поддержки механизмов авторизации добавляется подсистема однократной аутентификации и авторизации.

Уровень бизнес-сервисов и служб взаимодействия характеризуется началом использования систем управления бизнес-

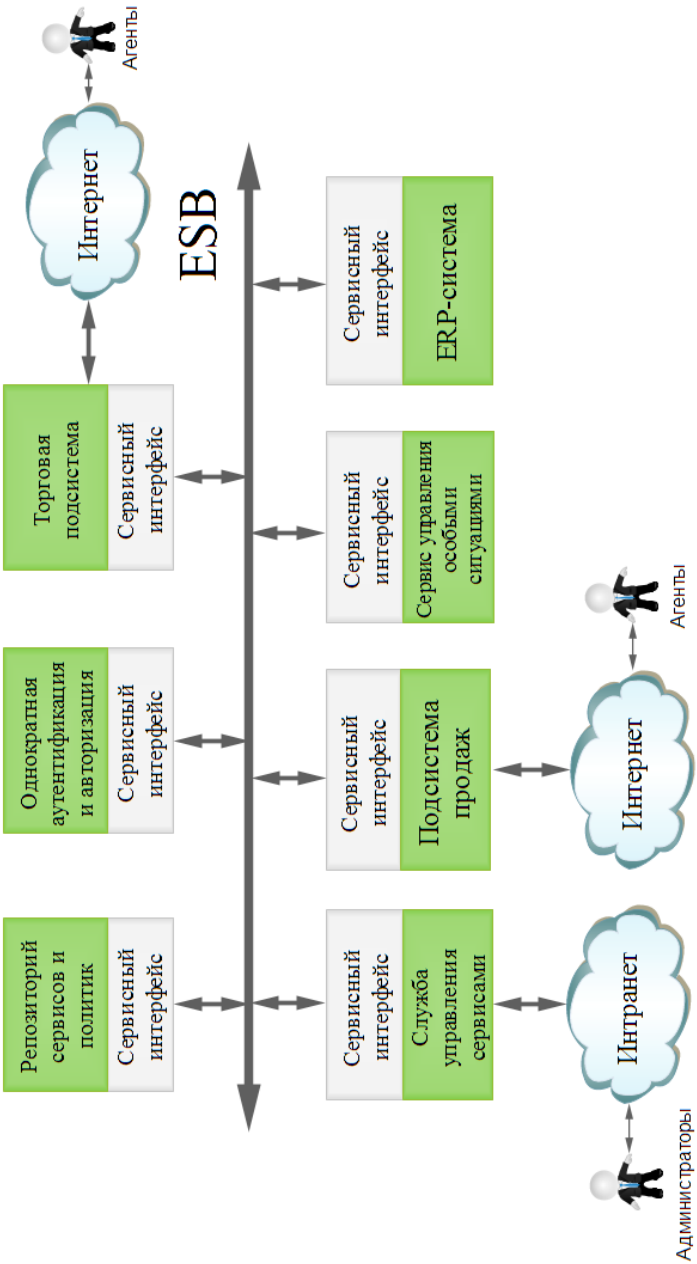



Рисунок 22 – Система второго уровня зрелости СОА

процессами как внутренних сервисов, так и сервисов партнёров. На этом уровне развития активно используется стандарт WSBPEL.

Третий уровень зрелости разделяет сервисы на две группы: бизнес-сервисы (Business Services) и сервисы сотрудничества (Collaboration Services). В связи с этим, в этом уровне можно выделить два подуровня: подуровень бизнес-сервисов, подуровень сервисов сотрудничества. Бизнес-сервисы предназначены для совершенствования внутренних процессов организации, сервисы сотрудничества – взаимодействия между организациями.

Чтобы систему можно было отнести к третьему уровню зрелости в ней должен поддерживаться любой из подуровней (бизнес-сервисы, сервисы сотрудничества).

Отличие данного уровня состоит в переходе от Web-сервисов к бизнес-сервисам.

Пример системы третьего уровня зрелости представлен на рис.23.

Уровень измеряемых бизнес-сервисов характеризуется анализом заданных метрик бизнес-процессов. Анализ чаще всего представляет процесс извлечения знаний из данных – data mining. Четвёртый уровень зрелости требует внедрения ряда новых подсистем:

- подсистема мониторинга бизнес-активностей (Business Activity Monitoring, BAM);
- подсистема работы с бизнес-правилами (Rule Engine);

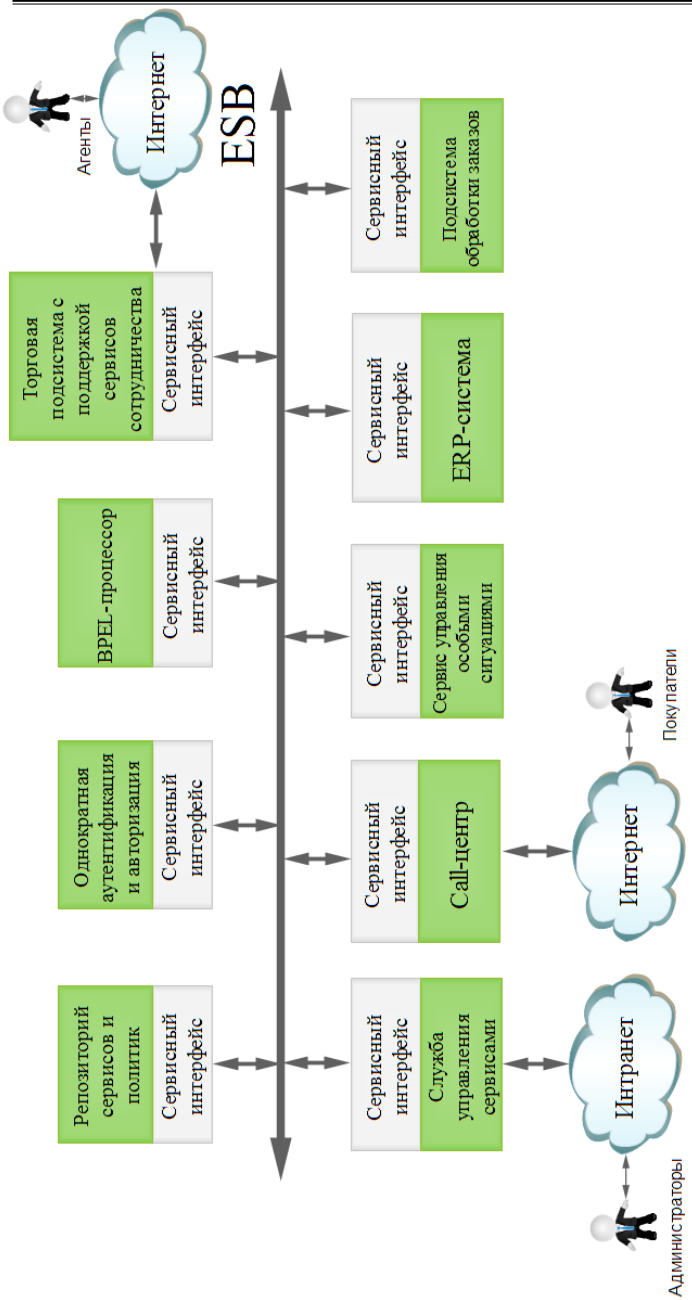



Рисунок 23 – Система третьего уровня зрелости СОА

- подсистема работы с событиями;
- подсистема обработки событий.

Следует отметить, что решения об изменении параметров бизнес-процессов на данном уровне системы всё ещё принимаются человеком.

Пример системы четвёртого уровня зрелости представлен на рис.24.

Уровень оптимизации бизнес-процессов отличается от остальных наличием автоматической реакции на возникающие события. Организация оптимизирует свою деятельность в терминах бизнес-целей и бизнес-правил. Для обеспечения возможности автоматически реагировать на ситуацию, в таких системах введена система поддержки принятия решений, выполняющая два действия:

- определение бизнес-политики и бизнес-правил;
- редактирование правил вследствие результатов мониторинга.

Существуют два режима работы для такой системы: автоматический и режим выдачи рекомендаций (полуавтоматический).

Пример системы пятого уровня зрелости представлен на рис.25.

Модель зрелости СОА содержит как технологические, так и бизнес-аспекты. В частности, принадлежности к первым трём уровням определяется в зависимости от используемых технологий. А последние уровни используют терминологию бизнес-логики, принадлежность к ним определяется по параметрам деятельности бизнеса.

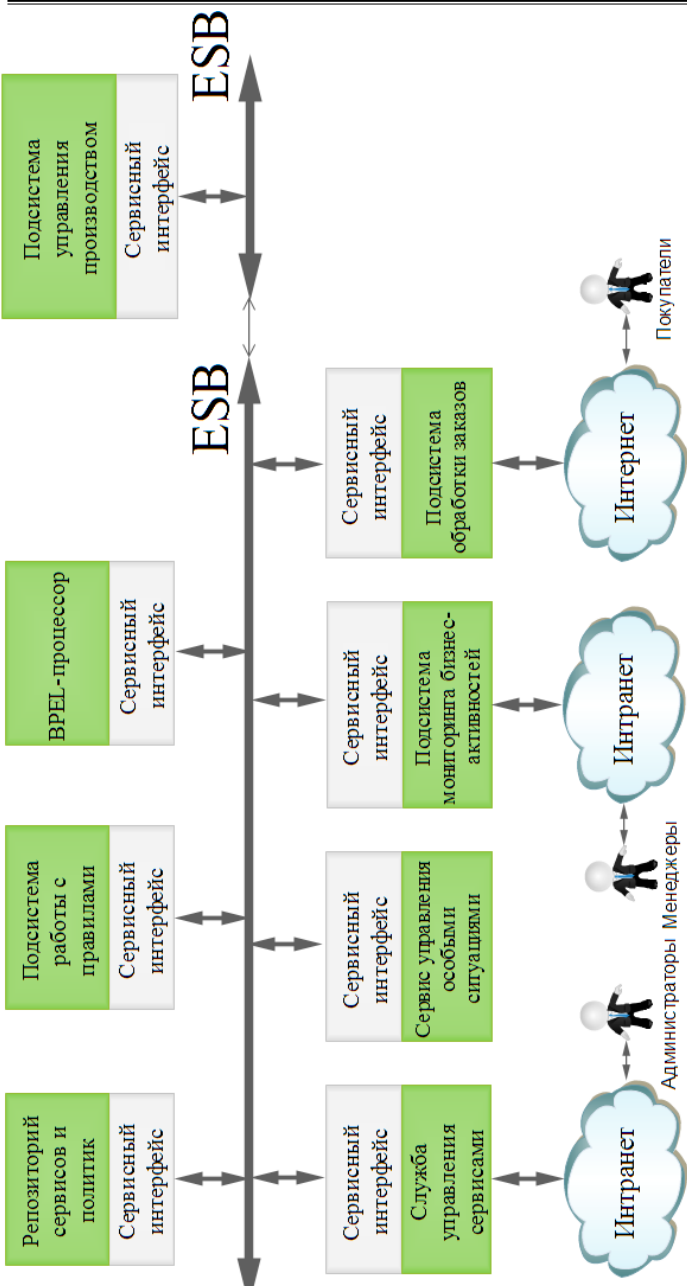



Рисунок 24 - Система четвёртого уровня зрелости СОА

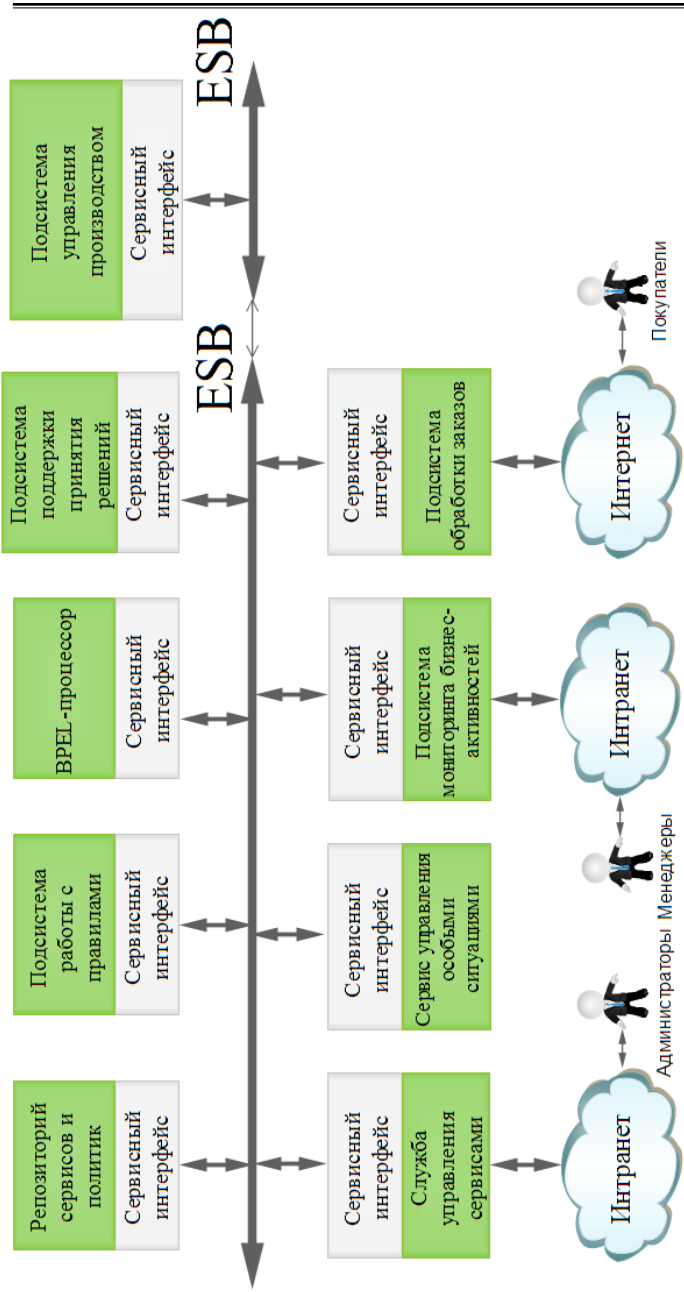



Рисунок 25- Система пятого уровня зрелости СОА

Существует ещё одна модель зрелости для СОА – Модель зрелости сервисно-ориентированной организации (Maturity Model for Service Oriented Enterprises, SOE). Она была разработана IBM и BEA Systems. Служит для определения зрелости моделей систем в бизнес-терминах. Следует отметить, что данная модель также рассматривает методы возврата инвестиций, потраченных на создание СОА.

SOE определяет пять уровней:

1. Базовый уровень (SEO foundation). На этом уровне должна быть создана сервисно-ориентированная инфраструктура, определены все бизнес-процессы, составлен список показателей эффективности. Первый уровень предполагает большие затраты и отсутствие возврата инвестиций.

2. Повторяемый, ориентированный на Интранет (SOE repeatable projects – infra-focused). Данный уровень предполагает наличие развитой СОА, возможно с несколькими ESB, спецификацию задач, определение потенциальных пользователей и их ролей. Основное внимание уделяется использованию СОА внутри организации. На втором уровне начинает процесс возврата инвестиций, связанный с уменьшением стоимости внедрения бизнес-процессов.

3. Расширенный, ориентированный на организацию (SOE extended enterprise-focused). Характеризуется объединением бизнес-процессов различных организаций. На данном уровне создаётся система мониторинга бизнес-процессов, протекающих в нескольких

организациях. Возврат инвестиций достигается за счёт уменьшения стоимости поддержания партнёрский связей.

4. Ориентированный на решения (SOE solution focused). Данный уровень подразумевает совершенствование инфраструктуры, поддерживающей сервисы. Основное внимание уделяется созданию новых горизонтальных и вертикальных решений реализации бизнес-процессов (в частности, внедрение фреймворков). Возврат инвестиций достигается за счёт снижения стоимости внедрения новых бизнес-процессов.

5. Оптимизация производительности, гибкости и интеллектуализация (SOE performance, agility and intelligence focused). Последний уровень подразумевает сосредоточение внимания на решение проблем, связанных с разработкой гибкий систем, адаптирующихся к изменениям среды. Интеллектуализация может быть представлена в виде семантического поиска сервисов. Возврат инвестиций достигается за счёт уменьшения стоимости владения системой при быстро изменяющейся среде.

Рассмотренные модели зрелости считаются взаимодополняющими. Модель СОА предназначена для интеграторов, а модель SOE для управленцев.

# 4. Анализ и моделирование бизнес-процессов при проектировании информационных систем

## 4.1. Технология описания бизнес-процессов при проектировании информационных систем

Технология описания бизнес-процессов реализуется через процесс подробного описания всех бизнес-процессов компании в виде графических моделей с обозначением всех взаимных связей. Для максимальной эффективности этой технологии необходимо иметь утвержденную методологию, квалифицированный персонал и средство описания бизнес-процессов.

Моделирование и анализ бизнес-процессов параллельно может быть необходимо и для решения других задач компании, а не только в процессе модернизации информационныой системы. Например, таких как:

- для сертификации системы менеджмента качества по стандартам ИСО серии 9000;
- для оптимизации структуры компании и ее численности;
- для формирования корпоративных регламентов;
- для разработки должностных инструкций;
- для модернизации производства и реинжиниринге компании;
- для минимизации затрат и повышения эффективности компании.

Технология описания бизнес-процессов включает в себя определенную методологию, которая включает в себя:

1. Описание организационной структуры компании. Строится иерархическая модель или модели компании по принципу подчиненности и взаимосвязанности.

2. Описание информационной системы компании. Разрабатывается модель информационной системы, обозначающая все модули и применяемое программное обеспечение.

3. Описание функций подразделений компании. Разрабатывается функциональная модель компании с подробным описанием всех функций структурных подразделений.

4. Описание выбранных бизнес-процессов компании. Разрабатываются модели всех бизнес-процессов компании.

5. Описание продуктов и услуг компании. Строится модель продуктов и услуг, которые производятся или реализуются компанией как на внешнем, так внутреннем рынке.

6. Определение и построение дерева бизнес-процессов компании. Осуществляется классификация всех бизнес-процессов на основные, обеспечивающие и управления с последующим построение модели бизнес-процессов компании.

7. Описание информационных потоков всех бизнес-процессов. Происходит процесс идентификации информационных потоков выявленных бизнес-процессов с последующим построением модели информационных потоков.

Перечень средств описания бизнес-процессов достаточно широк и хорошо представлен на россий-

ском рынке. От самых простых программ типа MS Visio, до специализированных типа ARIS и CaseWise. При этом существуют как дорогостоящие профессиональные программные средства, так и бесплатные или учебные версии известных продуктов. Выбор средств описания бизнес-процессов чаще всего определяется их сложностью, глубиной и требованиями заказчика к качеству описания и моделирования бизнес-процессов компании.

Опыт успешных компаний показывает необходимость постоянного совершенствования информационных систем, а значит и бизнес-процессов. Это позволяет им находиться в состоянии постоянного поиска наиболее оптимальных технологий, сокращать издержки, добиваться высокой эффективности производства и всей компании, принимать взвешенные и продуманные решения на основе фактов и знания реальной обстановки.

## 4.2. Методы анализа и оптимизации бизнес-процессов

Интерес к управлению бизнес-процессами в компаниях вызван не только необходимостью разработки и внедрения современных информационных систем, а прежде всего необходимостью быстрой реакции на изменяющиеся условия рыночной среды, особенно в условиях мирового экономического кризиса. Своевременный анализ бизнес-процессов позволяет грамотно осуществлять регламентацию и управление ими, совершенствуя сам процесс и информационную систему.

Методы оптимизации бизнес-процессов:

1. Минимизация участников бизнес-процесса.
2. Типизация бизнес-процессов.
3. Упрощение бизнес-процессов.
4. Сокращение времени протекания бизнес-процесса.
5. Обеспечение параллельности бизнес-процессов.
6. Исключение излишнего контроля в бизнес-процессе.
7. Минимизация участия руководства в бизнес-процессах.
8. Постоянное совершенствование бизнес-процесса.
9. Реинжиниринг бизнес-процессов.
10. Изучение и внедрение опыта лучших компаний (бенчмаркинг).
11. Сокращение стоимости бизнес-процессов.

Принципы осуществления реинжиниринга бизнес-процессов:

- твердая решимость и воля руководителя;
- интеграция рабочих этапов бизнес-процесса в один;
- единообразие производства в рамках одного бизнес-процесса;
- минимизация согласований;
- целесообразность места бизнес-процесса;
- назначение менеджера бизнес-процесса;
- минимизация проверок и управляющих воздействий;

- применение смешанного подхода к управлению;
- повышение ответственности исполнителей бизнес-процесса;
- расчет необходимых ресурсов для бизнес-процессов.

Реинжиниринг в настоящее время один самых распространенных способов повышения эффективности компаний, широко использующийся в процессе ликвидации последствий мирового экономического кризиса.

## 4.3. Моделирование бизнес-процессов (Business Process Modeling) при проектировании информационных систем

Моделирование бизнес-процессов – сложный процесс описания различными методологиями реальных бизнес- процессов с целью повышения эффективности в процессе разработки и внедрения информационной системы или реинжиниринга. Сложился определенный порядок моделирования бизнес-процессов:

1. Сбор и изучение всей информации о бизнес-процессе.
2. Разработка модели «как есть» для бизнес-процесса.
3. Анализ реального бизнес-процесса.
4. Разработка показателей эффективности бизнес-процесса.
5. Выявление проблем выполнения бизнес-процесса.

6. Разработка модели «как должно быть» для бизнес-процесса.

7. Разработка регламента бизнес-процесса «как должно быть».

8. Разработка плана внедрения бизнес-процесса «как должно быть».

9. Обучение участников бизнес-процесса.

10. Введение бизнес-процесса в опытную эксплуатацию.

11. Доработка бизнес-процесса и ввод в эксплуатацию.

12. Контроль показателей эффективности бизнес-процесса.

### Методологии процесса моделирования бизнес-процессов в нотациях IDEF [11]

**IDEF** – Сокращение от **Integration Definition Metodology** (Объединение Методологических Понятий). Семейство совместно используемых методов для процесса моделирования. IDEF технология используется, начиная с конца 1980-х годов. Department of Defense USA (Министерство обороны США) является основным пользователем данной технологии. Ей, также, пользуются некоторые крупные корпорации в США.

**IDEF0** (**Function Modeling**) – данный метод используется для создания функциональной модели, которая является структурированным отображением функций производственной системы или среды, а также информации и объектов, связывающих эти функции.

**IDEF1** (**Information Modeling**) – данный метод применяется для построения информационной моде-

ли, которая представляет собой структурированную информацию, необходимую для поддержки функций производственной системы или среды.

**IDEF2** (**Simulation Model Design**) – данный метод позволяет построить динамическую модель меняющегося во времени поведения функций, информации и ресурсов производственной системы или среды. Данная модель используется редко. В основном востребована на предприятиях, где необходимо описать непрерывную дейтельность на конвейерах или аналогичные функции.

**IDEF3** (**Process Description Capture**) – данный метод используется для сбора информации о состоянии моделируемой системы. Это структурный метод, показывающий причинно-следственные связи и события. Он также показывает, как организована работа, и какие пользователи работают с моделируемой системой. IDEF3 состоит из двух методов.

**Process Flow Description (PFD)** – описание процессов, с описанием того, как организована работа между различными элементами моделируемой системы. **Object State Transition Description (OSTD)** – описание переходов состояний объектов, с описанием того, какие существуют промежуточные состояния у объектов в моделируемой системе.

**IDEF4 (Object-Oriented Design)** – данный метод объектно-ориентированного планирования был разработан для поддержки объектно-ориентированной идеологии. Подробнее - Технология UML.

**IDEF5 (Ontology Description Capture)** – данный метод позволяет разрабатывать, изучать и

поддерживать онтологию моделируемой системы. Термин «онтология» включает в себя каталог терминов области знаний; правила, объясняющие, как термины могут комбинироваться, создавая при этом корректные ситуации в области знаний и согласованные выводы, используемые в моделируемой системе.

**IDEF6 (Design Rational Capture Method)** - данный метод позволяет использовать рациональный опыт проектирования.

**IDEF7 (Information System Auditing)** - данный метод описывает проведение методологии аудита информационной системы.

**IDEF8 (User Interface Modeling)** – данный метод позволяет разрабатывать необходимые модели Графического Интерфейса Пользователя (Human-System Interaction Design). Метод предназначена для проектирования взаимодействия человека и технической системы.

**IDEF9 (Business Constraint Discovery)** - данная модель предназначена для анализа имеющихся условий и ограничений (в том числе физических, юридических или любых других) и их влияния на принимаемые решения в процессе реинжиниринга.

**IDEF10 - Implementation Architecture Modeling**.

**IDEF11 - Information Artifact Modeling**.

**IDEF12 - Organization Modeling**.

**IDEF13 - Three Schema Mapping Design**.

**IDEF14** (**Network Design**) - данный метод позволяет моделировать вычислительные сети. Модель предназначена для представления и анализа

данных при проектировании вычислительных сетей на графическом языке с описанием конфигураций, очередей, сетевых компонентов, требований к надежности.

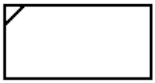
Наиболее популярные методологии моделирования бизнес-процессов и их анализа:

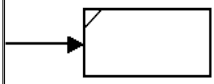
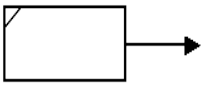
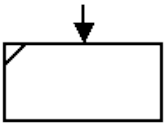
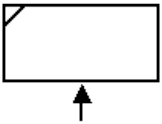
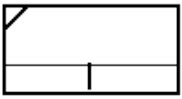
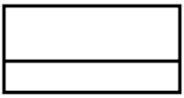
- Моделирование бизнес-процессов в нотации IDEF0 предназначено для функционального описания бизнес-процессов.
- Моделирование рабочих потоков в нотации IDEF3 предназначено для описания рабочих процессов.
- Моделирование потоков данных в нотации DFD предназначено для описания потоков информации в процессе выполнения работ.

Моделирование в нотации IDEF0 представляет из себя рассмотренную ранее систему из блоков и дуг (стрелочек), обозначающих внешние связи бизнес-процесса и его декомпозицию. Нотации IDEF0 и IDEF3 используют следующие объекты, представленные в таблицах 5 и 6.

Таблица 5.

**Графические объекты представления бизнес-процессов**

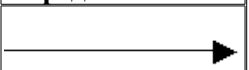
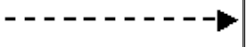
| № | Наименование | Описание | Графическое представление |
|---|---|---|---|
| **Нотация IDEF0** | | | |
| 1 | Модуль поведения (UOB) | Объект служит для описания функций (процедур, работ), выполняемых подразделениями/ сотрудниками предприятия. |  |

| | | | |
|---|---|---|---|
| 2 | Стрелка слева | Стрелка описывает входящие документы, информацию, материальные ресурсы, необходимые для выполнения функции. |  |
| 3 | Стрелка справа | Стрелка описывает исходящие документы, информацию, материальные ресурсы, являющиеся результатом выполнения функции. |  |
| 4 | Стрелка сверху | Стрелка описывает управляющее воздействия, например распоряжение, нормативный документ и т.д. В нотации IDEF0 каждая процедура должна обязательно иметь не менее одной стрелки сверху, отражающей управляющее воздействие. |  |
| 5 | Стрелка снизу | Стрелка снизу описывает т.н. механизмы, т.е. ресурсы, необходимые для выполнения процедуры, но не изменяющие в процессе ее выполнения свое состояние. Примеры: сотрудник, станок и т.д. |  |
| **Нотация IDEF3** | | | |
| 1 | Модель работы (UOW) | Объект служит для описания функций (процедур, работ), выполняемых подразделениями/ сотрудниками предприятия. |  |
| 2 | Ссылочный объект | Объект, используемый для описания ссылок на другие диаграммы модели, циклические переходы в рамках одной модели, различные комментарии к функциям |  |

| | | | |
|---|---|---|---|
| 3 | Логическое «И» | Логический оператор, определяющий связи между функциями в рамках процесса. Позволяет описать ветвление процесса |  |
| 4 | Логическое «ИЛИ» | Логический оператор, определяющий связи между функциями в рамках процесса. Позволяет описать ветвление процесса |  |
| 5 | Логическое исключающее «ИЛИ» | Логический оператор, определяющий связи функциями в рамках процесса. Позволяет описать ветвление процесса |  |

Таблица 6.

**Типы стрелок в нотациях IDEF0 и IDEF3**

| № | Тип стрелки | Графическое представление |
|---|---|---|
| 1 | Стрелка предшествования. Соединяет последовательно выполняемые функции. |  |
| 2 | Стрелка отношения. Используется для привязки объектов-комментариев к функциям. |  |
| 3 | Стрелка потока объектов. Показывает поток объектов от одной функции к другой. |  |

Пример построения диаграмм в нотации IDEF0 представлен на рис. 26 для оптового магазина. Декомпозиция этого бизнес-процесса изображена на рис.27.

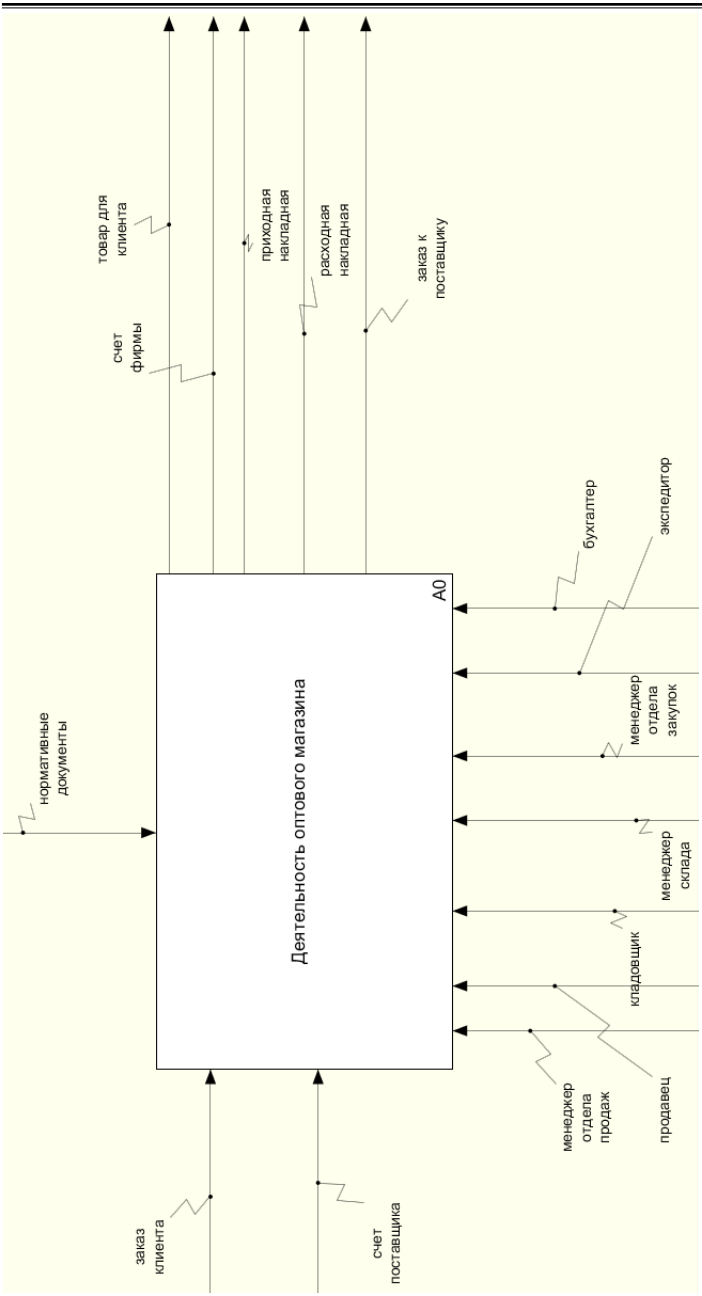



Рисунок 26 – Диаграмма моделирования бизнес-процесса в нотации IDEF0

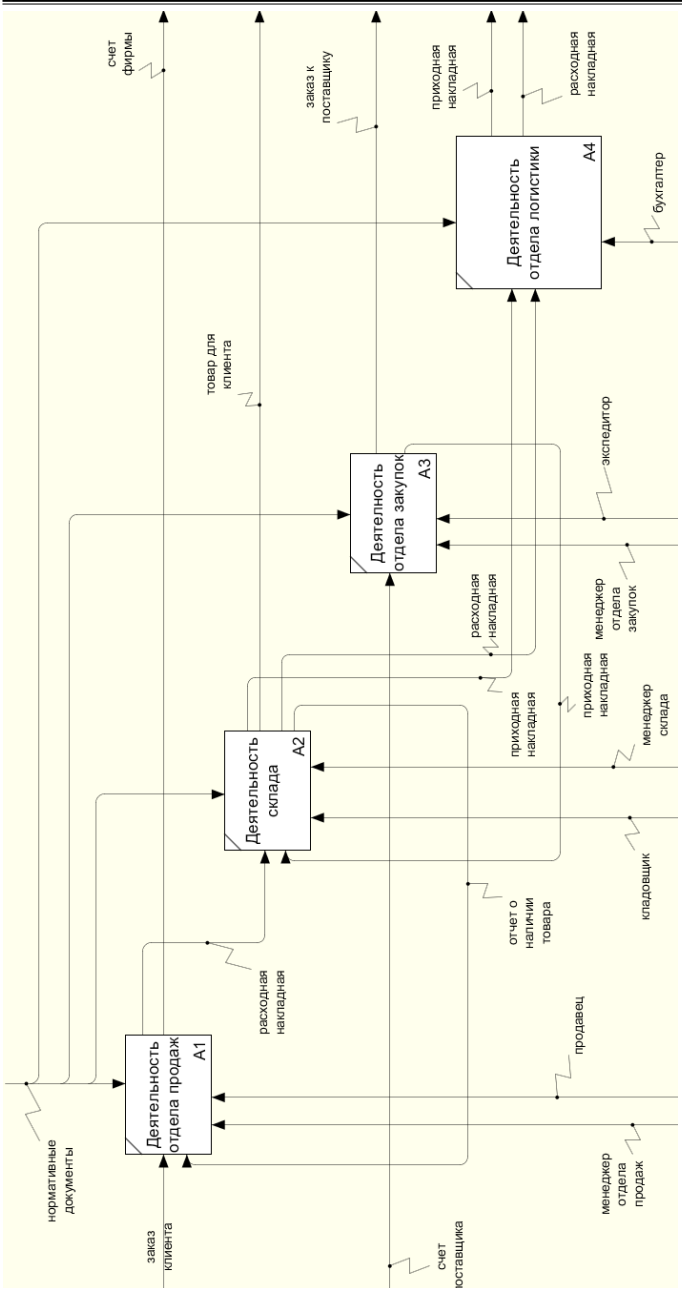



Рисунок 27 – Декомпозиция бизнес-процесса в нотации IDEF0

Моделирование рабочих потоков в нотация IDEF3 представляет из себя диаграмму последовательности выполнения определенных действий и взаимозависимости между ними.

Диаграммы IDEF3 представляют из себя набор фигур с обозначенными действиями. Все действия именуются с использованием глаголов, каждому из действий присваивается уникальный идентификационный номер. Все связи в IDEF3 являются однонаправленными и организуются слева направо.

Типы связей в нотации IDEF3:

- Временное предшествование (Temporal precedence), простая стрелка. Исходное действие должно завершиться, прежде чем конечное действие сможет начаться.
- Объектный поток (Object flow), стрелка с двойным наконечником. Выход исходного действия является входом конечного действия. Исходное действие должно завершиться, прежде чем конечное действие сможет начаться. Наименования потоковых связей должны чётко идентифицировать объект, который передается с их помощью.
- Нечеткое отношение (Relationship), пунктирная стрелка.

Пример диаграммы в нотации IDEF3 представлен на рис. 28.

Завершение одного действия может инициировать начало выполнения сразу нескольких других действий, или наоборот, определенное действие может требовать завершения нескольких других действий до начала своего выполнения (ветвление процесса).

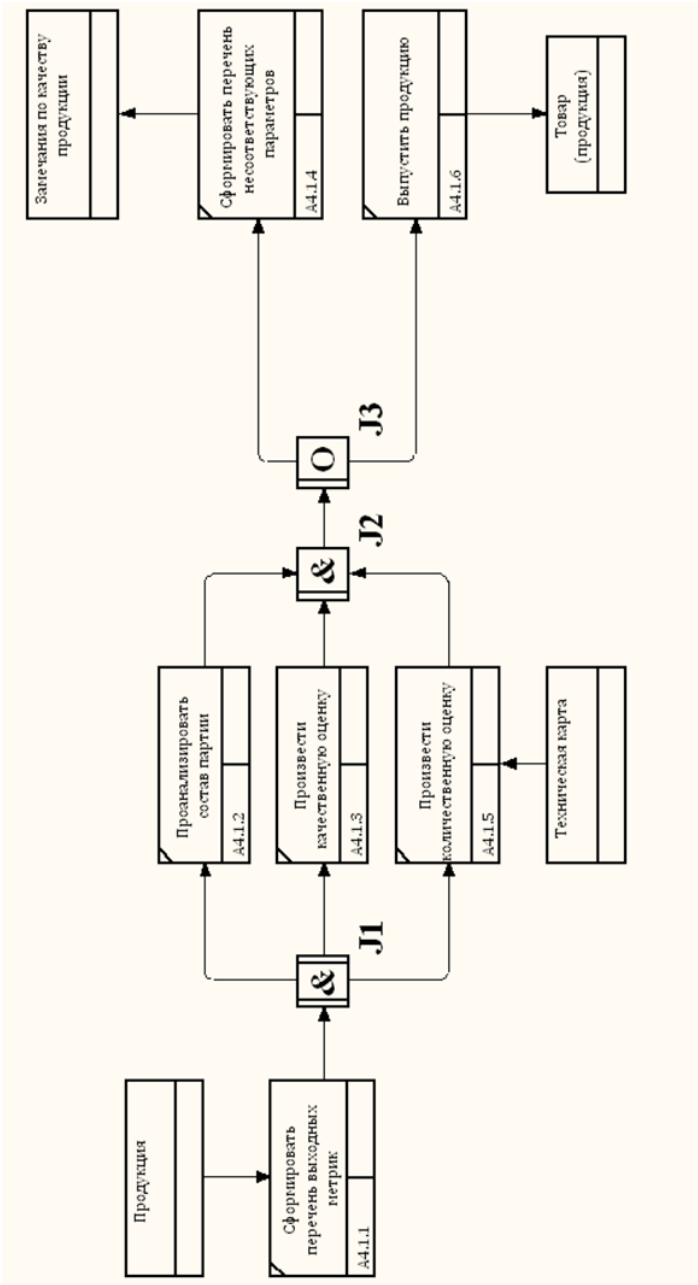



Рисунок 28 - Диаграмма в нотации IDEF3

Если действия “И”, “ИЛИ” должны выполняться синхронно, это обозначается двумя двойными вертикальными линиями внутри блока, асинхронно - одной.

Диаграммы в нотации DFD показывают, каким образом каждый процесс преобразует свои входные данные в выходные. При этом отображаться могут не только информационные, но и материальные потоки. Процесс может быть декомпозирован.

Основными компонентами диаграмм потоков данных являются:

- внешние сущности (материальный объект или физическое лицо, являющиеся источником или приёмником информации, например, заказчики, персонал, поставщики, клиенты, склад);
- системы и подсистемы (например, подсистема по работе с физическими лицами);
- процессы (преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом; физически это может быть, например, подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т.д.);
- накопители данных (абстрактные устройства для хранения информации);
- потоки данных (на диаграмме - стрелки).

Пример диаграммы DFD представлен на рис. 29.

Диаграммы потоков данных DFD используются для построения моделей «как есть» и «как надо», отражая, существующую и предлагаемую структуру бизнес-процессов организации.

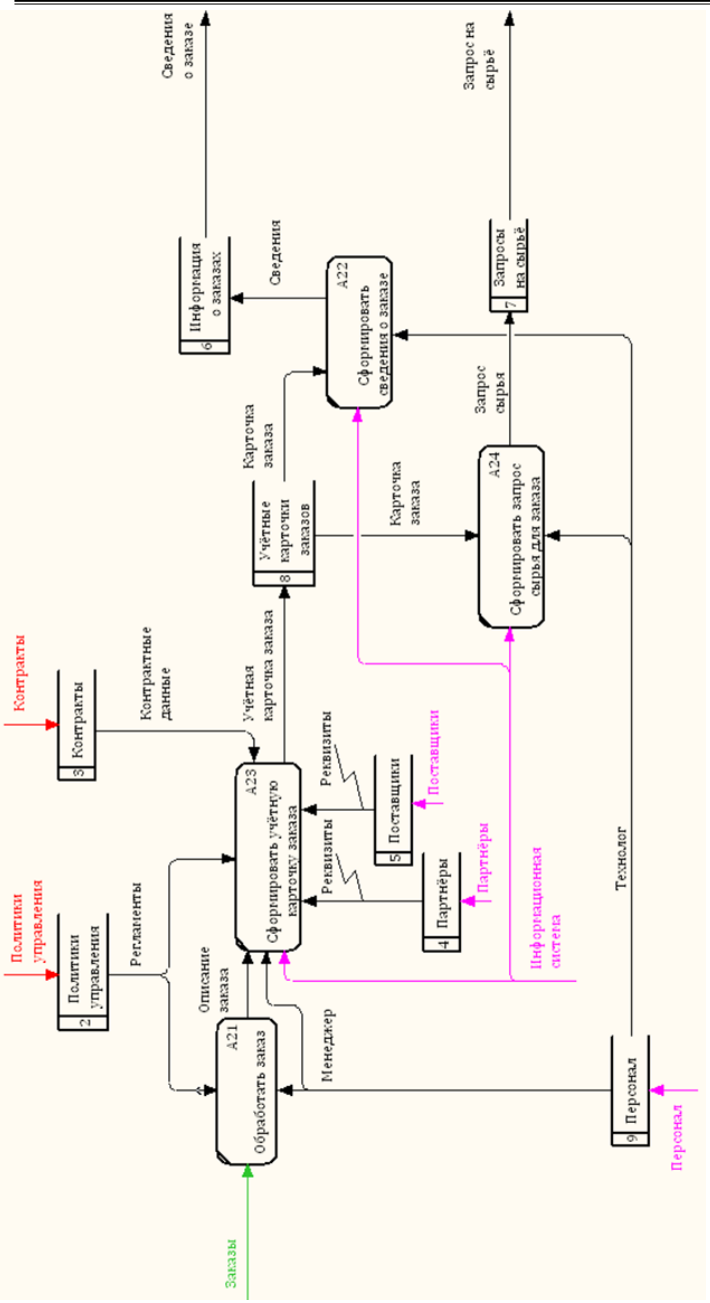



Рисунок 29 – Диаграмма потоков данных DFD

# 5. Автоматизированное проектирование информационных систем на основе CASE- технологии

## 5.1 Назначение CASE-средств

Учитывая постоянный рост требований, сильно увеличивается сложность современных информационных систем. Можно выделить следующие особенности крупных существующих информационных систем:

- сложность описания;
- наличие совокупности тесно взаимодействующих компонентов;
- отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;
- необходимость интеграции существующих и вновь разрабатываемых приложений;
- функционирование в неоднородной среде на нескольких аппаратных платформах;
- разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;
- существенная временная протяженность проекта разработки.

Создание всей требуемой проектной документации вручную является крайне сложной задачей, а редактирование созданного пакета документов влечёт за собой ещё большие трудности. В связи с этим, можно

выявить следующие проблемы, возникающие при ручном процессе проектирования:

- неадекватная спецификация требований;
- неспособность обнаруживать ошибки в проектных решениях;
- низкое качество документации, снижающее эксплуатационные качества;
- затяжной цикл и неудовлетворительные результаты тестирования.

Существенное подспорье в решении подобных проблем вносят CASE-средства (Computer Aided Software Engineering). Под CASE-средством понимается специальное программное обеспечение, поддерживающее процессы создания и сопровождения информационных систем: анализ и формулировка требований, проектирование прикладного программного обеспечения и баз данных, генерация кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы.

Полная среда разработки информационной системы представляет из себя совокупность используемых CASE-средств, системного программного обеспечения и технических средств.

Следует заметить, что CASE-средства далеко не всегда сразу же дают ожидаемый эффект, кроме того, реальный бюджет, требуемый на внедрение, в большинстве случаев существенно превышает их рыночную стоимость.

Чтобы увеличить шансы на успешное внедрение CASE-средства, необходимо руководствоваться тремя аспектами:

- технология (ограниченность существующих возможностей);
- культура (готовность к внедрению новых процессов);
- управление (четкое руководство важными этапами и процессами внедрения).

Недостаток внимания к какому-либо аспекту может негативным образов сказаться на успешности процесса внедрения, даже, несмотря на скрупулёзное следование существующим рекомендациям.

## 5.2 Состав и классификация CASE-средств

CASE-средства обладают мощными графическими средствами описания и документирования информационных систем, обеспечивают управляемость процесса разработки, за счёт интеграции некоторых компонент, а также позволяют централизованно хранить данные при помощи репозиториев.

Конкретная CASE-технология включает в себя методологию проектирования информационных систем и инструментальные средства анализа и моделирования.

Архитектуру CASE-средства можно представить в виде совокупности шести компонентов (рис.30):

1. Репозиторий данных.
2. Графический редактор диаграмм.
3. Верификатор диаграмм.
4. Генератор отчётов.
5. Администратор проекта.
6. Сервис.

Репозиторий представляет собой базу данных, предназначенную для обмена информацией между


Рисунок 30 - Компоненты CASE-средства

компонентами CASE-средства, а также для хранения сведений обо всех объектах проектируемой системы.

Графический редактор диаграмм предназначен для отображения проектируемой информационной системы в заданной графической нотации. Позволяет выполнять следующие действия:

- создавать элементы диаграмм и взаимосвязи между ними;
- задавать описания элементов диаграмм;
- задавать описания связей между элементами диаграмм;
- редактировать элементы диаграмм, их взаимосвязи и описания.

Верификатор диаграмм выявляет несоответствия разрабатываемой диаграммы

методологии проектирование. Среди его функций можно выделить:

- мониторинг правильности построения диаграмм;
- диагностику и выдачу сообщений об ошибках;
- выделение на диаграмме ошибочных элементов.

Генератор отчётов позволяет получать информацию о состоянии проекта в виде, формируемых по различным признакам, отчётов.

Администратор проекта представляет собой набор инструментальных средств, необходимых для выполнения административных функций. К таким функциям относятся:

- инициализация проекта;
- задания начальных параметров проекта;
- назначения и изменения прав доступа к элементам проекта;
- мониторинга выполнения работ.

Компонент сервиса представляет собой набор системных утилит для обслуживания репозитория данных. Используется для архивации данных, восстановления данных и создания нового репозитория.

В зависимости от области применения различные CASE-средства могут содержать следующие компоненты:

- репозиторий;
- графические средства анализа и проектирования;
- средства разработки приложений;
- средства конфигурационного управления;

- средства документирования;
- средства тестирования;
- средства управления проектом;
- средства реинжиниринга.

Типовая классификация CASE-средств по большей части соотносится с их компонентным составом. Можно разделить CASE-средства на следующие типы:

- средства моделирования предметной области (построения и анализа моделей предметной области);
- средства анализа и проектирования (создание спецификации компонентов, интерфейсов системы, архитектуры системы, алгоритмов и структур данных);
- средства проектирования баз данных (построение моделей данных);
- средства разработки приложений;
- средства реинжиниринга (анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций);
- средства планирования и управления;
- средства конфигурационного управления;
- средства тестирования;
- средства документирования.

В исторической ретроспективе можно выделить CASE-системы двух поколений:

1. Первое поколение. Обеспечивает:
   - поддержку графических моделей;
   - проектирование спецификаций;
   - проектирование экранных редакторов;
   - проектирование словарей данных.

2. Второе поколение. Обеспечивает:

- поддержку графических представлений требований к системе;
- поддержку представлений спецификаций проектирования;
- поддержку контроля и анализа системной информации,
- информационную поддержку управления проектированием,
- построение прототипов и моделей системы;
- автоматическую кодогенерацию;
- поддержку тестирования, верификации и анализа сгенерированных программ;
- генерацию документов по проекту;
- контроль на соответствие стандартам по всем этапам ЖЦ;

Большая часть подобных технологий основывается на методологиях структурного и объектно-ориентированного анализа. Представление полученных данных производится при помощи текстов и диаграмм.

## 5.3 Технология внедрения CASE-средств

Процесс внедрения CASE-средств состоит из следующих этапов:

1. Определение потребностей в CASE-средствах.
2. Оценка и выбор CASE-средств.
3. Выполнение пилотного проекта.
4. Практическое внедрение CASE-средств.

На первом этапе необходимо выявить области деятельности компании, в которых обоснованно применение CASE-средств. По завершению данного этапа формируется документ описывающий стратегию внедрения.

Для успешного завершения этапа определения потребностей необходимо:

- проанализировать возможности организации в отношении её технической базы, персонала и используемого программного обеспечения;
- определить организационные потребности на основании проблем и целей;
- проанализировать рынок CASE-средств на предмет соответствия желаемым требованиям;
- определить критерии успешного внедрения CASE-средства и их оптимальные значения;
- разработать стратегию внедрения.

Стратегия внедрения включает следующие составляющие:

- организационные потребности;
- базовые метрики, необходимые для последующего сравнения результатов;
- критерии успешного внедрения;
- подразделения организации, в которых должно выполняться внедрение CASE-средств;
- влияние, оказываемое на другие подразделения организации;
- стратегии и планы оценки и выбора, пилотного проектирования и перехода к полномасштабному внедрению;

- основные факторы риска;
- ориентировочный уровень расходов и источники финансирования процесса внедрения CASE-средств;
- ключевой персонал и другие ресурсы.

Вторым этапом является процесс оценки функциональности и качества CASE-средств, для последующего выбора подходящих. Оценка выполняется в соответствии с конкретными критериями, ее результаты включают как объективные, так и субъективные данные по каждому средству.

Процесс оценки включает следующие действия:

- формулировка задачи оценки, включая информацию о цели и масштабах оценки;
- определение критериев оценки, вытекающее из определения задачи;
- определение средств-кандидатов путем просмотра списка кандидатов и анализа информации о конкретных средствах;
- оценка средств-кандидатов в контексте выбранных критериев;
- подготовка отчета по результатам оценки.

Процесс выбора тесно взаимосвязан с процессом оценки и включает следующие действия:

- формулировка задач выбора, включая цели, предположения и ограничения;
- определение и ранжирование критериев
- определение средств-кандидатов и сбор необходимых данных;

- выполнение необходимого количества итераций с тем, чтобы выбрать (или отвергнуть) средства, имеющие сходные показатели;
- подготовка отчета по результатам выбора.

Каждый критерий, используемый в процессах оценки и выбора, должен быть выбран и адаптирован экспертом с учетом особенностей конкретного процесса. Выбор и уточнение набора используемых критериев является критическим шагом данного процесса.

Можно выделить следующие группы критериев:

1. Функциональность: обеспечение требуемых функций, наличие дополнительных нерегламентированных функций, возможность расширения набора функций.
2. Надежность: обеспечение целостности данных и их резервирование, защита от несанкционированного доступа, обнаружение ошибок, анализ отказов.
3. Простота использования: удобство пользовательского интерфейса, локализация, простота освоения, качество документации, доступность и качество учебных материалов, требования к уровню знаний; унифицированность пользовательского интерфейса, онлайновые подсказки, понятность и полезность диагностических сообщений, допустимое время реакции на действия пользователя, простота установки и обновления версий.
4. Эффективность: требования к техническим средствам, эффективность выполнения CASE-средством своих функций в зависимости от интенсивности работы пользователя, производительность.

5. Сопровождаемость: уровень поддержки со стороны поставщика, простота освоения отличий новых версий от существующих, совместимость обновлений, сопровождаемость конечного продукта.

6. Переносимость: совместимость с различными версиями операционных систем, переносимость данных между различными версиями CASE-средства, соответствие стандартам переносимости.

7. Общие критерии: затраты на CASE-средство, оценочный эффект от внедрения CASE-средства, общие показатели возможностей дистрибьютора, лицензионная политика, экспортные ограничения; общая информация о продукте, поддержка поставщика и качество предоставляемых услуг.

Перед полномасштабным внедрением выбранного CASE-средства в организации выполняется пилотный проект, целью которого является экспериментальная проверка правильности решений, принятых на предыдущих этапах, и подготовка к внедрению.

Пилотный проект представляет собой реальное использование CASE-средства в предназначенной для этого среде и обычно подразумевает более широкий масштаб использования CASE-средства по отношению к тому, который был достигнут во время оценки. Пилотный проект должен обладать многими из характеристик реальных проектов, для которых предназначено данное средство. Целями пилотного проекта являются:

- подтверждение достоверности результатов оценки и выбора;

- определение годности CASE-средства для использования в организации и область его применения;
- сбор информации, необходимой для разработки плана практического внедрения;
- приобретение собственного опыт использования CASE-средств.

Важной функцией пилотного проекта является принятие решения относительно приобретения или отказа от использования CASE-средства. Первоначальное использование новой CASE-технологии в пилотном проекте должно тщательно планироваться и контролироваться.

Пилотный проект включает в себя следующие шаги:

- планирование пилотного проекта;
- выполнение пилотного проекта;
- оценка пилотного проекта.

Планирование пилотного проекта должно вписываться в обычный процесс планирования проектов в организации. План должен содержать следующую информацию:

- цели, задачи и критерии оценки;
- персонал;
- процедуры и соглашения;
- обучение;
- график и ресурсы.

Ожидаемые результаты пилотного проекта должны быть четко определены. Степень соответствия этим результатам представляет собой основу для последующей оценки проекта.

Специалисты, выбранные для участия в пилотном проекте, должны иметь соответствующий авторитет и влияние и быть сторонниками новой технологии. Группа должна включать как технических специалистов, так и менеджеров, заинтересованных в новой технологии и разбирающихся в ее использовании.

Необходимо четко определить процедуры и соглашения, регулирующие использование CASE-средств.

Должны быть определены виды и объем обучения, необходимого для выполнения пилотного проекта. При планировании обучения нужно иметь в виду три вида потребностей: технические, управленческие и мотивационные.

Должен быть разработан график, включающий ресурсы и сроки (этапы) проведения работ. Ресурсы включают персонал, технические средства, программное обеспечение и финансирование.

После завершения пилотного проекта его результаты необходимо оценить и сопоставить их с изначальными потребностями организации, критериями успешного внедрения CASE-средств, базовыми метриками и критериями успеха пилотного проекта.

В процессе оценки организация должна определить свою позицию по следующим трем вопросам:

1. Целесообразно ли внедрять CASE-средство?
2. Какие конкретные особенности пилотного проекта привели к его успеху (или неудаче)?
3. Какие проекты или подразделения в организации могли бы получить выгоду от использования средств?

Возможны четыре категории результатов и соответствующих действий:

- пилотный проект потерпел неудачу, и его анализ показал неадекватность ожиданий организации (изменение ожиданий и пересмотр результатов);
- пилотный проект потерпел неудачу, и его анализ показал, что выбранные средства не удовлетворяют потребности организации (пересмотреть подход к выбору CASE-средств);
- пилотный проект потерпел неудачу, и его анализ показал наличие таких проблем, как неудачный выбор пилотного проекта, неадекватное обучение и недостаток ресурсов (пересмотр процесса внедрения с возможностью начать новый пилотный проект);
- пилотный проект завершился успешно, и признано целесообразным внедрять CASE-средства в некоторых подразделениях или, возможно, во всей организации в целом (определение подходящего масштаба внедрения).

Возможным решением должно быть одно из следующих:

- внедрить средство. В этом случае рекомендуемый масштаб внедрения должен быть определен в терминах структурных подразделений и предметной области;
- выполнить дополнительный пилотный проект (если остались конкретные неразрешенные вопросы относительно внедрения);
- отказаться от средства (причины отказа от конкретного средства должны быть определены в

терминах потребностей организации или критериев, которые остались неудовлетворенными);

- отказаться от использования CASE-средств вообще (организация либо не готова к внедрению CASE-средств, либо автоматизация данного аспекта процесса создания и сопровождения программного обеспечения не дает никакого эффекта для организации).

Результатом пилотного проекта является документ, в котором обсуждаются его результаты и детализируются решения по внедрению.

Процесс перехода к практическому использованию CASE-средств начинается с разработки и последующей реализации плана перехода.

План перехода должен включать следующее:

- информацию относительно целей, критериев оценки, графика и возможных рисков, связанных с реализацией плана;
- информацию относительно приобретения, установки и настройки CASE-средств;
- информацию относительно интеграции каждого средства с существующими, включая как интеграцию CASE-средств друг с другом, так и их интеграцию в процессы разработки и эксплуатации программного обеспечения, существующие в организации;
- ожидаемые потребности в обучении и ресурсы, используемые в течение и после завершения процесса перехода;
- определение стандартных процедур использования средств.

Реализация плана перехода требует постоянного мониторинга использования CASE-средств,

обеспечения текущей поддержки, сопровождения и обновления средств по мере необходимости.

Для доказательства эффективности CASE-средств и их возможностей улучшать продуктивность необходимы такие базовые метрические данные, как:

- использованное время,
- время, выделенное персонально для конкретных специалистов,
- размер, сложность и качество ПО,
- удобство сопровождения.

Метрическая оценка должна начинаться с реальной оценки текущего состояния среды еще до начала внедрения CASE-средств и поддерживать процедуры постоянного накопления данных.

Результатом данного этапа является внедрение CASE-средств в повседневную практику организации, при этом больше не требуется какого-либо специального планирования. Кроме того, поддержка CASE-средств включается в план текущей поддержки ПО в данной организации.

## 5.4 Примеры существующих CASE-средств

В качестве примеров, можно выделить следующие популярные CASE-средства:

1. CA ERwin Process Modeler
2. CA ERwin Data Modeler
3. Visual Paradigm for UML
4. ARIS Express

CA ERwin Process Modeler (ранее BPwin) является инструментом позволяющим моделировать, анализировать, документировать и оптимизировать бизнес-процессы. Данный продукт поддерживает

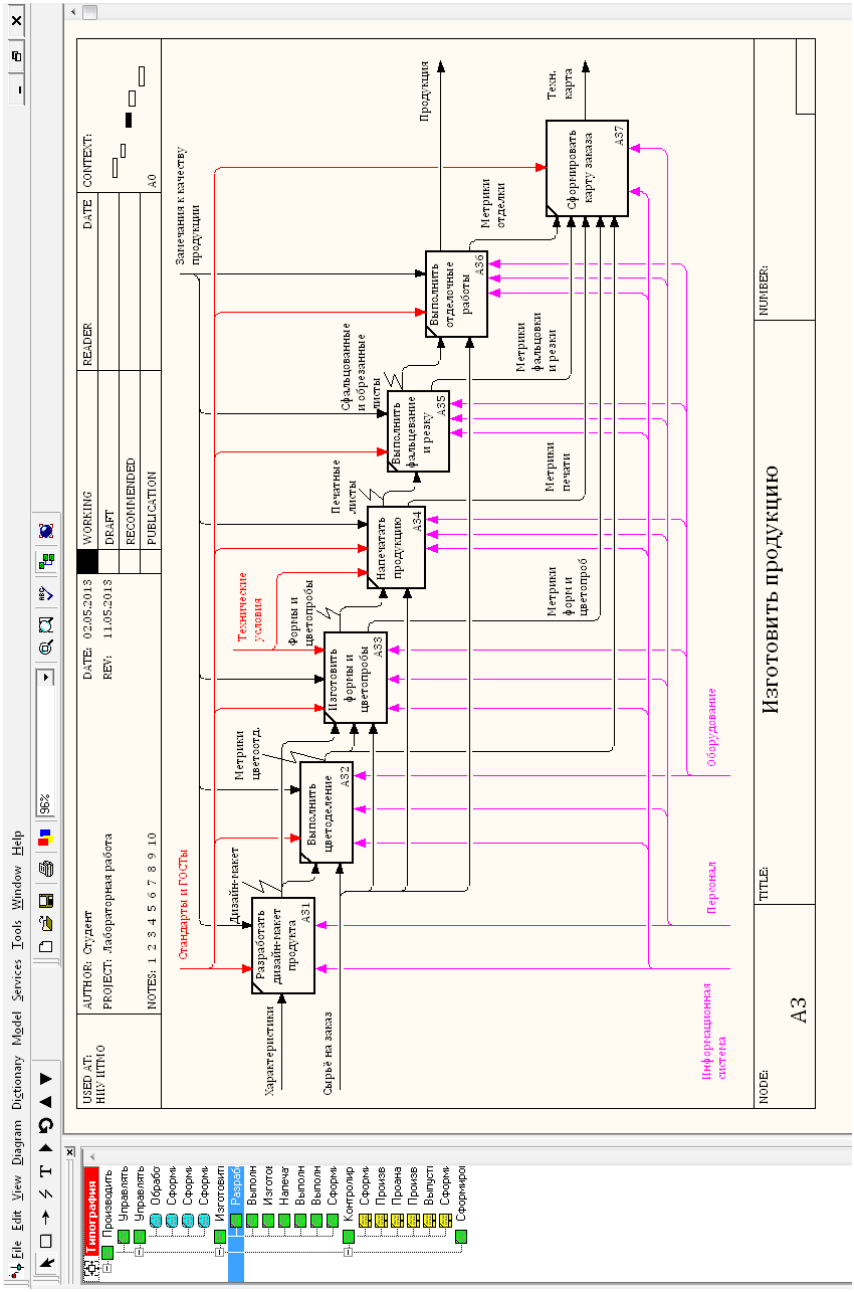



Рисунок 31- Интерфейс CA ERwin Process Modeler

такие нотации как: IDEF-0, IDEF0, IDEF3, DFD, FEO, Swimlane. Интерфейс программы показан на рис. 31.

CA ERwin Data Modeling представляет собой среду моделирования данных.

CA ERrwin Data Modeler позволяет проектировать структуру баз данных в нотациях IDEF1x, IE и Dimensional, генерировать SQL-код разработанной базы данных, осуществлять прямое и обратное проектирование, составлять различные отчёты.

Интерфейс программы представлен на рис. 32

Visual Paradigm for UML относится к профессиональным инструмента работы со стандартом UML. При помощи встроенного функционала данный пакет способен поддерживать весь рабочий цикл программы: анализ, ориентированный на объекты, дизайн, ориентированный на объекты, конструкция, тестирование и разработка.

Visual Paradigm for UML позволяет:

- создавать UML диаграммы;
- создавать SysML диаграммы;
- создавать SoaML диаграммы;
- проектировать корпоративные архитектуры, используя фреймворки;
- проектировать структуру баз данных и генерировать SQL-код;
- осуществлять прямое и обратное проектирование на множество языков программирования;
- создание диаграмм бизнес-процессов BPMN;
- создавать модели BPEL;
- создавать базовые диаграммы.

Интерфейс программ показан на рис. 33.

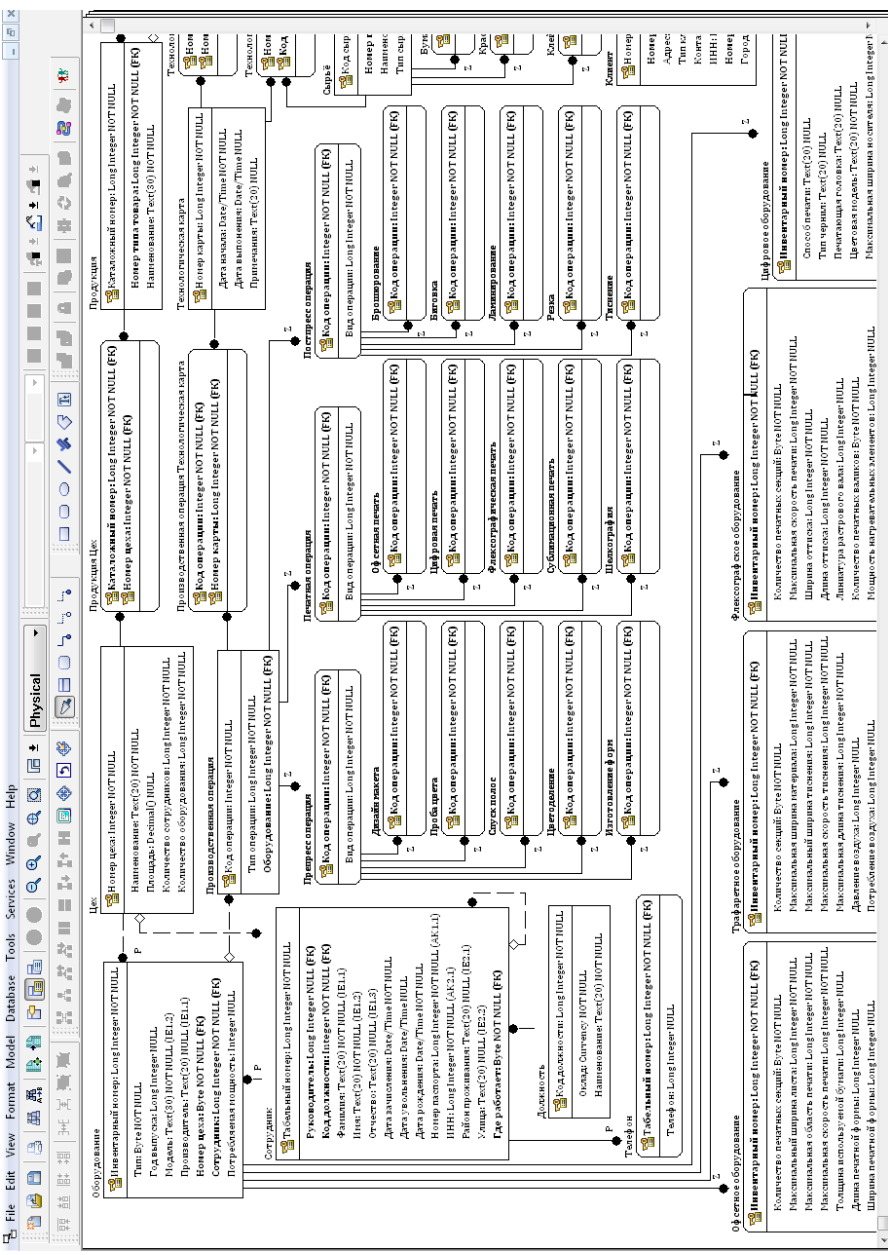


Рисунок 32 - Интерфейс CA ERwin Data Modeler

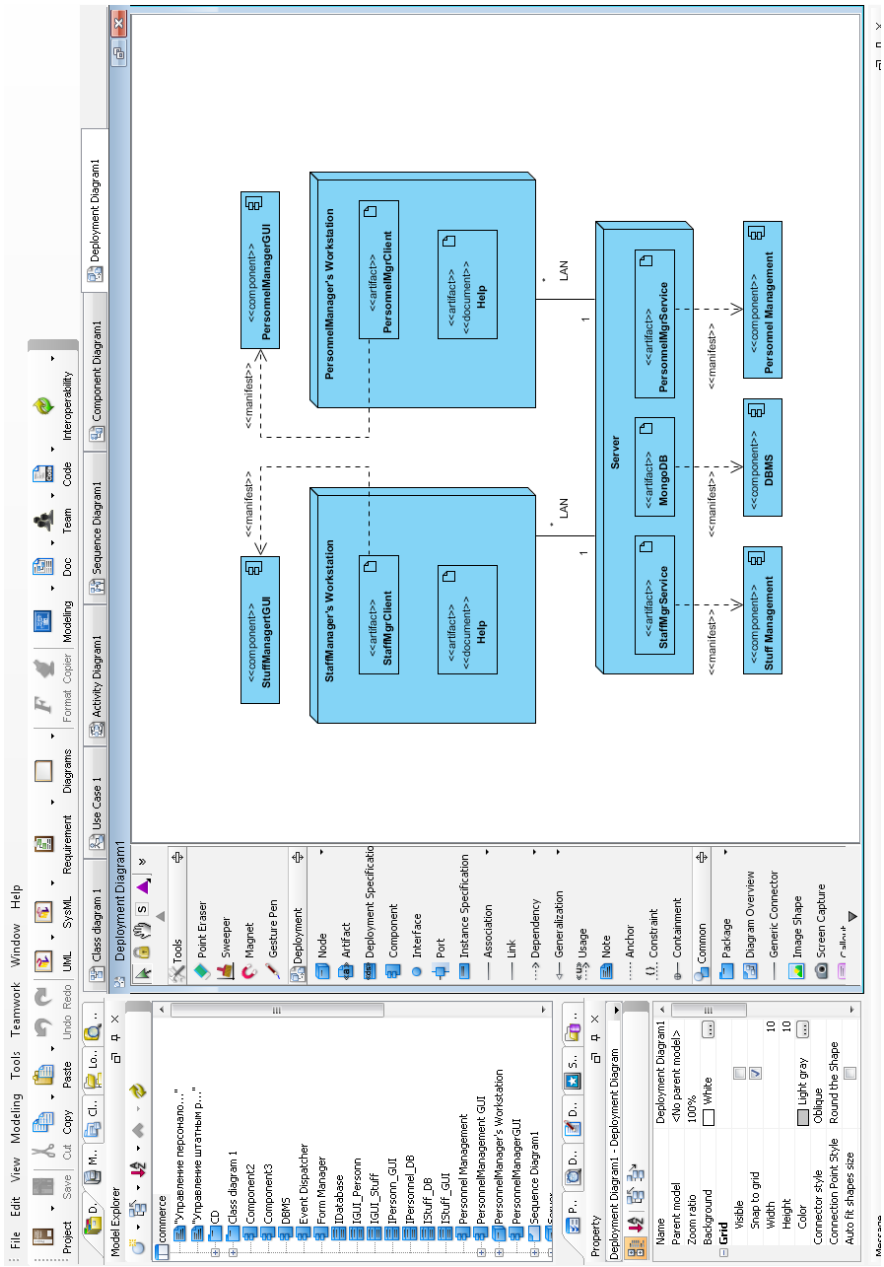



Рисунок 33 - Интерфейс программы Visual Paradigm for UML

ARIS Express принадлежит к семейству средств моделирования ARIS (ARchitecture of Integrated Information Systems) компании IDS Scheer, которая является частью компании Software AG. Ведущие аналитические компании Gartner Group и Forrester Research относят компанию IDS Scheer к лидерам мирового рынка средств моделирования и анализа бизнес-процессов.

ARIS Express поддерживает следующие типы моделей:

- Организационная диаграмма (Organizational chart)
- Бизнес-процесс (Business process)
- ИТ-инфраструктура (IT infrastructure)
- Карта процессов (Process landscape)
- Модель данных (Data model)
- Карта систем (System landscape)
- Доска (Whiteboard)
- BPMN диаграмма версии 2.0 (BPMN diagram)
- Общие диаграммы (General diagram)

Интерфейс программы показан на рис. 34.

В настоящее время при проектировании информационных систем широко применяюся как отечественные CASE-средства, так и продукты иностранных разработчиков. Почти каждый год появляются или новые версии известных программ, или новые CASE-средства, учитывающие пролемы предшественников.

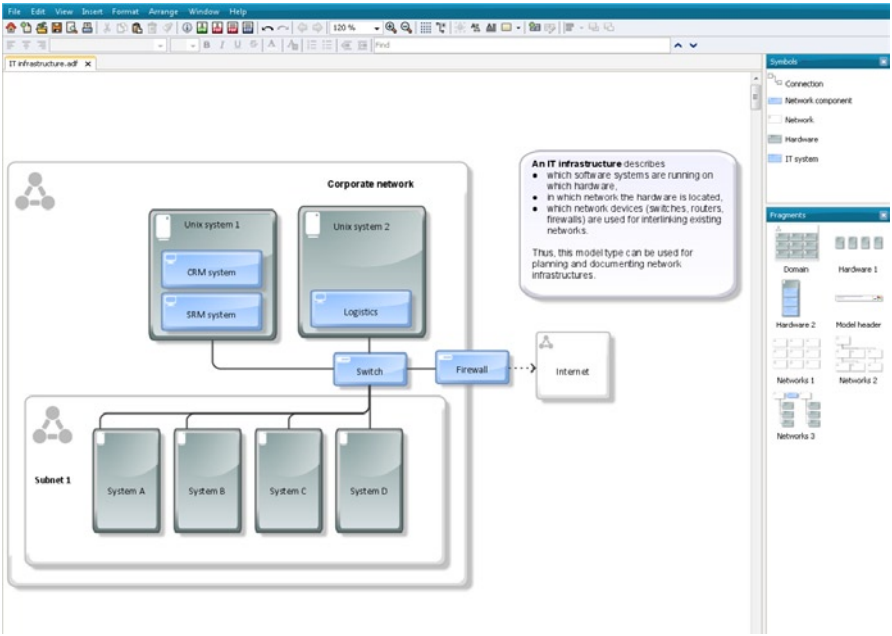



Рисунок 34 - Интерфейс ARIS Express

# 6. Проектирование информационных систем на основе унифицированного языка моделирования UML

## 6.1 Основы унифицированного языка моделирования UML

Существует большое количество инструментальных средств, используемых для реализации проекта ИС от этапа анализа до создания программного кода. Отдельно выделяют так называемые CASE-средства верхнего уровня (upper CASE tools) и CASE-средства нижнего уровня (lower CASE tools).

Среди основных проблем использования CASE-средств верхнего уровня выделяют проблемы их адаптации под конкретные проекты, так как они жестко регламентируют процесс разработки и не дают возможности организовать работу на уровне отдельных элементов проекта. Альтернативой им может стать использование CASE-средства нижнего уровня, но их использование влечет другие проблемы – трудности в организации взаимодействия между командами, работающими над различными элементами проекта.

Средством, позволяющим объединить эти подходы, явился унифицированный язык объектно-ориентированного моделирования (Unified Modeling Language – UML). К преимуществам языка UML можно отнести разнообразные инструментальные средства, которые как поддерживают жизненный цикл ИС, так и позволяют настроить и отразить специфику деятельности разработчиков различных элементов проекта.

В конце 1980-х годов получили большое распространение объектно-ориентированные языки программирования. Тенденции их активного использования определили задачи разработки языка моделирования, дающего возможность реализовать объектно-ориентированный подход и построить наилучшую модель системы с указанием ее значимых свойств. Этим языком стал UML. В настоящее время UML как нотация моделирования ИС поддерживается рядом объектно-ориентированных CASE-продуктов.

Основными характеристиками объектно-ориентированного языка моделирования UML являются:

- организация взаимодействия заказчика и разработчика (групп разработчиков) ИС путем построения репрезентативных визуальных моделей;
- специализация базовых обозначений для конкретной предметной области.

Базовый набор диаграмм UML содержится в большом количестве средств моделирования. Однако в связи с тем, что каждая прикладная задача имеет свои особенности и не требует всех концепций в каждом приложении, язык предоставляет пользователям такие возможности, как:

- моделирование с использованием только средств «ядра» для типовых приложений;
- моделирование с использованием дополнительных условных обозначений, если они отсутствуют в «ядре», или специализация нотации и ограничений для данной предметной области.

Для поддержки моделирования различных этапов жизненного цикла ИС язык UML предлагает целую совокупность диаграмм.

При разработке **концептуальной модели** применяют диаграммы вариантов использования и диаграммы деятельности, модели бизнес-объектов, диаграммы последовательностей.

На этапе работы над **логической моделью** ИС описать требования к системе позволяют диаграммы вариантов использования, а при предварительном проектировании используют диаграммы классов, диаграммы состояний, диаграммы последовательностей.

Детальное проектирование при разработке **физической модели** выполняют с применением диаграмм классов, диаграмм развертывания, диаграмм компонентов.

Далее будут подробнее рассмотрены перечисленные диаграммы с указанием их назначения в процессе проектирования ИС.

## 6.2 Проектирование логической модели ИС и моделей баз данных

Одной из диаграмм, применяющихся на этапе проектирования логической модели ИС, является **диаграмма вариантов использования** (диаграмма прецедентов, use case diagram), предназначенная для построения на концептуальном уровне модели того, как функционирует система в окружающей среде.

Основными элементами для построения модели прецедентов на диаграмме являются:

- актёр (actor) – элемент, обозначающий роли пользователя, который взаимодействует с определенной сущностью;
- прецедент – элемент, отражающий действия, выполняемые системой (в т.ч., с указанием возможных вариантов), которые приводят к результатам, наблюдаемым актерами.

Между прецедентами в модели могут быть установлены связи, такие, как:

- обобщение (Generalization) – указывает общность ролей;
- включение (include) – указывает взаимосвязь нескольких вариантов использования, базовый из которых всегда использует функциональное поведение связанных с ним прецедентов;
- расширение (extend) – указывает взаимосвязь базового варианта использования и вариантов использования, являющихся его специальными случаями.

Пример диаграммы вариантов использования представлен на рис. 35:

Более детально описать бизнес-процессы позволяют диаграммы деятельности и диаграммы последовательностей.

**Диаграмма деятельности** (activity diagram) — диаграмма, использующаяся при моделировании бизнес-процессов, на которой представлено разложение на составные части некоторой деятельности, а именно: скоординированного выполнения отдельных действий и вложенных видов

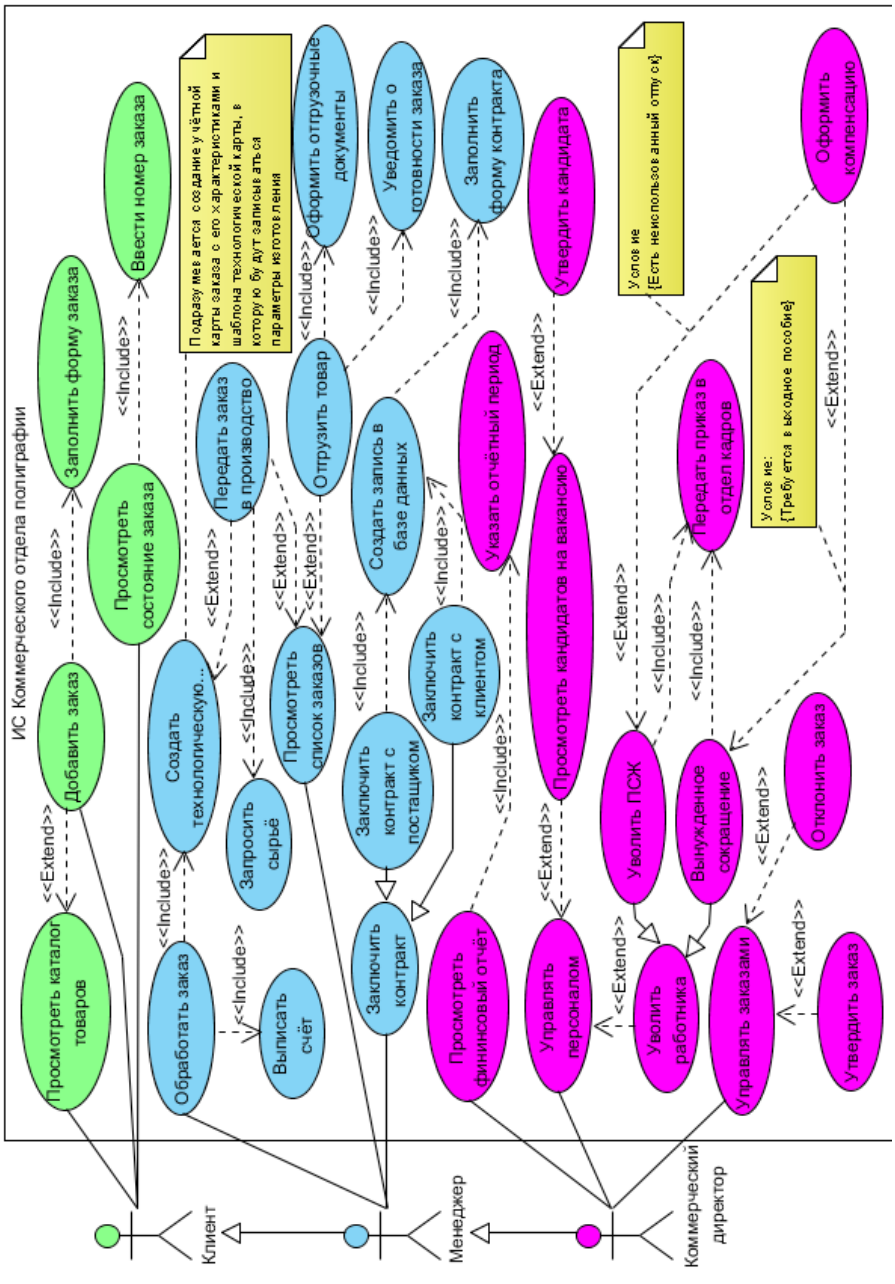



Рисунок 35 - Диаграмма вариантов использования

деятельности, которые соединяются между собой потоками от выходов одного узла к входам другого, с указанием их исполнителей.

Пример диаграммы деятельности представлен на рис. 36:

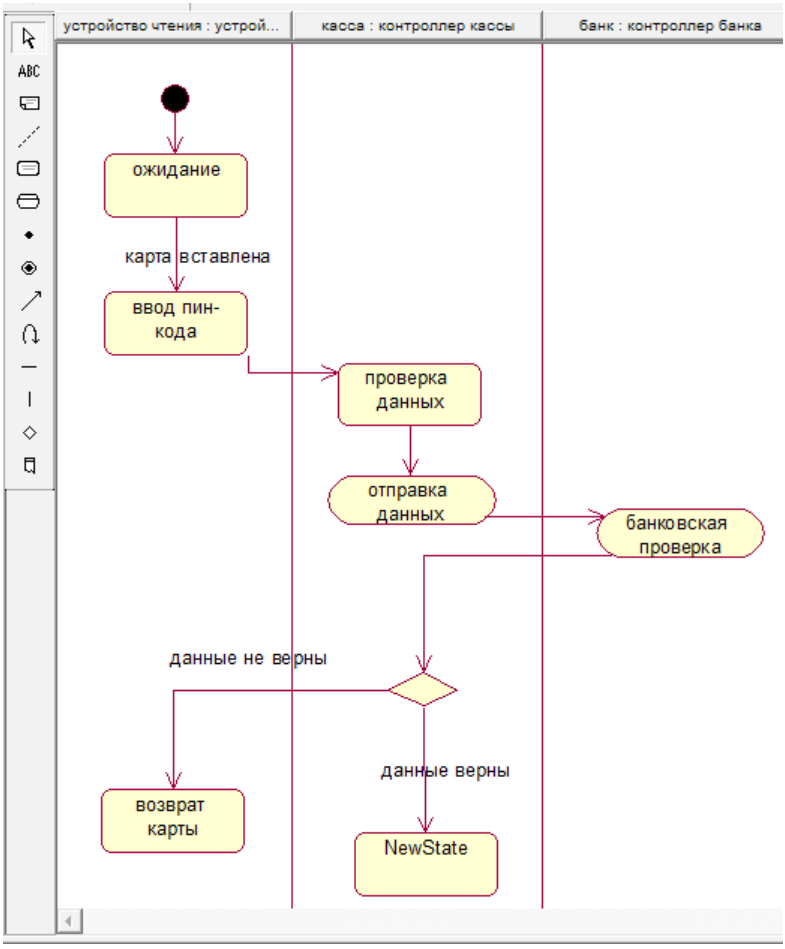



Рисунок 36 - Диаграмма деятельности

Разработанные на этапе построения моделей бизнес-прецедентов диаграммы видов деятельности могут корректироваться вследствие выявления новых подробностей в описании бизнес-процессов объекта автоматизации на этапах анализа и проектирования.

**Диаграмма последовательности** (sequence diagram) — диаграмма, отражающая упорядоченные по времени проявления взаимодействия объектов.

На диаграмме данного типа слева направо помещаются основные элементы: объекты; вертикальные линии (lifeline), моделирующие течение времени при выполнении действий объектом; стрелки, определяющие действия, выполняемые объектом.

Пример диаграммы последовательности представлен на рис. 37:

В результате построения диаграмм на этом этапе разработаны подробные описания действий специалистов по внедрению ИС, которые необходимы для обеспечения ее функциональности.

На этапе формирования требований разрабатывается модель системных прецедентов, на которой для внутренних и внешних исполнителей указываются их конкретные обязанности с использованием ИС. Модели системных прецедентов разрабатываются на основе бизнес-моделей, построенных на предыдущем этапе, однако при этом необходимо детально описать прецеденты с определениями используемых данных и указанием последовательности их выполнения, то есть подробно описать реализацию функций проектируемой системы.

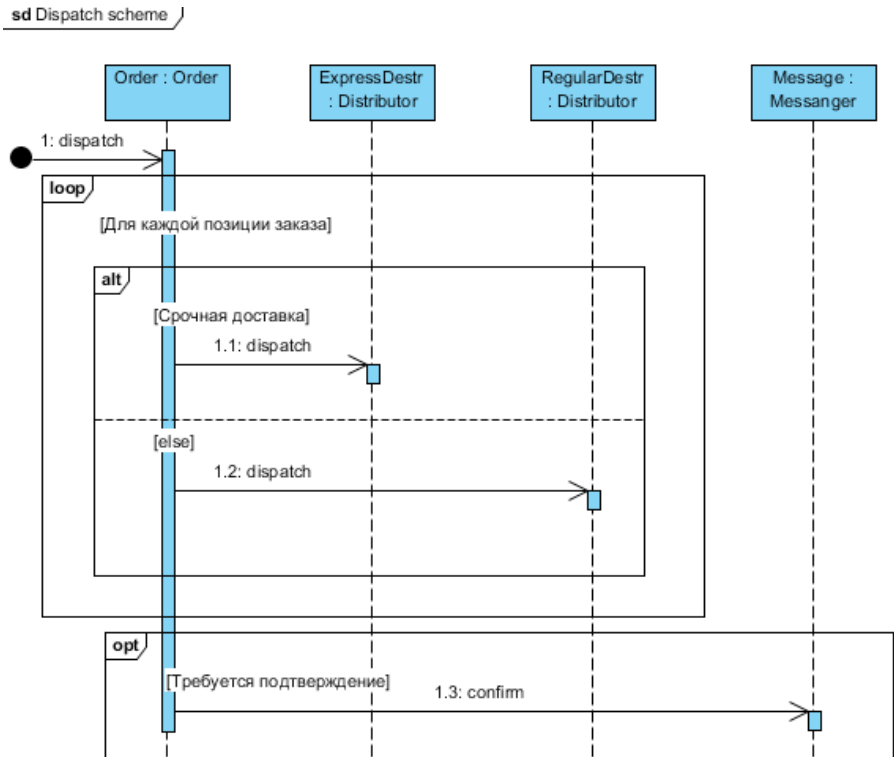



Рисунок 37 - Диаграмма последовательности

На этапе анализа требований и предварительного проектирования системы на основе построенных моделей системных прецедентов строятся диаграммы классов системы.

Диаграмма классов, являясь логическим представлением модели, представляет детальную информацию о структуре модели системы с использованием терминологии классов объектно-ориентированного программирования, а именно: о

внутреннем устройстве системы (об архитектуре системы). На диаграмме классов могут быть указаны внутренняя структура и типы отношений между отдельными объектами и подсистемами, что приводит к развитию концептуальной модели системы.

Класс в языке UML обозначает некоторое множество объектов, обладающих одинаковой структурой и взаимосвязями с объектами других классов. В диаграммах классов системы указываются объекты из модели системных прецедентов с их описанием и указанием взаимосвязей между классами.

Синтаксис диаграмм классов является эффективным средством структурирования требований к элементам проектируемой системы, к их данным, интерфейсам, функциональности.

Пример диаграммы классов представлен на рис. 38:

На этом этапе проектирования подробно описаны состав и функции системы в соответствии с разработанными бизнес-моделями, что дает уверенность в соответствии проектируемой системы требованиям заказчика.

На следующем этапе элементы разработанных моделей классов отображаются в элементы моделей базы данных и приложений, а именно:

- классы – в таблицы;
- атрибуты – в столбцы;
- типы – в типы данных СУБД;
- ассоциации – в связи между таблицами (в том числе, создавая дополнительные таблицы связей);

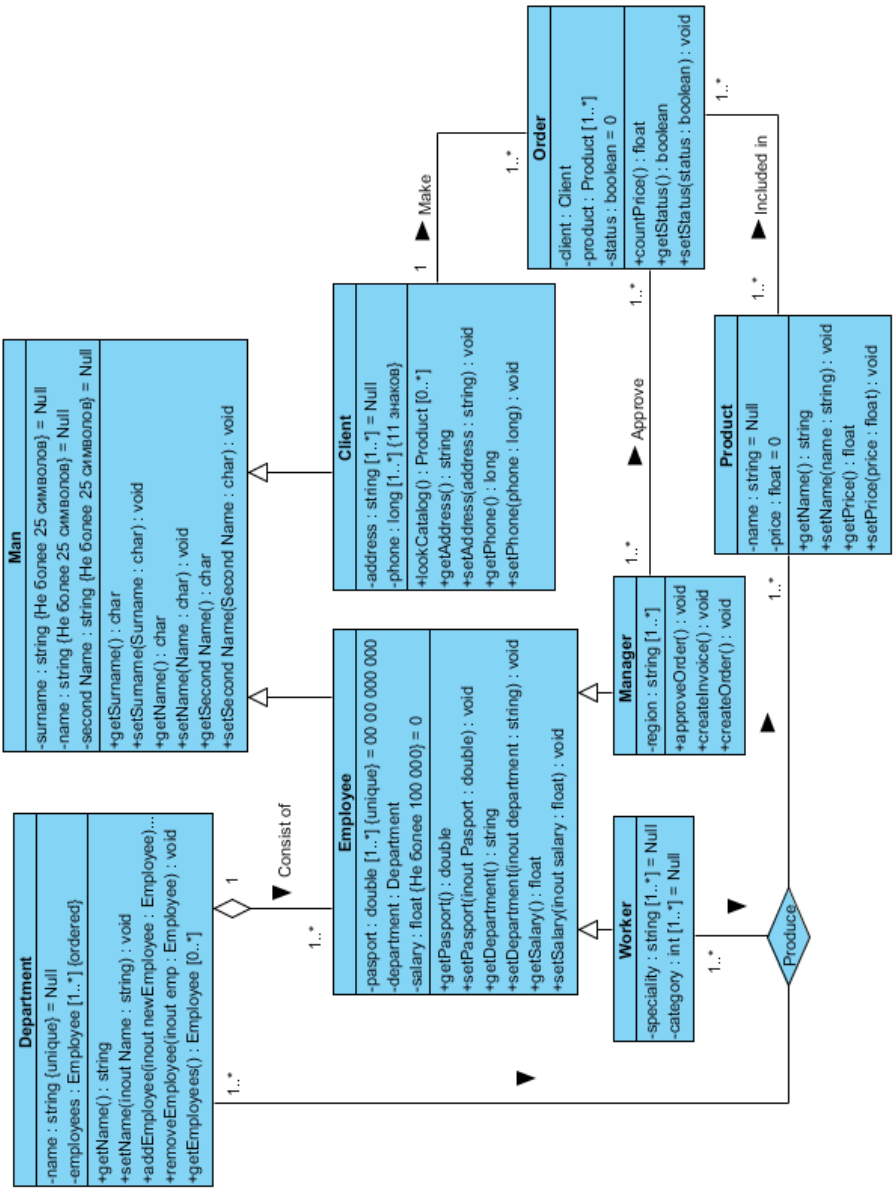



Рисунок 38 - Диаграмма классов

- приложения – в классы с определенными методами и атрибутами, связанными с данными в базе.

В модели базы данных для каждого простого класса формируется таблица, которая включает столбцы, поставленные в соответствие атрибутам класса.

Классы подтипов могут отображаться в таблицы несколькими способами:

- в случае отображения одной таблицы на класс отдельная таблица формируется для каждого класса с установлением последующих связей;
- в случае отображения одной таблицы на суперкласс для суперкласса создается таблица, а затем в таблицы подклассов размещаются столбцы для атрибутов суперкласса;
- в случае отображения одной таблицы на иерархию формируется единая таблица, в которой расположены атрибуты суперкласса и всех подклассов с добавлением дополнительных столбцов для определения исходных таблиц подклассов.

Язык UML для разработки проекта БД предлагает специальный профиль Profile for Database Design, в котором используются основные компоненты диаграмм, такие, как: таблица, столбец, ключи, связи, домен, процедура и тд. На диаграммах также можно указывать дополнительные характеристики столбцов и таблиц: ограничения, триггеры, типы данных и тд. В результате этого этапа проект базы данных и приложений системы становится детально описанным.

## 6.3 Проектирование физической модели ИС

Следующий этап проектирования системы включает в себя дополнения модели баз данных и приложений диаграммами их размещения на технических средствах.

На данном этапе рассматриваются такие понятия UML, как:

- компонент – элемент физического представления системы, реализующий определенный набор интерфейсов;
- зависимость – связь между двумя элементами, обозначающая ситуацию, при которой изменение одного элемента модели влечет за собой изменение ее другого элемента;
- процессор и устройство – узел, выполняющий и не выполняющий обработку данных соответственно;
- соединение – связь между процессорами и устройствами.

Наиболее полно особенности физического представления системы на языке UML позволяют диаграммы компонентов и диаграммы развертывания.

Диаграмма компонентов (component diagram) отображает иерархию подсистем, структурных компонентов и зависимостей между ними. Физическими компонентами выступают базы данных, исполняемые файлы, приложения, библиотеки, интерфейсы ИС и т.д. В случае использования диаграммы компонентов для отображения внутренней структуры компонентов,

интерфейсы составного компонента делегируются в определенные интерфейсы внутренних компонентов.

Основными целями построения диаграмм компонентов являются:

- определение архитектуры проектируемой системы;
- построение концептуальной и физической моделей баз данных;
- представление структуры исходного и специфики исполняемого кода системы;
- многократное использование определенных фрагментов программного кода.

Пример диаграммы компонентов представлен на рис. 39:

Для описания аппаратной конфигурации ИС применяют диаграмму развертывания (deployment diagram).

С помощью диаграммы развертывания моделируют физическое распределение различных программных, информационных, аппаратных компонентов системы по комплексу технических средств. Особое внимание на диаграмме развертывания уделяется отображению того, какие используются аппаратные компоненты («узлы» - серверы баз данных и приложений, веб-верверы), какие программные компоненты («артефакты» - базы данных, веб-приложения) работают на каждом из них и как части комплекса соединены друг с другом.

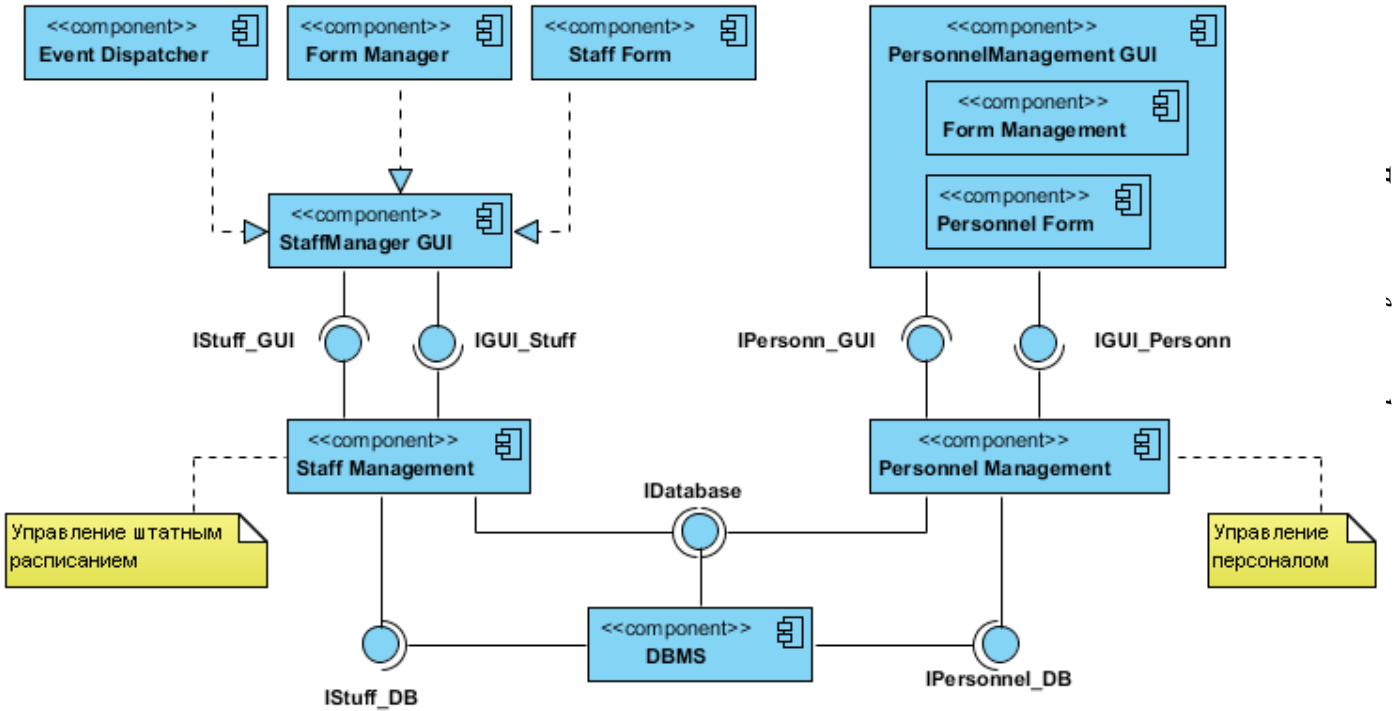



Рисунок 39 – Диаграмма компонентов

Пример диаграммы развертывания представлен на рисунке 40:

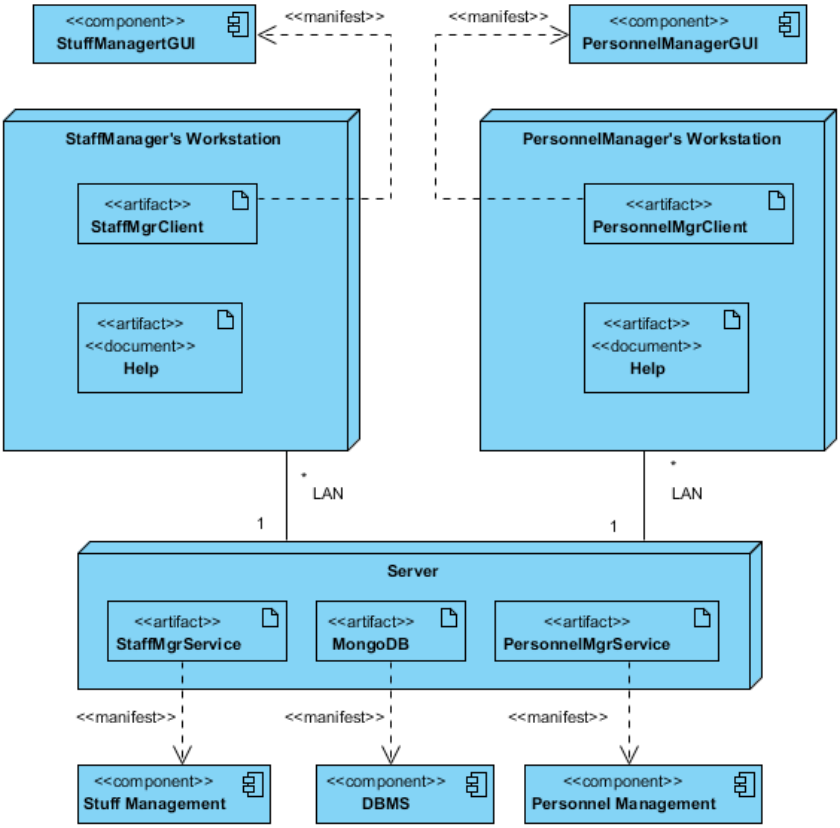



Рисунок 40 – Диаграмма развертывания

Таким образом, в случае проектирования сложной ИС ее необходимо разделить на части и исследовать каждую часть отдельно. Существуют

два основных способа разбиения ИС на такие подсистемы:

- структурная (функциональная) декомпозиция;
- объектная (компонентная) декомпозиция.

Характерной особенностью объектной декомпозиции является выделением объектов (компонентов), взаимодействующих между собой, выполняющих определенные функции (методы) объекта.

При проектировании ИС язык UML используют для визуального моделирования системы при ее разбиении на объекты. В случае функциональной декомпозиции ИС при проектировании использование UML нецелесообразно, здесь применяются другие структурные нотации, рассмотренные в предыдущих разделах.

## Литература

1. Федеральный закон Российской Федерации от 27 июля 2006 г. N 149-ФЗ. Об информации, информационных технологиях и о защите информации.
2. ГОСТ 24.202-80 . Требования к содержанию документа «Технико-экономическое обоснование создания АСУ.
3. ГОСТ 34.201-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем.
4. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированных систем.
5. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения.
6. ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадия создания.
7. ГОСТ 34.603-92. Виды испытаний автоматизированных систем.
8. ГОСТ Р ИСОМЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.
9. ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств.

10. ГОСТРВ51987-2002.Информационнаятехнология. Комплекс стандартов на автоматизированные системы. Требования и показатели качества функционирования информационных систем (ИС). Общие положения.
11. ГОСТ Р ИСО 15288-2005 Информационная технология. Системная инженерия. Процессы жизненного цикла систем.
12. Анисимов В.В. Проектирование информационных систем. Электронный ресурс: https://sites.google.com/site/anisimovkhv/learning/pris.
13. Буч Г., Якобсон А., Рамбо Дж., UML. Классика CS. Издание второе, - СПб.: Питер, 2006. - 736 с.
14. Галямина И.Г., Управление процессами, - СПб.: Питер, 2013. - 304 с.
15. Грекул В. И., Денищенко Г. Н., Коровкина Н. Л.— Проектирование информационных систем: учебное пособие / 2-е изд., испр. — М.: Интернет-Университет информационных технологий (ИНТУИТ.РУ): БИНОМ. Лаборатория знаний, 2010 .С 299.
16. Дейт К. Дж., Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом “Вильяме”, 2005. — 1328 с.: ил. — Парал. тит. англ.
17. Диго С.М., Базы данных. Проектирование и создание: Учебно-методический комплекс. – М.: Изд. центр ЕАОИ. 2008. – 171 с.
18. Дубейковский В.И., Эффективное моделирование с CA ERwin Process Modeler (BPwin; AllFusion Process Modeler), - М.: Диалог-МИФИ, 2009, - 384 с.
19. Кириллов, В. В., Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов. — СПб.: БХВ-Петербург, 2009. — 464 с.

20. Ларман К., Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку, - М.: Вильямс, 2013. - 736 с.
21. Леоненков А.В., Самоучитель UML 2, - СПб.: БХВ-Петербург, 2007. - 576 с.
22. Маклаков С.В., Туманов В.Е., Проектирование реляционных хранилищ данных, - М.: Диалог-МИФИ, 2007, - 336 с.
23. Новиков Ф.А., Иванов Д.Ю., Моделирование на UML. Теория, практика, видеокурс, - СПб.: Профессиональная литература, 2010. - 640 с.
24. Новиков Ф.А., Иванов Д.Ю., Моделирование на UML [Электронный ресурс]: Интернет книга. - Электронные данные. - 2013. - Режим доступа: http://book.uml3.ru.
25. Рамбо Дж., Блаха М., UML 2.0. Объектно-ориентированное моделирование и разработка, - СПб.: Питер, 2007. - 544 с.
26. Репин В.В., Елиферов В.Г., Процессный к управления. Моделирование бизнес-процессов, - М.: Манн, Иванов и Фербер, 2013. - 544 с.
27. Роберт Дж. Мюллер, Проектирование баз данных и UML, - М.: Лори, 2013, - 432 с.
28. Советов Б. Я., Базы данных: теория и практика : Учебник для бакалавров / 2-е изд., - М.: Юрайт, 2012. - 464 с.
29. Инфопортал: www.finexpert.ru
30. Интернет-сайт www.idef.ru

УНИВЕРСИТЕТ ИТМО

**Миссия университета** – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

---

# КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ В ГУМАНИТАРНОЙ СФЕРЕ

Кафедра интеллектуальных технологий в гуманитарной сфере была организована на естественнонаучном факультете в 1998 году и при образовании получила название «кафедра технологий профессионального обучения». С 2002 года кафедра стала выпускающей. Её выпускники до 2008 года получали специальность «Профессиональное обучение. Компьютерные технологии». С 2004 года началась подготовка инженеров по специальности «Информационные технологии в образовании», а с 2011 – бакалавров по направлениям «Информационные системы и технологии» и «Интеллектуальные системы в гуманитарной сфере». В 2012 году кафедра была переименована в соответствии с основным направлением деятельности.

Центральной идеей образовательных программ, реализуемых кафедрой, является участие студентов в выполнении работ, связанных с возможными направлениями будущей деятельности, и с задачами, решаемыми университетом. Научные исследования, проводимые на кафедре, связаны с интеллектуальным анализом данных, математическим моделированием и проектированием информационных систем. В этих областях много интересных, сложных и нерешённых задач. На старших курсах студенты имеют возможность выбрать то направление исследования в рамках профиля, которое им наиболее интересно. После успешного окончания бакалавриата выпускники поступают в магистратуру по направлению «Информационные системы и технологии», кафедра ведет подготовку с 2009 года. Двухлетнее образование в магистратуре позволяет развить компетенции в следующих видах профессиональной деятельности: научно-исследовательской; проектно-аналитической; организационно-управленческой; научно-педагоги-ческой и инновационной. Всё это возможно благодаря разносторонним научным интересам преподавателей кафедры, тесным связям с выпускниками.

http://itgs.ifmo.ru/

**Авторы**

Коцюба Игорь Юрьевич

Чунаев Антон Владимирович

Шиков Алексей Николаевич

# Основы проектирования информационных систем

**Учебное пособие**

В авторской редакции

Редакционно-издательский отдел университета ИТМО

Зав. РИО Н.Ф. Гусарова

Подписано к печати

Заказ №

Тираж 100 экз.

Отпечатано на ризографе.

**Редакционно-издательский отдел**
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49