

Проектирование информационных систем

Конспект лекций

Версия 0.3.3

Санкт Петербург

2008 г.

Содержание

<i>Введение</i>	<i>4</i>
1.1 Цели обучения	4
1.2 Рекомендуемая литература	4
1.3 Структура конспекта	5
2. <i>Введение в проектирование, основные понятия проектирования</i>	<i>6</i>
3. <i>Понятие системы. Понятие системного подхода и системного анализа</i>	<i>21</i>
4. <i>Документ. Электронный документ. Информационная система. Информационная технология.</i>	<i>40</i>
5. <i>Комплексная архитектура предприятия</i>	<i>51</i>
5.1 Концепция устройства комплексной архитектуры предприятия	55
5.2 Основные понятия бизнес – модели предприятия	59
6. <i>Моделирование информационных систем</i>	<i>62</i>
6.1 Общие положения	62
6.2 Методы структурного моделирования	66
6.3 Методы объектно- ориентированного моделирования	67
7. <i>Архитектура информационной системы</i>	<i>69</i>
8. <i>Модели жизненного цикла информационных систем</i>	<i>80</i>
8.1 Каскадная модель	81
8.2 Инкрементная модель	83
8.3 Эволюционная модель	84
9. <i>Ключевые концепции унифицированного процесса</i>	<i>87</i>
9.1 Унифицированный процесс – управляемый вариантами использования	88
9.2 Унифицированный процесс - ориентирован на архитектуру	90
9.3 Унифицированный процесс - итеративный и инкрементный	91
9.4 Жизненный цикл в унифицированном процессе	93
9.5 Продукт унифицированного процесса	94
9.6 Унифицированный процесс – методология разработки	95
10. <i>Анализ. Концептуальная (аналитическая) модель</i>	<i>96</i>
10.1 Граничные классы	101
10.2 Классы сущностей	102
10.3 Управляющие классы	103
11. <i>Проектирование. Модель проектирования (логическая модель)</i>	<i>108</i>
11.1 Подходы к разработке модели проектирования	111
11.2 Определение операций класса проектирования	114
11.3 Шаблоны проектирования	116

11.3.1	Шаблон MVC (Model-View-Controller)	117
11.3.2	Шаблон Expert	120
11.3.3	Шаблон Controller	121
11.3.4	Шаблон Polymorphism	123
11.4	Определение атрибутов класса проектирования	124
11.5	Определение ассоциаций и агрегаций класса проектирования	124
11.6	Определение обобщений класса проектирования	126
11.7	Определение методов класса проектирования	126
12.	Экстремальные методологии	127
13.	Перечень использованных источников	129
14.	Приложения	131
14.1	Приложение 1. Пример текстового описания варианта использования	131
14.2	Приложение 2. Пример описания предметной области (по ГОСТ 34.320-96)	134
14.2.1	Правила и требования	134
14.2.2	Некоторые факты и события в пространстве сущностей	136
14.3	Приложение 3. Концептуальная схема для подхода сущность-атрибут-связь (необходимые высказывания по ГОСТ 34.320-96)	138
14.4	Приложение 4. Содержание отчета по лабораторной работе	142

Введение

1.1 Цели обучения

Целью дисциплины проектирование информационных систем является овладение студентами теоретическими знаниями и практическими навыками в области проектирования современных информационных систем, используемых для решения проблем (задач), в различных областях деятельности предприятий.

В процессе достижения цели решаются следующие задачи:

1. Изучение теоретических основ проектирования, основных понятий логики, системного анализа, информационных систем
2. Изучение комплексной архитектуры предприятия и архитектуры информационной системы
3. Овладение ключевыми процедурами методологии проектирования, основанными на методах анализа и синтеза (аналитического и проектного моделирования) и поддерживаемых CASE-средствами
4. Выполнение лабораторной работы по проектированию информационной системы посредством объектно-ориентированного подхода к проектированию с использованием CASE- средства. Описание разделов отчета и краткое их содержание дано в Приложение 4. Содержание отчета по лабораторной работе. Кроме того, к файлу конспекта прилагается электронный учебник по UML и файл-шаблон отчета по лабораторной работе.

1.2 Рекомендуемая литература

1. А. Якобсон, Г.Буч, Дж. Рамбо
Унифицированный процесс разработки программного обеспечения
Изд-во «Питер», 2002
2. Крэг Ларман
Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование
Изд-во «Вильямс», 2001
3. Государственные стандарты серии «Информационная технология»
(Перечень использованных источников)

1.3 Структура конспекта

Содержательный материал курса сгруппирован в 12 разделов, каждый из которых раскрывает определенную тему, необходимую для понимания процесса проектирования и выполнения лабораторной работы.

В разделе 2 вводятся основные понятия проектирования, рассматриваются основные понятия формальной логики (понятие, предмет, свойство, отношение, объект), приводятся основные положения диалектической логики и содержательно-генетической логики. В качестве основных понятий проектирования рассматриваются логическая схема проектирования и задача проектирования, формулируется возможность представления процесса проектирования как процесса решающего задачи проектирования.

В разделе 3 вводятся основные понятия системного подхода и системного анализа. Введено понятие системы (семантическое и формальное) По текстам С. П. Никанорова системный анализ рассматривается как методология решения крупных проблем. С этой точки зрения рассмотрена и организация как основной объект автоматизации (информатизации).

В разделе 4 введено понятие документа, электронного документа и на его основе дано понятие информационной системы.

В разделе 5 рассмотрена комплексная архитектура предприятия, которая дается в форме «плоской» таблицы, основной для которой явилась схема Захмана. Описание комплексной архитектуры в конспекте недостаточно подробно, но это не является препятствием к его пониманию, так как на лекционных занятиях он обсуждается достаточно подробно.

В разделах 6-11 рассмотрены вопросы моделирования, архитектуры информационной системы и методологии создания информационных систем. Знания этих разделов будут полезны для выполнения практического задания.

В разделе приложений (Приложение 4. Содержание отчета по лабораторной работе) приведено описание отчета, который составляется в результате выполнения лабораторной работы.

2. Введение в проектирование, основные понятия проектирования

Введение в проектирование, основные понятия проектирования (процесс проектирования, объект проектирования, логическая схема проектирования, методология проектирования, проблемы в проектировании)

Проект (от латинского *projectus*-буквально: брошенный вперед)

- a) Совокупность документов (расчетов, чертежей) для создания какого либо сооружения или изделия;
- b) Предварительный текст документа;
- c) Замысел, план, прототип, прообраз какого либо объекта;

Объект проектирования

- a) Предвидимый объект;
- b) Некая система, которая будет и которая должна заполнить функциональную нишу во внешней среде;
- c) Конструируемый идеальный объект, чье качество меняется от одной стадии проектирования к другой стадии проектирования;
- d) Развивающаяся «модель будущего» ;
- e) Объект проектирования есть «процесс проектирования объекта», то есть система получения взаимосвязанных проектных решений (ПР) – моделей будущего;
- f) Объект проектирования раскрывается в логической схеме проектирования (ЛСП) через систему проектных решений;
- g) Объектом проектирования является жизненный цикл; жизненный цикл как структурно-процессуальный срез объекта проектирования оказывается тождественным самому объекту;
- h) Объект проектирования есть перевод цели в результат (процесс трансформаций исходных данных в результат) ;
- i) Объектом проектирования является информация об объекте проектирования, информационная модель объекта;

Процесс проектирования *(определение 1, относится к 70-годам 20 века)*

Информационно-логический процесс, состоящий из операций принятия проектных решений, выполняемых согласно некоторой методологии и приводящих к преобразованию цели в результат.

Проектирование *(определение 2, относится к 70-годам 20 века)*

Вид целенаправленной деятельности человека (или коллектива специалистов) по решению задач проектирования, направленной на создание устройств или систем, соответствующих техническому заданию, оптимально удовлетворяющих поставленным требованиям и удовлетворительно функционирующих в течение заданного промежутка времени при прогнозируемых условиях

Процесс проектирования

(определение 3, по глоссарию «Унифицированный процесс разработки программного обеспечения» Питер, 2002)

Основной рабочий процесс разработки программного обеспечения, целью которого является создание модели, содержащей проектные решения, удовлетворяющие функциональным и нефункциональным требованиям, а также ограничениям, относящимся к среде реализации. Процесс проектирования предназначен для подготовки к реализации и тестированию системы.

Проанализировав определения, сформулируем следующие выводы:

- a) процесс проектирования включает информационно-логические операции, которые тесно связаны с таким видом человеческой деятельности как мышление. На множестве этих операций задается отношение упорядочивания, которое должно приводить к цели проектирования
- b) Результатом информационно-логических операций являются решения, которые содержат «разворачивающуюся» от этапа к этапу модель будущей информационной системы. То есть решения образуют модель, включающую знания о будущей (будет разработана, реализована сред-

ствами той или иной технологии) информационной системе.

- с) Поскольку процесс проектирования тесно связан с мышлением, то для дальнейшего продвижения в понимании проектирования, необходимо получить, некоторые знания о мышлении. Такие знания можно получить в теориях, изучающих мышление. Теория, в которой в той или иной форме содержатся знания о мышлении, называется логика. Поэтому далее рассмотрим следующие теории: формальная логика, диалектическая логика. содержательно-генетическая логика.

Формальная логика.

Формальная логика — одна из дисциплин, в которой описывается и анализируется процесс мышления. С позиций формальной логики мышление рассматривается как система высказываний и переходов между ними — от одних к другим. Каждое из высказываний можно анализировать с точки зрения соответствия или несоответствия предмету высказывания, то есть как истинное или ложное. А сами переходы от высказывания к высказыванию — как обоснованные или необоснованные и осуществляемые по заранее фиксированным правилам.

Одним из разделов формальной логики является логическая онтология.

Онтология¹ (в широком смысле = греч. *ontos* - сущее и *logos* - мысль, слово, учение) это учение о видах бытия, составляющих **условие возможности мышления о мире и описания его в языке**.

В логической онтологии имеются следующие ключевые категории существующего: **предметы, свойства, отношения**

¹ Современная философская онтология (в отличие от натурфилософии на ранних этапах развития науки) занимается изучением *онтологических форм* явлений и отношений между такими формами, преднамеренно *абстрагируясь от конкретного содержания* этих явлений. Конкретное содержание существующих явлений (т.е. содержательная онтология в отличие от философской, *формальной онтологии*) есть предмет конкретно-научного знания [Современный, 1998]

Предмет это то, что может иметь свойства и вступать в отношения, но само не является свойством или отношением

Свойство это то, что каким-то образом характеризует вещь и не требует для своего описания более одной вещи.

Отношение это связь между двумя и более предметами. Отношение, в отличие от свойства, требует более одного предмета. Отношение превращается в свойство, если на всех его местах, кроме одного, вместо переменных стоят конкретные предметы.

Факт – *fact* - форма эмпирического познания; знание, достоверность которого строго установлена

Признак это наличие или отсутствие свойства или отношения. Признак это то, при помощи чего мы можем опознавать, отождествлять и различать предметы.

Простой признак это характеристика объекта, указывающая на наличие или отсутствие у него какого-то одного свойства или отношения (в том числе сложного).

Отличительный признак для данного множества это признак, присущий только объектам этого множества и не присущий никаким другим объектам.

Объект это предмет, свойство, отношение или множество

В дисциплине объектно-ориентированного проектирования (подробнее рассматривается в последующих лекциях и практических занятиях) **объектом** называется все то, что можно мысленно выделить из окружающей его среды, путем указания свойств и признаков, существенных для данного мыслительного образа.

Формальное представление объекта состоит из двух подмоделей:

1. Статическая модель объекта

$$O \Rightarrow (K, \Theta, \Phi, t);$$

K – идентификатор объекта

Θ - описание атрибутов объекта

Φ - описание операций (функций) объекта

t - здесь указывает на то, что статическое описание объекта всегда относится к какому-то моменту времени, то есть характеризует некоторое состояние объекта.

2. Динамическая модель объекта

$$\tilde{O} = \Phi(K, \Theta, X, t);$$

X - внешние и внутренние факторы

Динамическая модель (модель поведения объекта) есть процесс изменения его состояний во времени под воздействием множества X внешних и внутренних факторов

Понятие это мысль, которая обобщает объекты некоторого множества и выделяет это множество по отличительным для него признакам.

Объем понятия это множество объектов, выделяемых и обобщаемых в понятие по содержанию понятия

Содержание понятия это перечень всех признаков, по которым выделяются и обобщаются в понятие объекты

Процесс образования понятия:

1. *Анализ* мысленное разложение образца на отдельные признаки;
2. *Абстрагирование* (абстракция) это отвлечение признаков от предмета и превращение их в объект самостоятельного рассмотрения.

(Швейцарский психолог Жан Пиаже доказал, что нормальные дети до 10-11 лет не способны рассматривать признаки сами по себе, но только вместе с предметами, которым эти признаки присущи)

3. *Сравнение* рассмотрение различных видов предмета для выделения общих признаков и отбрасывания частных (т.е. присущих только отдельным видам признаков)
4. *Синтез* операция соединения признаков всех предметов данного множества в единый сложный признак, выделяющий рассматриваемое множество объектов из всех остальных.
5. *Познавательное обобщение* объединение разных объектов в одно множество по общим для них признакам.

Для того чтобы отчетливо мыслить и уметь передавать наше понятие об онтологии другим, мы должны его как-то обозначить, т.е. *выразить в языке*. Язык \approx это система знаков, служащих для хранения и передачи информации. Пусть даже это будут условные (формальные) языки. Поэтому, завершая тему онтологии дадим ее формальное определение (определение схемы), которое будет использоваться в данном курсе для построения концептуальных моделей предметных областей и информационных систем. Итак, формально онтология определяется :

$O = \langle X, R, F \rangle$, где

X – конечное множество понятий предметной области

R – конечное множество отношений между понятиями

F – конечное множество функций интерпретации

Диалектическая логика

Но формальная логика не объясняет многих вопросов, поэтому ее иногда называют “мертвым” мышлением. Это высказывания, суждения, различные словесные выражения мысли. Так, например, “живое” движущееся мышление, которое существует и развертывается в ситуации диалога, полилога, невозможно описать на основе формальной логики. Но это не означает, что логика вообще не нужна и

что логики для описания мышления не существует. Так для объяснения подобных вопросов существует другая логика, она называется **диалектической**.

Каковы характеристики диалектической логики? В соответствие с принципами этой логики подлинный предмет мышления описывается взаимопротиворечивыми утверждениями: A есть B и одновременно A не есть B . Это один из важнейших принципов диалектической логики, утверждающей, что законы развития знания определяются выделением подобных двойных высказываний и сведением их в противоречивые пары. И только при выделении подобных противоречивых пар можно говорить о восстановлении истины и правильного взгляда на некий описываемый предмет. Подобные парные высказывания могут быть обнаружены по всему полю содержательных представлений человечества в каждой дисциплине, в каждой области человеческой практики, в каждом мировоззренческом постулате (две параллельные прямые не пересекаются ни в какой точке - две параллельные прямые пересекаются в бесконечно удаленной точке).

Одним из методов познания, рассматриваемых в диалектике является метод «восхождения **от абстрактного к конкретному**». Применение этого метода предполагает логические операции: **анализ и синтез**, то есть мысленный анализ объектов и синтез получаемых в анализе знаний. Знания, получаемые в анализе, являются абстрактными по отношению к тому знанию, которое получается в синтезе. Синтетическое знание является конкретным по отношению к аналитическому. Конкретное (синтетическое) знание является не просто суммой абстрактных (аналитических) знаний, а новым знанием, получаемым из абстрактных знаний посредством специально изобретенных для этого логических операций. Эти операции специально изобретаются такими, чтобы результат их применения удовлетворил критериям соответствия некоторой предметной области. При анализе исследователь обеспечивает возможность введения **определений абстрагированных объектов**. Эти определения становятся явными или не явными **аксиомами**. При этом исследователь обеспечивает возможность открытия некоторых законов поведения системы. **При синтезе** исследователь выясняет, как эти законы согла-

суются с реальностью. Это происходит как исследование, упорядоченное определенными правилами науки.

Содержательно-генетическая логика

По одной из гипотез российских философов, ученых - методологов мышление имеет деятельностьную природу и деятельностьные механизмы осуществления. Разработанное ими представление о **мышлении как о деятельности** означало:

- а). Мышление можно и нужно целенаправленно строить как деятельность
- б). В мышлении можно ставить цели, создавать инструменты, средства и технологии;
- в). Необходимо постоянно вырабатывать нормы мышления, создавать его схематические, знаково-символические языки, а также анализировать и описывать процессы употребления знаний о мышлении.

Но если мышление - деятельность, **есть ли у мышления, рассматриваемого в форме деятельности, логика?** Такая логика была разработана и создана в свободных ассоциациях российских философов, логиков, методологов. Ассоциации эти назывались: *Московский методологический кружок* и *Комиссия по проблемам логики и психологии*, а созданная в них логика — **содержательно-генетической**. Эта логика, в оппозицию к формальной, была названа **содержательной**.

Основная задача этой логики состоит в том, чтобы описывать процессы употребления и происхождения знаний в мышлении. При этом знание не отождествляется с мышлением. Знание есть воспроизводимая структура мышления, очищенная от всего лишнего специально для задач этого воспроизводства.

Содержательная логика предназначена:

- а). для описания в мышлении **процессов понимания**, от которых абстрагируется (которые не учитывает) формальная логика

б). для учета работы **процессов сознания**, которые опять же не важны для описаний процессов суждения, осуществляемых при помощи языка формальной логики.

Важнейшим представлением, созданным и используемым в содержательно-генетической логике, является **идея уровней (или слоев)** замещения знаками процессов оперирования с **различными мыслительными предметами**. Подобное представление о мышлении как о **многослойном образовании**, имеющем принципы замыкания (обоснования) друг на друга различных слоев, характерно для понимания природы мышления в других, отличных от европейской системы философии науках, например в китайской философии. Очень важно понимать, что за этой **многослойностью мышления** стоит разнопозиционная мыследеятельность, где функция каждой из позиций символизирует различные слои мышления и функция ни одной из позиций не тождественна функциям других позиций.

Логическая схема проектирования

Познакомившись с основными видами «теорий мышления», определим некоторые правила мышления в процессе проектирования. Для этого используем понятие модели проектирования (организационно-технологической модели проектирования = ОТМП), задающей контекст правил мышления при проектировании:

ОТМП = <О, Т,МП,П,ОП, ПЦ>

О – множество объектов проектирования

Т – множество технологических процессов проектирования

МП – множество субъектов проектирования и их рабочих мест

П – множество проекторов (операторов проектирования)

ОП – множество отношений по проектированию между компонентами модели

ПЦ – множество проектных циклов

П – множество проекторов (операторов проектирования)

Оператор проектирования осуществляет перевод исходных данных в резуль-

таты проектирования – проектные решения (**ПР**)

Проектный цикл состоит из множества операторов проектирования, раскрываемых через логическую схему проектирования (например, **ОП пц1** – интегрированный оператор проектирования соответствующий 1-му процессу проектирования)

ОП – множество отношений по проектированию между компонентами модели

Множество отношений по проектированию включает в себя следующие отношения между компонентами ОТМП

1. Методологическое отношение
2. Логические отношения
3. Ресурсные
4. Технологические
5. Информационные

Методологическое отношение (в широком смысле) раскрывается посредством выполнения последовательности следующих составляющих:

1. Рефлексии ситуации (составления многопозиционной схемы, разработки модели)
2. Формулирования проблемы (оформление «пустоты» или проблематизация)
3. Решения проблемы (заполнения «пустоты»), посредством конструктивного процесса принятия решений (**ПР**), (процесса определенной конструкции; процесса на множестве операций которого задано конструктивное отношение)

Дадим определение процессу решения проблемы

Процесс решения проблемы (приводящий к результату), на множестве операций которого задано конструктивное отношение, **называется методологией проектирования в узком смысле этого слова или логической схемой проектирования (ЛСП)**

Понятие методологии, которое встречается в технической литературе последних лет по разработке информационных систем или программного

обеспечения, как правило, употребляется в узком смысле слова (методология проектирования)

Логические отношения

Логические отношения раскрывается посредством логической схемы проектирования (ЛСП) **или методологии проектирования в узком смысле слова**

$P \leftrightarrow \{Z (ИД, ОГ, РП), ПР, К(ПР)\} k, s.$

Где:

k, s – уровни, этапы (стадии)

Z – множество задач проектирования

ИД – множество исходных данных

ОГ – множество ограничений

РП – множество решающих процедур, методов

ПР – множество проектных решений

К(ПР) – множество качеств проектных решений

Из названий множеств, образующих ЛСП (или методологию в узком смысле) следует, что основными операциями процесса проектирования являются операции по решению задач с целью получения проектных решений (ПР) определенного качества. Поэтому основными **проблемами проектирования** можно назвать следующие проблемы:

1. Определение задач проектирования
2. Определение логической схемы проектирования
3. Решение задач проектирования (получение ПР с определенными характеристиками качества)
4. Сравнение полученных характеристик качества с заданными характеристиками качества и определение отклонений
5. Определение задач по устранению отклонений, с целью достижения оптимальных характеристик качества, и переход к следующему циклу проектирования (**или к следующей итерации**)

Определение задач проектирования

Определение перечня задач проектирования и их содержания является важнейшим этапом процесса проектирования. Задачи проектирования информационных систем на практике определяются исходя из результатов анализа ситуации, в которой начинается проект. В отдельных случаях, (например, при понимании возможностей информационных технологий), значительную часть задач формулирует заказчик, в других случаях задачи могут быть сформулированы на основании уже существующего решения (то есть по аналогии с имевшей место ситуацией).

Но каковы бы ни были ситуации, существует **стержневой подход** к формулированию задач проектирования, который в той или иной степени используют практически все проектировщики. Суть подхода в том, что в результате анализа концептуальной модели предметной области (ПрО) осуществляется поиск проблем (или проблематизация ПрО), которые в последствие должны быть разрешены с помощью информационной системы (ИС). То есть, должна быть разработана ИС, функциональные возможности которой позволяли бы полностью или частично разрешать проблемы ПрО. Функциональные возможности (или функции ИС) иногда в литературе называют задачами, которые решает ИС.

В итоге, перечень задач проектирования ИС это те задачи, которые должны способствовать разрешению проблем ПрО. Но как было сказано выше, эти задачи не появляются сами по себе. Они появляются в результате анализа концептуальной модели предметной области и ее проблематизации. То есть, перечень задач, их содержание в значительной степени определяются проблемами предметной области. Поэтому в список задач проектирования включают задачи моделирования предметной области и ее проблематизацию.

Задачи моделирования и проблематизации ПрО имеют не проектный, а исследовательский характер. То есть, при их решении проектировщик выступает как исследователь и выполняет соответствующие действия: обследование предметной области, отражение предметной области в модели (моделирование), исследование модели, проблематизация. В результате исследователь определяет в

виде требований перечень задач ИС (функциональных возможностей ИС), которые должны способствовать разрешению проблем ПрО. Причем, при определении задач исследователь применяет **метод замещения**, согласно которому операции, выполняемые в ПрО, замещаются операциями, выполняемыми с помощью (или полностью) функциональных возможностей ИС.

Понятия, необходимые для решения задач обследования, моделирования и проблематизации вводятся в разделах 3 и 4. В лабораторной работе, которая является неотъемлемой частью данного лекционного курса, указанные задачи решаются в процессе обследования и описания ПрО, разработки концептуальной модели ПрО с использованием UML, проблематизации ПрО и разработки концепции информационной системы.

Определение логической схемы проектирования (методологии проектирования)

Определение или выбор методологии, посредством которой организуется и осуществляется процесс проектирования и разработки ответственная задача.

На практике компании, занимающиеся разработкой программного обеспечения, выбирают методологию на длительное время. Во многом, выбор методологии зависит от характера задач, которые должна решать ИС (или отдельные ее приложения) на протяжении жизненного цикла. Исходя из ситуации на отечественном рынке программного обеспечения и ИС в целом, такой выбор должен быть, сделан в пользу универсальных методологий проектирования и разработки. Но разрабатывать ИС согласно универсальным, учитывающим множество аспектов разработки сложная организационная и техническая задача, требующая затрат. То есть использовать для проектирования и разработки такие монументальные методологии дорогое удовольствие.

Альтернативой монументальным методологиям являются так называемые методологии экстремального проектирования и разработки. Основное отличие этих методологий от монументальных в том, что они практически не используют документирования проектных решений, как на концептуальном, так и на логическом уровне. А основным документом считают тексты программ, разработанных в

обеспечение работы ИС.

В данном лекционном курсе в качестве методологии разработки рассматриваются основы универсальной методологии RUP (Rational Unified Process), разработанной фирмой Rational Software. Эта методология согласуется с рекомендациями международного стандарта ГОСТ Р ИСО \ МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств», действующего в РФ с 2000 года. Ценность этой методологии (RUP) не только в том, что она является универсальной и может с методической точки зрения поддерживать множество проектов по разработке ИС, но и в том, что она построена по компонентному принципу и может достаточно легко быть модифицирована для поддержки небольших проектов.

Методология существует в форме базы знаний, работа с которой обеспечивается специальным программным обеспечением. Поэтому базу знаний можно установить на персональный компьютер и использовать при организации процессов проектирования и разработки ИС. Основные положения RUP описаны в разделе 9

Решение задач проектирования

Задачи проектирования и разработки ИС достаточно детально рассматриваются в методологии RUP.

Процесс решения задач проектирования и разработки ИС и проект в целом, должен согласно RUP, подчиняться следующим основным положениям:

- а). Управляться вариантами использования (прецедентами),
- б). Ориентироваться на архитектуру ИС
- в). Носить итеративный и инкрементный характер

Управление вариантами использования означает, что в процессе разработки и проектирования выполняются серии **рабочих процессов** (отрабатываются управляющие воздействия), **порожденные вариантами использования**. Но поскольку варианты использования управляют процессом разработки, то они должны разрабатываться совместно с архитектурой системы. Таким образом, вариан-

ты использования управляют и архитектурой, а архитектура, в свою очередь, оказывает влияние на варианты использования. Причем и варианты использования и архитектура развиваются в процессе жизненного цикла по правилам, определяемым отношением итеративности и инкрементности.

Архитектура - это представление всего проекта ИС с выделением ключевых составляющих и затумашивание деталей. Архитектура вырастает из требований к результату, в том виде, как их понимает пользователь и другие заинтересованные лица. **Архитектура определяется** в виде представлений всех **моделей** системы, объединенных (сконфигурированных) в систему. В качестве основного архитектурного представления, задающего содержание процесса проектирования и содержание моделей, в данном курсе и лабораторной работе рассматривается **комплексная архитектура предприятия (Enterprise Architecture)** более известная как схема Захмана. Рассмотрению комплексной архитектуры, являющейся стандартом де-факто, посвящен раздел 5.

Но методология RUP, не предлагает удовлетворительных методов решение задач концептуального моделирования ПрО, и проблематизации, а также ряда задач проектирования и разработки ИС. В таких случаях обычно используют более широкую теорию, которая является методологической основой решения указанных задач. К таким теориям можно отнести системный подход, общую теорию систем и системный анализ

3. Понятие системы. Понятие системного подхода и системного анализа

Понятие системы. Понятие системного подхода и системного анализа. Понятие модели и моделирования. Системное проектирование

Рассмотрим следующие определения понятие системы:

- 1) **Система** – *system* – любой объект, который одновременно рассматривается и как единое целое, и как совокупность разнородных объектов, объединенных для достижения определенного результата
- 2) **СИСТЕМА** - (ГОСТ Р ИСО/МЭК12207:99): комплекс, состоящий из процессов, технических и программных средств, устройств и персонала, обладающий возможностью удовлетворять установленным потребностям или целям
- 3) **Система** (*system*) (ММК, 1985)

“Напомним, что основной смысл категории системы в ее современном виде состоит в требовании, что всякий объект исследования, который мы хотим представить в виде моносистемы, должен быть расслоен, по меньшей мере, в пять представлений (Рисунок 1):

- а). **процесса**, конституирующего данную моносистему;
- б). **набора элементов и связей между ними**, образующих структуру этой системы; далее эта структура может быть фокусирована либо на связях структуры и тогда мы получаем чистую структуру внутренних связей системы, либо на элементах структуры и тогда, в зависимости от способа фокусировки, мы получим либо состав элементов системы, либо множество фокусированных элементов системы с задающими и определяющими их структурами функций, которые обычно характеризуются как внутренняя структура функций элементов системы;
- в). **набора внешних функций системы**, которые вводятся, исходя из объемлющих ее систем аналогично методу фокусировки структуры связей на одном элементе объемлющей системы и образуют внешнюю структуру функций системы;

- г). **организованностей материала системы**, которые обеспечивают протекание процесса, конституирующего данную систему, и закрепление его на этом материале. По традиции множество таких организованностей материала называется морфологией системы;
- д). **материала**, на котором система разворачивается и строит себя”

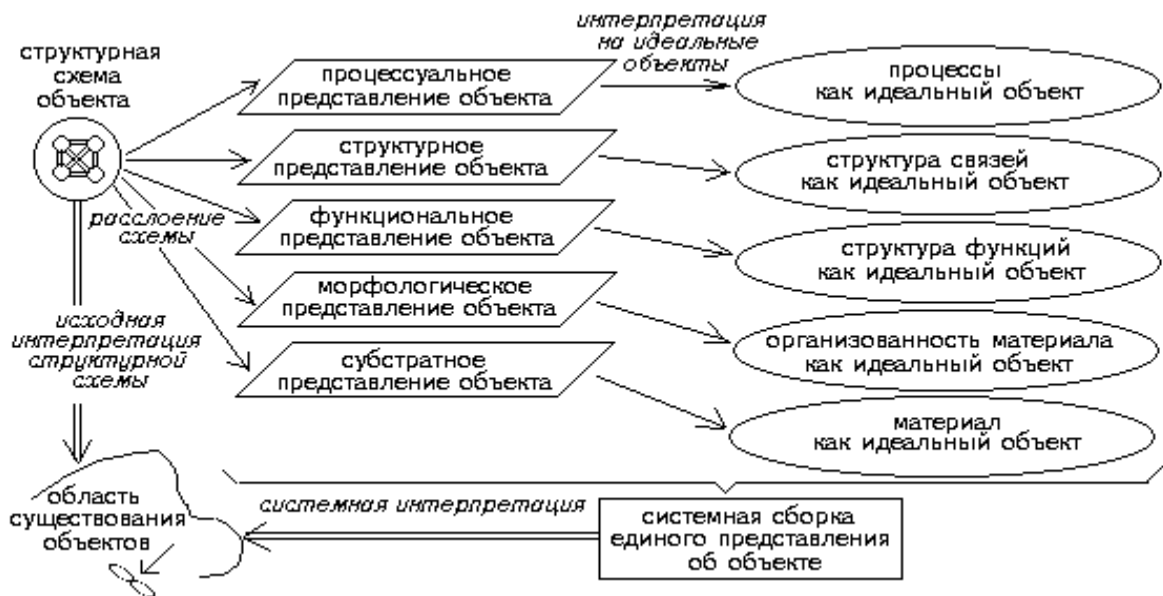


Рисунок 1

- 4) **Формальная система** - конечное множество принятых по соглашению (Фактически соглашение обуславливается всегда только дальнейшей действительной или кажущейся полезностью вытекающих из него построений для решения стоящих перед проектировщиком задач) символов, называемых формулами и терминами, и конечное число точных правил оперирования этими символами, которые дают возможность образовать из символов некоторые комбинации

Рассмотрим одно из формальных представлений систем управления. Для этого введем понятие **формальной модели**. Назовем кортеж (1) формальной моделью

$$\Psi \equiv \langle \{M\}, P_1, P_2 \dots P_n \rangle \quad (1)$$

Компоненты формальной модели имеют следующий смысл:

M - множество элементов модели, соответствующих элементам предметной области, называемое **носителем модели**

P₁, P₂, ... P_n - предикаты, отображающие наличие того или иного отношения между элементами предметной области.

Предикат (определение 1) это логическая n-арная пропозициональная функция, определенная для предметной области и принимающая значения либо истинности, либо ложности.

В контексте нашего курса представляется полезным знать еще одно определение понятия предикат, которое приводится в ГОСТ 34.320-96

Предикат (определение 2): лингвистический объект, аналогичный глаголу, сообщающий что-либо о сущностях, обозначенных терминами.

Терм: лингвистический объект, обозначающий сущность.

Лингвистический объект: грамматически допустимая языковая конструкция.

Например, предложение «Автомобиль РСХХ999 является моделью Мустанг» является примером высказывания. В этом предложении глагол «является» представляет собой предикат. Термы «автомобиль РСХХ999» и «модель Мустанг» относятся к сущностям.

Носитель модели является содержательной областью предикатов P₁, P₂, ... P_n. Множество предикатов **образует сигнатуру** модели Ψ.

Выбор носителя и сигнатуры при определении модели определяется предметом исследования. **Введем понятие системы**, основанное на понятии модели и содержащее возможность сопоставления поведения и структуры (преобразования типа Ψ_a → Ψ_b)

Системой называется кортеж:

$$S \equiv \langle \Psi_a, \Psi_b, P(\Psi_a, \Psi_b) \rangle \quad (2)$$

Где Ψ_a - подмодель определяющая поведение системы. Иногда эта подмодель может рассматриваться как <черный ящик>, о котором известно лишь то, что на определенное воздействие он реагирует лишь определенным образом.

Ψ_b – подмодель, определяющая структуру системы при ее внутреннем рассмотрении

$P_0(\Psi_a, \Psi_b)$ - предикат целостности, определяющий назначение системы, семантику подмоделей (Ψ_a, Ψ_b), а также семантику сопоставления (преобразования $\Psi_a \rightarrow \Psi_b$). Наличие предиката целостности позволяет говорить о том, что такое представление системы - это семантическая модель, имеющая внутреннюю интерпретацию.

Подмодель (Ψ_a) может быть представлена в виде следующего кортежа

$$\Psi_{a \equiv} \langle X, Y, Z, f, g \rangle \quad (3)$$

(X) – входной сигнал

(Y) – выходной сигнал

(Z) – переменная состояния подмодели Ψ_a

(f,g) – функционалы (глобальные уравнения системы), задающие текущее значение выходного сигнала и внутреннего состояния.

Кроме указанных выражений систему задают следующими аксиомами:

Аксиома 1. Для системы определены пространство состояний Z , в которых может находиться система и параметрическое пространство T , в котором задано поведение системы. В связи с этим системы, математически описываемые как подмодель поведения (Ψ_a), принято называть динамическими системами, так как они отражают способность систем изменять состояние в параметрическом, пространстве T . В отличие от динамических статические системы таким свойством не обладают. В качестве параметрического пространства обычно рассматривают временной интервал.

Аксиома 2. Пространство состояний Z содержит не менее двух элементов. Эта аксиома отражает представление о том, что сложная система может находиться в разных состояниях

Аксиома 3. Система обладает свойством функциональной эмерджентности.

При таком рассмотрении система является совокупностью моделей и, главное, отражает семантику предметной области, в отличие от неинтерпретированных частных математических моделей. Другими словами, система – это совокупность элементов, обладающая интегративными свойствами (эмерджентностью), а также способ отражения реальных объектов

Системный подход, направление методологии специально-научного познания и социальной практики, в основе которого лежит исследование объектов как систем. С. п. способствует адекватной постановке проблем в конкретных науках и выработке эффективной стратегии их изучения. Методология, специфика С. п. определяется тем, что он ориентирует исследование на раскрытие целостности объекта и обеспечивающих её механизмов, на выявление многообразных типов связей сложного объекта и сведение их в единую теоретическую картину

Системный анализ – *systems analysis* - в узком смысле совокупность методологических средств, используемых для подготовки и обоснования решений по сложным проблемам различного характера. Он опирается на системный подход, а также на ряд математических методов и современных методов управления. Основная процедура – построение обобщенной модели, отображающей взаимосвязи реальной ситуации. Полученная модель исследуется с целью выяснения близости результата применения того или иного из альтернативных вариантов действий к желаемому результату, сравнительных затрат ресурсов по каждому из вариантов, степени чувствительности модели к различным нежелательным внешним воздействиям. Термин появился в связи задачами военного управления в исследованиях RAND Corporation (1948). А в отечественной литературе после выхода книги С. Оптнера «Системный анализ для решения деловых и промышленных проблем» (1969) с предисловием С. П.Никанорова.

Ниже приводится та часть предисловия, которая описывает основные положения системного анализа как методологии решения крупных проблем. Текст практически соответствует оригиналу, поскольку обладает одновременно и ясностью и краткостью и вообще считается классическим текстом по системному

анализу.

«Системный анализ—это методология решения крупных проблем, основанная на концепции систем. Системный анализ может также рассматриваться как методология построения организаций, поскольку организации могут рассматриваться как то, что реализует методологию решения проблем. Оба эти определения неразрывно связаны, однако мы вначале рассмотрим методологию решения проблем как таковую, а затем ее влияние на организацию. При этом мы будем в основном следовать С. Л. Оптнеру и С. Янгу.

В центре методологии системного анализа находится операция количественного сравнения альтернатив, которая выполняется с целью выбора альтернативы, подлежащей реализации. Если требование равнокачественности альтернатив выполнено, могут быть получены количественные оценки. Но для того, чтобы количественные оценки позволяли вести сравнение альтернатив, они должны отражать участвующие в сравнении свойства альтернатив (выходной результат, эффективность, стоимость и другие). Достичь этого можно, если учтены все элементы альтернативы и даны правильные оценки каждому элементу. Так возникает идея выделения «всех элементов, связанных с данной альтернативой», т. е. идея, которая на быденном языке выражается как «всесторонний учет всех обстоятельств». Выделяемая этим определением целостность и называется в системном анализе полной системой или просто системой. Система, таким образом, есть то, что решает проблему.

Но как выделить эту целостность, «систему», как установить, входит данный элемент в данную альтернативу или нет? Единственным критерием может быть участие данного элемента в процессе, приводящем к появлению выходного результата данной альтернативы. Коль скоро это так, понятие процесса оказывается центральным понятием системного анализа.

Таким образом, то, что, прежде всего, должно быть выделено, если мы хотим думать и действовать «системно», есть процесс. Не может быть системного мышления без ясного понимания процесса.

Система определяется заданием системных объектов, свойств и связей. Системные объекты — это **вход, процесс, выход, обратная связь и ограничение**.

Входом называется то, что изменяется при протекании данного процесса. Во многих случаях компонентами входа являются «рабочий вход» (то, что «обрабатывается») и процессор (то, что «обрабатывает»). Выходом называется результат или конечное состояние процесса. Процесс переводит вход в выход. Способность переводить данный вход в данный выход называется свойством данного процесса. Связь определяет следование процессов, т. е. что выход некоторого процесса является входом определенного процесса. Всякий вход системы, является выходом этой или другой системы, а всякий выход—входом. Выделить систему в реальном мире значит указать все процессы, дающие данный выход. Искусственные системы это такие, элементы которых сделаны людьми т.е. являются выходом сознательно выполняемых процессов человека.

Во всякой искусственной системе существуют три различных по своей роли подпроцесса: основной процесс, обратная связь и ограничение. Основной процесс преобразует вход в выход. Обратная связь выполняет ряд операций: сравнивает выборку выхода с моделью выхода и выделяет различие, оценивает содержание и смысл различия, вырабатывает решение, сочлененное с различием, формирует процесс ввода решения (вмешательства в процесс системы) и воздействует на процесс с целью сближения выхода и модели выхода. Процесс ограничения возбуждается потребителем (покупателем) выхода системы, анализирующим ее выход. Этот процесс воздействует на выход и управление системы, обеспечивая соответствие выхода системы целям потребителя. Ограничение системы, принимаемое в результате процесса ограничения, отражается моделью выхода. Ограничение системы состоит из цели (функции) системы и принуждающих связей (качеств функции). Принуждающие связи должны быть совместимы с целью.

Всякая система состоит из подсистем. Всякая система является подсистемой некоторой системы. Постулируется, что любая система может быть описана в

терминах системных объектов, свойств и связей. Граница системы определяется совокупностью входов от окружающей среды. Окружающая среда—это совокупность естественных и искусственных систем, для которых данная система не является функциональной подсистемой.

Проблемой называется ситуация, характеризующаяся различием между необходимым (желаемым) выходом и существующим выходом. Выход является необходимым, если его отсутствие создает угрозу существованию или развитию системы. Существующий выход обеспечивается существующей системой. Желаемый выход обеспечивается желаемой системой. Проблема есть разница между существующей и желаемой системой. Проблема может заключаться в предотвращении уменьшения выхода или же в увеличении выхода. Условие проблемы представляет существующую систему («известное»). Требование представляет желаемую систему. Решение проблемы есть то, что заполняет промежуток между существующей и желаемой системами. Система, заполняющая промежуток, является объектом конструирования и называется решением проблемы.

Проблемы могут проявляться в симптомах. Систематически проявляющиеся симптомы образуют тенденцию. Обнаружение проблемы есть результат процесса идентификаций симптомов. Идентификация возможна при условии знания нормы или желательного поведения системы. За обнаружением проблемы следует прогнозирование ее развития и оценка актуальности ее решения, т.е. состояния системы при нерешенной проблеме. Оценка актуальности решения проблемы позволяет определить необходимость ее решения.

Процесс нахождения решения концентрируется вокруг итеративно выполняемых операций идентификации условия, цели и возможностей для решения проблемы. Результатом идентификации является описание условия, цели и возможностей в терминах системных объектов (входа, процесса, выхода, обратной связи и ограничения), свойств и связей, т. е. в терминах структур и входящих в них элементов. Если структуры и элементы условия, цели и возможностей данной проблемы известны, идентификация имеет характер определения количествен-

ных отношений, а проблема называется количественной. Если структура и элементы условия, цели и возможностей известны частично, идентификация имеет качественный характер, а проблема называется качественной или слабоструктурированной. Как методология решения проблем системный анализ указывает принципиально необходимую последовательность взаимосвязанных операций, которая (в самых общих чертах) состоит из выявления проблемы, конструирования решения проблемы и реализации этого решения. Процесс решения представляет собой конструирование, оценку и отбор альтернатив систем по критериям стоимости, времени, эффективности и риска с учетом отношений между предельными значениями приращений этих величин (маргинальных отношений). Выбор границ этого процесса определяется условием, целью и возможностями его реализации. Наиболее адекватное построение этого процесса предполагает всестороннее использование эвристических заключений в рамках постулированной структуры системной методологии.

Редуцирование числа переменных производится на основе анализа чувствительности проблемы к изменению отдельных переменных или групп переменных, агрегирования переменных в сводные факторы, выбором подходящей формы критериев, а также применением там, где это возможно, математических способов сокращения перебора (методов математического программирования и т. п.).

Логическая целостность процесса обеспечивается явными или скрытыми предположениями, каждое из которых может являться источником риска. Постулируется, что структура функций системы и решения проблемы является стандартной для любых систем и любых проблем. Меняться могут только методы выполнения функций. Совершенствование методов при данном состоянии научных знаний имеет предел, определяемый как потенциально достижимый уровень. В результате решения проблемы устанавливаются новые связи и отношения, часть которых обуславливает желаемый выход, а другая часть определяет непредвиденные возможности и ограничения, которые могут стать источником будущих проблем. «»

Таковы в общих чертах основные представления системного анализа как методологии решения проблем.

«»Применение системного анализа на практике может происходить в двух ситуациях: когда исходным пунктом является появление новой проблемы и когда исходным пунктом является новая возможность, найденная вне непосредственной связи с данным кругом проблем. Решение проблемы в ситуации новой проблемы проводится по следующим основным этапам: обнаружение проблемы, оценка ее актуальности, определение цели и принуждающих связей, определение критериев, вскрытие структуры существующей системы, определение дефектных элементов существующей системы, ограничивающих получение заданного выхода, оценка веса их влияния на определяемые критериями выходы системы, определение структуры для построения набора альтернатив, построение набора альтернатив, оценка альтернатив, выбор альтернатив для реализации, определение процесса реализации, согласование найденного решения, реализация решения, оценка результатов реализации решения.

Реализация новой возможности проходит другим путем. Использование данной возможности в данной области зависит от наличия в ней или в смежных областях актуальной проблемы, нуждающейся для своего разрешения в такой возможности. Использование возможностей в отсутствие проблем может таить в себе, как минимум, бесполезную растрату ресурсов. Использование возможностей при наличии проблем, но игнорирующее проблемы, превращающееся в самоцель, может способствовать углублению и обострению проблемы. Развитие науки и техники приводит к тому, что возникновение ситуации новой возможности становится заурядным явлением. Это требует серьезного анализа ситуации при появлении новой возможности. Возможность утилизируется, если лучшая альтернатива включает в себя эту возможность. В противоположном случае возможность может остаться неиспользованной. Внедрение новой техники на основе одного только критерия срока самокупаемости может быть примером подхода, когда утилизация новой технической возможности осуществляется вне анализа проблем. Большой процент неудач при внедрении машинных систем управления в

США на первом этапе их создания является в значительной мере следствием отсутствия в этот период проблемно-ориентированного подхода. «»

Рассмотрим теперь, каким образом системный анализ представляет организацию.

Несвоевременное, расточительное решение или же обострение проблемы и возникающие, как следствие, потери свидетельствуют о том, что механизм контроля состояния системы, в которой возникла проблема, выработки и реализации необходимых решений работает неудовлетворительно. Например, это могло быть при определении перспективной для данного рынка продукции или при принятии на вооружение данной технической системы. Но неудовлетворительная работа этого механизма означает неудовлетворительную работу организации, реализующей этот механизм. Улучшение его работы может быть достигнуто улучшением выполнения функций решения проблем, предусматриваемых системным анализом. Для этого необходимо рассматривать организацию не как структуру подчинения с установленными или сложившимися отношениями, а как процесс решения проблемы. Такой подход позволяет рассматривать организацию как систему, а для ее описания, изучения и улучшения использовать концептуальный аппарат системного анализа.

Для улучшения выполнения функций решения проблем, реализуемых организацией, могут быть использованы разнообразные методы: от рационализации форм документов до применения математических моделей и вычислительных машин. Методы могут, следовательно, иметь альтернативы, их отбор может производиться в соответствии с принципами системного анализа. «Мощность» всех функциональных подсистем от обнаружения (идентификации) проблем до реализации решения должна быть примерно одинаковой. Бессмысленно иметь мощные методы выработки решения, если функция идентификации состояния выполняется неудовлетворительно. Решение о совершенствовании организации должно вырастать из ее проблем и соответствовать им по масштабу и сложности. Таким образом, отдельные методы совершенствования функций могут найти свое место

только при конструировании организации как целостной системы.

В прошлом роль научных методов выполнения отдельных функций была существенно ограниченной из-за слабости или отсутствия методов. Поэтому вопрос о рассмотрении организации как системы не мог возникнуть. Сейчас для реализации многих функций решения проблем созданы весьма совершенные методы; их разработка интенсивно и целенаправленно продолжается. Однако применение отдельных методов в рамках существующей организации затруднительно и малоэффективно. Причина заключается в том, что применение метода требует вычленения функции как целостного процесса из последовательности операций, в которой она традиционно выполняется, что без изменения способа работы существующих организаций невозможно. В существующих организациях, которые складывались, реализуя наличные, по существу эвристические методы, а не конструировались сознательно и в которых вследствие этого исторический элемент преобладает над логическим, функции почти никогда не бывают вычленены так, как это нужно для применения мощных методов. Другая причина может заключаться в бюрократическом характере существующих организаций. Картина существенно усложняется, если имеется в виду не один метод, а целый набор, и не одна частная функция, а целый комплекс связанных между собой функций. Существующая организация может доступными ей средствами решать проблемы, но она не может эффективно использовать для их решения современный научный инструмент.

Когда был изобретен бензиновый двигатель, он был поставлен на деревянную коляску. По мере введения других усовершенствований коляска изменялась, превращаясь в современный автомобиль. Но современный двигатель мощностью в триста сил нельзя поставить на легкую деревянную коляску. Подобным образом обстоит дело и при применении мощных современных методов (правил решения), таких, например, как транспортные модели, модели очереди, сетевые модели в рамках сложившихся организаций.

Организации, следовательно, должны строиться вокруг методов выполне-

ния функций. Операционный смысл любой модели, используемой в организации, тот же, что и обычного совещания, на котором те, кто формулирует правила решения или сами решения, проводят всестороннее обсуждение предполагаемых решений или правил их построения.

Для оценки альтернатив конструкций организационных систем используются критерии измеримости, эффективности, надежности, оптимальности и стабильности. Измеримость - способность системы измерить свои характеристики. Эффективность - возможность решить проблему с помощью данной системы. Если система не имеет измеримости, то нельзя определить, дает ли она улучшение или ухудшение. Эффективность предполагает баланс между частями системы. Недостаточная эффективность будет заставлять руководителя возвращаться к самой примитивной части всей системы, так как в несбалансированной системе она представляет основное ограничение. Иметь решение, которое оптимально, но не измеримо и не эффективно и бессмысленно. Решение должно быть измеримым, эффективным и надежным прежде, чем можно будет рассматривать его оптимальность.

Руководство, занятое решением отдельной проблемы, то есть созданием системы, решающей проблему, называется системным руководством. Комплекс работ по решению отдельной проблемы называется программой. Поэтому системное руководство называется иногда программным руководством.

Сводя решение проблемы к конструированию системы, системный анализ, по существу, перенес в область организации методы, хорошо известные в практике инженерной разработки технических систем, придал решению организационных проблем характер исследовательской и инженерно-конструкторской деятельности. Некоторые ученые считают, что перестройка организаций в соответствии с требованиями системного анализа приведет к «потрясающим переменам в руководстве в ближайшие десять лет».

Таковы основные установки системного анализа в области организации. «»

Завершая рассмотрение основных положений системного анализа для ор-

ганизаций (предприятий) уточним понятие системного руководства (). Это уточнение будет важно для нас при выработке понятия системного проектирования.

«Задача высшего руководства организации - не выработка решений, а конструирование процесса выработки решения и наблюдение за его действием. Способность руководителя среднего звена предлагать хорошие решения не является основанием для выдвижения его в состав высшего руководства. Это было бы подобно тому, чтобы поручать проектирование грузоподъемной машины штангисту, на основании того, что он хорошо поднимает тяжести. «»

Согласно этому уточнению **руководство организацией можно считать системным** только в том случае, когда топ – менеджеры конструируют и внедряют процесс выработки решений, осуществляют его мониторинг и, при необходимости, согласно правилам, оговоренным в конструкции процесса, принимают конкретные решения.

По аналогии с системным руководством в ряде отраслей промышленности появилась необходимость удерживать знания в той или иной деятельности посредством соответствующего процессуального отношения. Так, например, в конце прошлого столетия появилось системное проектирование, задача которого не только решение проектных задач в классическом понимании, а конструирование и внедрение такого процесса проектирования, который бы интегрировал решения различных дисциплин и определял средства, необходимые для успешного использования созданных систем на протяжении их жизненного цикла.

Системное проектирование (Systems Engineering, *INCOSE*) – междисциплинарный подход и средства, предназначенные для создания успешных систем. Фокусируется на определении нужд потребителя и требуемой функциональности в начале цикла разработки, на документации требований, с переходом к конструкторскому синтезу и комплексной аттестации системы при полном учете следующих проблем: функционирование, производительность, испытания, изготовление, затраты и планирование, обучение и сопровождение, вывод из эксплуатации. Системное проектирование интегрирует все нужные дисциплины и группы специалистов в командные усилия, формируя структурированный процесс разработ-

ки, который выполняется от формирования концепции до осуществления продуктивной работы системы. В системном проектировании учитываются и нужды бизнеса и технические потребности всех клиентов для получения качественного продукта, который отвечает потребностям пользователей. [INCOSE (*International Council on Systems Engineering*) <http://www.incose.org/whatis.html>]

Известными примерами таких процессов являются процессы (методологии) разработки программных систем. Одной из наиболее популярных методологий, основы которой используются в данном лекционном курсе, является методология RUP, разработанная компанией Rational Software.

Рассмотрев основные понятия системного анализа вообще и для организаций в частности, которые были сформулированы примерно в начале шестидесятых годов прошлого столетия, отметим, что работы по системному анализу в тот период во многом базировались на идеях теории оптимизации и исследовании операций. При этом особое внимание уделялось стремлению получить выражение, связывающее цель со средствами, аналогичное критерию функционирования или показателю эффективности, то есть отобразить объект в виде хорошо организованной системы. Так, например, в ранних руководящих материалах по разработке ИС (АСУ) рекомендовалось представлять цели в виде набора задач и составлять матрицы, связывающие задачи с методами и средствами их достижения. При практическом применении такого подхода выяснилась, его недостаточность и проектировщики стали обращать внимание на построение моделей, которые не просто фиксируют цели, компоненты и связи между ними, а позволяют накапливать информацию, вводить новые компоненты, вводить новые связи, то есть отображать объект в виде развивающейся системы. (правда не всегда предлагается как это сделать)

Поэтому в следующем разделе рассмотрим подробнее понятие модели как ключевого понятия системного анализа и понятия, без которого не обходится ни одна современная методология проектирования информационных систем.

Расширение понятия модели.

В последнее десятилетие в технической литературе понятие **модель** все чаще понимается как абстрактное знание (в смысле абстрагированных объектов, раздела диалектической логики). Некоторые авторы настаивают на том, что такое понимание не корректно и приводят следующие аргументы:

1. Модель (с точки зрения логики и методологии науки) есть имитация **существующего** или **проектируемого объекта**, создаваемая для решения каких либо проблем, касающихся моделируемого объекта, не путем изучения самого объекта непосредственно (его может и не быть в реальности), а путем манипуляций с моделью как своего рода заменителем объекта.
2. Модель специально создается или **выдумывается** такую, чтобы имело место ее **соответствие** и объекта по определенным признакам, чтобы можно было из знаний, полученных на модели по определенным (заранее известным правилам) получить знание об объекте. Модель объекта не есть знание об объекте, а есть средство получения знаний об объекте.
3. Абстрактное же знание есть именно знание об объекте. И правила согласования абстрактного и конкретного знания отличаются от правил моделирования. Это правила образующие в совокупности метод восхождения от абстрактного к конкретному.

Определение **смысла понятия модель** важно, так как это понятие часто используется в материале данного курса, особенно при рассмотрении методологии RUP (концептуальная модель, модель проектирования и т.п.). В данном лекционном курсе понятие модели, а также понятия анализа и синтеза рассматриваются как взаимообусловленные. Например, порождаемый в результате логической операции анализа набор моделей объекта (описание объекта с разных сторон) образует **аналитическую (концептуальную) модель** объекта, а модели, порожденные в результате логической операции синтеза, образуют **проектную (логическую) модель**. То есть модели используются как формы, в которых содержаться знания о проектируемой системе (объекте). Причем, в процессе ис-

пользования знаний компоненты модели могут уточняться и получать дополнительную или другую интерпретацию.

Далее будет показано, что для процесса проектирования информационных систем, **знания об объекте** проектирования в форме моделей являются ключевыми знаниями. С другой стороны, в данном лекционном курсе понятие модель используется и как средство получения знаний об объекте. Так, например, концептуальная модель предметной области специально создается в **соответствии** с объектом (например, предприятием), чтобы можно было из знаний, полученных на модели по определенным (заранее известным правилам) получить другое (дополнительное) знание об объекте. Поэтому в данном курсе модель понимается и как знания о проектируемом объекте, и средство получения знаний об объекте.

Приведем наиболее известные в области информационных технологий определения понятия модель.

- 1) **Модель – *simulator*** – программа либо устройство, обеспечивающее имитацию характеристик и поведения определённого объекта.
- 2) **Модель (Model)** [стандарт ISO-15704] Абстрактное представление реальности в какой-либо форме (в математической, физической, символической, графической, дескриптивной), предназначенное для представления определенных аспектов этой реальности и позволяющее отвечать на изучаемые вопросы.
- 3) **Модель** – это абстракция, описывающая моделируемую систему (объект) с определенной точки зрения и на определенном уровне абстрагирования. Под точкой зрения будем понимать представление аналитика требований или проектировщика

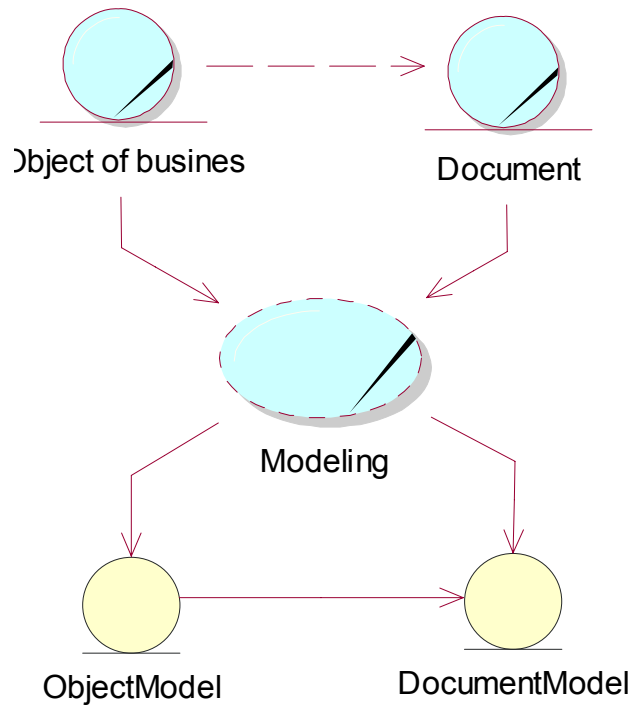


Рисунок 2

4) Моделирование (определение 1)

Разработка модели объекта для использования в процессах проектирования, производства, тестирования, эксплуатации с целью получения знаний об объекте. Моделирование предполагает построение и изучение моделей проектируемых (конструируемых) объектов, реально существующих предметов и явлений.

5) Моделирование (определение 2)

Разработка в процессе проектирования модели объекта, содержащей знания (решения) о проектируемом объекте (структура, функционирование и т.п.). Причем, в процессе использования знаний компоненты модели могут уточняться и получать дополнительную или другую интерпретацию (Рисунок 2)

Перед тем как перейти к обсуждению понятия электронный документ определимся с такими достаточно известными понятиями как онтология и концептуальная модель.

Понятие «**онтология**» формально в данном курсе определено как триада множеств $O = \langle X, R, F \rangle$ и определяется как спецификация (описание) некоторой концептуализации. Концептуализация - продукт концептуального анализа, а содержание последнего обычно интерпретируют как «абстрагирование существа физических объектов (X) и их взаимосвязи (R) в виде некоторой модели (F) или теории, которая соответственно называется концептуальной моделью изучаемого объекта. В любой онтологии каждое определение понятия (термин) требует полного понимания в отношении с другими определениями понятий (терминов) данной онтологии. В этом аспекте определения онтологии отличаются от определений понятий (терминов) в обычных словарях, трактующих каждый термин в отдельности.

Поэтому для задач решаемых в данном лекционном курсе понятия онтологии и концептуальной модели будем считать семантически эквивалентные (имеющие одинаковый смысл).

4. Документ. Электронный документ. Информационная система. Информационная технология.

Информационная система. Документ. Электронный документ, Информационная технология

Документ (бумажный документ) – *document* – материальный объект, содержащий в зафиксированном виде информацию (договор, счет, заказ, пропуск и т.п.), оформленную установленным порядком и имеющую в соответствии с действующим законодательством правовое значение.

Пример: Вы написали заявление на отпуск и передали его в отдел кадров. Так появился документ. Что же превратило чистый лист бумаги в документ? Во-первых, информация, представленная в виде текста. Во-вторых, текст в форме заявления. И, в-третьих, бумагу готовили с расчетом на последующую деятельность сотрудников отдела кадров. Теперь предположим, что вы обратились в отдел кадров с устным заявлением об отпуске. Можно ли назвать документом эту процедуру? Устная беседа не была зафиксирована физически, она не поддается точному воспроизведению и, следовательно, документом не является.

Итак, документ - это совокупность трех составляющих:

1. Физическая регистрация информации.
2. Форма представления информации
3. Активизация определенной деятельности.

Именно некоторая деятельность и превращает информацию в документ. Но документ перестает существовать, если в дальнейшем не подразумевает процедуры воспроизводства определенной деятельности. Так родилась бюрократия - неизбежный спутник цивилизации.

Бюрократическая технология - это технология взаимодействия людей, служб и подразделений внутри и вне организации. Не будет технологии - возникнет анархия. Если работник не знает что ему надо делать, он делает то, что считает нужным, а не то, что требует тот или иной бизнес-процесс предприятия. Сама

бюрократия неизбежна, опасность представляет отрыв реальных целей предприятия от работы текущей системы документооборота.

Документооборот - движение документов в организации с момента их создания или получения до завершения их исполнения и передачи в архив.

Кроме собственно документов важен еще регламент работы с ними. Любой менеджер может подтвердить, что работа не по регламенту порой отнимает намного больше времени, чем собственно производственная деятельность. Дублирование документов, их потеря, навязчивый способ их распространения, а также запутанный порядок их прохождения могут существенно усложнить работу, повысив вероятность допущения ошибки вследствие, например, потери нужной информации.

В итоге отметим, что документ занимает важное место в процессе некоторой деятельности на границе разделяемых функций исполнения деятельности. Поэтому правильно рассматривать документ как инструмент распределения функций между работниками [3]. На представлении документа как инструмента распределения функций между сотрудниками (ролями, рабочими местами) основываются все классические способы автоматизации документооборота и бюрократических систем в целом. При этом понятие документа превращается в понятие электронный документ, а понятие документооборота в **понятие электронного документооборота**.

Основные преимущества **электронного документооборота**:

- а). Полный контроль за перемещением и эволюцией документа, регламентация доступа и способ работы пользователей с различными документами и их отдельными частями.
- б). Уменьшение расходов на управление за счет высвобождения (на 90% и более) людских ресурсов, занятых различными видами обработки бумажных документов, снижение бюрократической волокиты за счет маршрутизации

ванного перемещения документов и жесткого контроля за порядком и сроками прохождения документов.

- в). Быстрое создание новых документов из уже существующих документов.
- г). Поддержка одновременной работы многих пользователей с одним и тем же документом, предотвращение его потери или порчи.
- д). Сокращение времени поиска нужных документов.

Комплексная автоматизация этих функций требует создания **единого информационного пространства предприятия**, в котором сотрудники и руководство могут осуществлять свою деятельность, руководствуясь едиными правилами представления и обработки информации в документном и не документном виде. Для этого в рамках предприятия требуется создать единую **систему управления документами**, включающую следующие возможности:

- а) Удаленной работы, когда члены одного коллектива могут работать в разных комнатах здания или в разных зданиях;
- б) Доступа к информации, когда разные пользователи должны иметь доступ к одним и тем же данным без потерь в производительности и независимо от своего местоположения в сети;
- в) Средств коммуникации, например: электронная почта, факс, печать документов;
- г) Сохранения целостности данных в общей базе данных;
- д) Полнотекстового и реквизитного поиска информации;
- е) Открытость системы, когда пользователи должны иметь доступ к привычным средствам создания документов и к уже существующим документам, созданным в других системах;
- ж) Защищенность информации;

- з) Удобства настройки на конкретные задачи пользователей;
- и) Масштабируемость системы для поддержки роста организаций и защиты вложенных инвестиций и т.д.

Начальным этапом создания такой системы **является построение модели предметной области или другими словами модели документооборота** для конкретного бизнеса и позиционировать (разместить) в ней свое предприятие. Средством автоматизации документооборота, необходимые аспекты которого содержатся в модели, является электронный документооборот. Дадим обобщенное определение этого понятия.

Электронный документооборот - *electronic data interchange, (EDI)* –обмен **электронными документами** между компьютерными программами различных подразделений на предприятии или различных компаний в стандартизированной форме.

Из определения следует, что ключевым понятием электронного документооборота является понятие **электронного документа**, которое рассматривается ниже.

Электронный документ

Общепризнанного понятия электронного документа пока не существует, и приведенные ниже определения в полной мере говорят об этом. Не многие специалисты, по роду своей деятельности имеющие дело с электронными документами отличают электронные документы и документы в электронной форме отображения. К сожалению, приведенные определения ПОНЯТИЯ "ЭЛЕКТРОННЫЙ ДОКУМЕНТ" В ПРИНЯТЫХ ЗАКОНАХ И РАЗРАБАТЫВАЕМЫХ ЗАКОНОПРОЕКТАХ НЕ СООТВЕТСТВУЕТ ПЕРСПЕКТИВЕ МАССОВОГО ПРИМЕНЕНИЯ ЭЛЕКТРОННЫХ ДОКУМЕНТОВ В БЕЗЛЮДНЫХ ТЕХНОЛОГИЯХ ЭЛЕКТРОННОГО ВЗАИМОДЕЙСТВИЯ ЭКОНОМИЧЕСКИХ СУБЪЕКТОВ, (об юридическом аспекте применения электронных документов вопрос не обсуждается).

Новая редакция Федерального закона "Об информации, информатизации и защите информации" [4] (вариант от 2000 года) предназначена для учета новых технологических изменений, про-

шедших за пять лет с момента принятия закона. В проекте утверждается: "ЭЛЕКТРОННЫЙ ДОКУМЕНТ - сведения, представленные в форме набора состояний элементов электронной вычислительной техники (ЭВТ), иных электронных средств обработки, хранения и передачи информации, могущей быть преобразованной в форму, пригодную для однозначного восприятия человеком, и имеющей атрибуты для идентификации документа".

Статья 2 проекта федерального закона "Об электронной цифровой подписи" [4]: - "ЭЛЕКТРОННЫЕ СООБЩЕНИЕ - информация, представленная в форме набора состояний элементов электронной вычислительной техники (ЭВТ), иных электронных средств обработки, хранения и передачи информации, могущей быть преобразована в форму, пригодную для однозначного восприятия человеком, и имеющей атрибуты для идентификации документа". "ЭЛЕКТРОННЫЙ ДОКУМЕНТ - электронное сообщение, имеющее реквизиты для идентификации его как документа".

Статья 4 проекта федерального закона "Об электронном документе" [8]: - "ЭЛЕКТРОННЫЙ ДОКУМЕНТ представляет собой зафиксированную на материальном носителе информацию в виде набора символов, звукозаписи или изображения, предназначенную для передачи во времени и пространстве с использованием средств ВТ и электросвязи в целях хранения и общественного использования". Статья 5 проекта: - "ЭлД должен быть представленным в форме, понятной для восприятия человеком"

Не вдаваясь в тонкости анализа понятия массового применения и безлюдных технологий, примем в качестве базового понятия данного курса – **понятие электронного конструкторского документа (далее ДЭ)**, которое введено в действие с 1.09.2006 г посредством стандарта "ГОСТ 2.051-2006 Электронные документы. Общие положения".

Раздел общих положений этого стандарта включает следующие положения:

1. ДЭ выполняют на стадии разработки изделия и применяют на всех стадиях жизненного цикла изделия. ДЭ получают в результате автоматизированного проектирования (разработки) или преобразования документов, выполненных в бумажной форме, в электронную форму
2. ДЭ имеют два представления – внутреннее и внешнее. Во внутреннем

(подлинном) виде ДЭ существует только в виде записи информации составляющей электронный документ, на электронном носителе и воспринимаемом только программно-техническими средствами. Внешним является представление ДЭ в доступной для визуального восприятия форме. Для получения формы внешнего представления внутреннее представление ДЭ должно быть преобразовано к требуемому виду различными техническими средствами отображения данных (дисплеями, печатающими устройствами и др.)

3. ДЭ состоит из двух частей: содержательной и реквизитной. Содержательная часть состоит из одной или нескольких ИЕ, содержащих необходимую информацию об изделии. Содержательная часть может состоять отдельно или в любом сочетании из текстовой, графической, аудиовизуальной (мультимедийной) информации. Реквизитная часть состоит из структурированного по назначению набора реквизитов и их значений. Номенклатура реквизитов ДЭ – по ГОСТ 2.104. В реквизитную часть ДЭ допускается вводить дополнительные реквизиты, с учетом особенностей применения и обращения ДЭ. Номенклатуру дополнительных реквизитов и правила выполнения и отображения в визуально воспринимаемом виде устанавливает разработчик*
4. ДЭ подразделяют на простые, составные и агрегированные в зависимости от состава и способа организации содержательной части: в простом ДЭ содержательная часть реализована в виде одной ИЕ; в составном ДЭ содержательная часть реализована в виде нескольких ИЕ, связанных друг с другом ссылками, как правило определяемыми применяемым форматом данных; в агрегированном ДЭ содержательная часть реализована в виде нескольких ИЕ, информационно связанных друг с другом;
5. ИЕ в ДЭ могут образовывать сложные иерархические структуры, имеющие совмещенные реквизитные части и общие описания составляющих компонентов. При многократном использовании компонентов допускается применение ссылок*.

6. При использовании в документе ссылок, при выпуске документа все ссылки должны быть заменены на соответствующее им содержание. В составном и агрегированном документах, если его формат требует наличия ссылок, допускается оставлять ссылки при условии, что целостность таких ДЭ обеспечивают программно-технические средствами*.
7. Подлинники, дубликаты и копии ДЭ имеют одинаковую силу с бумажной формой выполнения документов аналогичных наименований. В дубликатах и копиях должны быть сохранены обязательные реквизиты, содержащиеся в подлиннике ДЭ.
8. Аутентичные ДЭ, полученные путем преобразования их форматов, подписанные в установленном порядке ЭЦП, имеют то же наименование документа, что и ДЭ, из которого они получены. Аутентичному ДЭ присваивают дополнительный признак, который записывают в реквизитной части документа. Аутентичный документ должен содержать в реквизитной части указание на исходный ДЭ, из которого он получен*.
9. Твердая копия, изготовленная и подписанная в установленном порядке, может иметь то же наименование документа, что и ДЭ, с которого она получена. В этом случае ответственность за взаимное соответствие исходного ДЭ и его твердой копии в ходе жизненного цикла документов возлагается на разработчика. Твердая копия должна содержать указание на то, что исходным документом является ДЭ.
10. Порядок использования ЭЦП и применяемые программно-технические средства в пределах отдельной организации устанавливаются разработчиком документации в зависимости от наличия конкретного информационного, программного и организационного обеспечения.
11. Порядок управления данными ЭЦП устанавливает разработчик. При обращении ДЭ в корпоративных АС порядок управления данными ЭЦП (например, обмен ключами) устанавливают организации-участники*.

На Рисунок 3 средствами языка UML представлена обобщенная модель ДЭ, удовлетворяющая ключевым положениям стандарта.

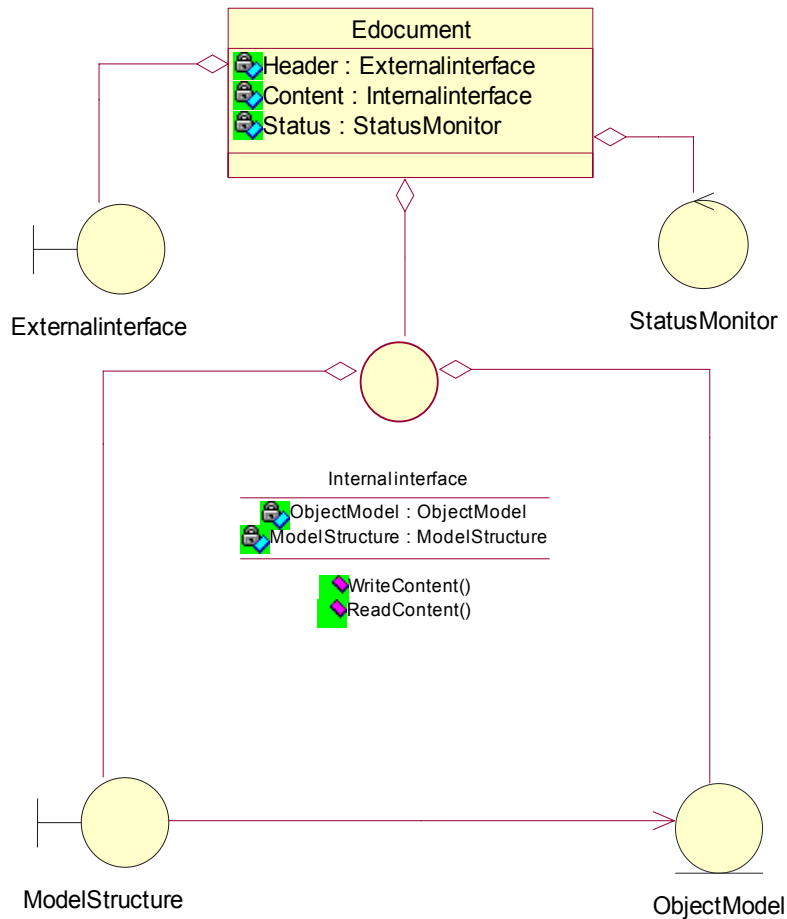


Рисунок 3

Рассмотрев подробнее понятие модели, понятие бизнес – объекта или объекта действительности, понятие бумажного документа, понятие электронного документа и модели бизнес - объекта, сделаем следующие выводы:

- 1) Бумажный документ – может включать информационную модель объекта действительности на бумажном носителе
- 2) Электронный документ (определение 1) – электронная форма информационной модели **объекта действительности**, представляющая собой структурированный набор данных, рассматриваемый в процессе его обработки как единое поименованное целое
- 3) Электронный документ (определение 2) – электронная форма **бумажного документа**, представляющая собой структурированный набор данных, рас-

смаатриваемый в процессе его обработки как единое поименованное целое

Подход к определению понятия информационной системы

Далее рассмотрим один из подходов подход к определению понятия информационной системы (ИС), состоящий из следующих положений:

- 1) ИС есть искусственная система, замещающая (поддерживающая) труд определенного пользователя, которому для принятия решений необходимы определенные информационные услуги.
- 2) Процесс оказания информационных услуг состоит в изготовлении и перемещении в информационном (или физическом) пространстве информационных объектов, роль которых могут выполнять бумажные и электронные документы

Информационная система (определение 1) (по законодательству РФ и www.glossary.ru) - организационно упорядоченная совокупность документов (массивов документов) и информационных технологий, в том числе с использованием средств вычислительной техники и связи, реализующих информационные процессы

Информационная система (определение 2) – *information system* – совокупность элементов (материальных или идеальных), образующих посредством связей некоторую целостность и предоставляющая **информационные услуги**, оперируя при этом **информационными объектами** (документами, информационными моделями)

Информационные технологии – *information technology* – совокупность методов, производственных и программно-технологических средств, объединенных в технологическую цепочку, обеспечивающую сбор, хранение, обработку, вывод и распространение информации для снижения трудоемкости процессов использования информационных ресурсов, повышения надежности и оперативности

Комментарий к данным определениям

Информационная система изначально рассматривается как индифферентная (лат. *indifferens, indifferentis* безразличная, безвредная) конкретным целям пользователей система, аналогичная АТС, библиотеке общего назначения или

справочной службе вокзала, которая предоставляет свои информационные услуги в качестве подсистемы или смежной системы более общей системе: предприятию, городу, отрасли, стране и т.д. В настоящее время, судя по содержанию данных определений, под ИС слишком часто понимают самые разные вещи – от комплекса средств автоматизации (КСА) до автоматизированной системы (АС)

Так в стандартах присутствует четкое определение технического понятия «ИТ-система», **которое часто и требуется использовать вместо ИС**. ГОСТ Р ИСО/МЭК ТО 10000-1-99 определяет **информационно-технологическую систему (IT system)**: *набор информационно-технологических ресурсов, обеспечивающий услуги по одному или нескольким интерфейсам*. Такое определение близко к понятию «комплекс средств автоматизации» в методических указаниях РД 50-680-88 (ГОСТ 34.*.*), где даны основные положения этого комплекса нормативных документов (НД).

В свою очередь, методические указания РД 50-680-88 серии стандартов ГОСТ 34.*.* определяют **автоматизированную систему (АС) следующим образом**: *в процессе функционирования автоматизированная система представляет собой совокупность комплекса средств автоматизации, организационно-методических и технологических документов и специалистов, использующих их в процессе своей профессиональной деятельности. (Из методических указаний РД 50-680-88 серии стандартов ГОСТ 34.*.* на автоматизированные системы (АС))*

ГОСТ 34.003-90, статья 1.1 определяет **автоматизированную систему** следующим образом: *Система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций*. В этом определении организационно-методические и технологические документы входят в понятие информационной технологии, назначение которой выполнение установленных функций, а результат автоматизация труда персонала.

С другой стороны, в последние годы отмечены качественным расширением значения термина «система», отраженным в документах международных комитетов и профессиональных сообществ, ориентированных на ИТ. Наблюдается переход к широкому толкованию, за счет явного включения компонентов других типов (**материалов, методов и др.**). В этой связи растет актуальность более широкого применения термина **«информационно-управляющая система»** и более узкого применения термина **«информационная система»**.

5. Комплексная архитектура предприятия

Комплексная архитектура предприятия. Архитектура информационной системы

Подробно рассмотрев классы систем применительно к проектированию ИС, перейдем к рассмотрению понятия архитектура предприятия

Предприятие в мировой практике и в дисциплине Enterprise Architecture рассматривается как обобщенное понятие, распространяющееся вовсе **не только на промышленные или торговые структуры**. Так предприятием по ISO 15704 считается «одна или более организаций, совместно выполняющих определенную миссию и руководствующихся общими целями и задачами для предоставления некоторого выхода, например, продукта или услуги».

На основе такого понимания область применения понятия предприятия расширена на организации - органы государственного управления, а также на правительства в целом, появилось понятие электронного правительства. В терминологии электронных правительств работа, выполненная госорганами как для конкретного гражданина или рыночной компании, так и для другого правительственного органа **называется услугой**.

Примем следующие определения понятия архитектуры:

Архитектура системы (ANSIIEEE Std 1471 - 2000) – фундаментальная организационная структура системы, воплощенная в ее компонентах, их взаимоотношениях между собой и с окружением, и принципы, управляющие ее построением и эволюцией"

Архитектура информационной системы (определение, используемое в продуктах IBM Rational) - это то, что делает система, на какие части она разделяется, как эти части взаимодействуют, где эти части размещены.

Ранее мы установили, что организацию (предприятие) можно рассматривать как систему, применяя методологию системного анализа. И следуя первому определению архитектуры, определим архитектуру организации, как организационную структуру, воплощенную во взаимодействующих компонентах (подразделениях) и включающую правила построения организации и управления ее развитием.

Определение архитектуры информационной системы также содержит поня-

тие структуры (взаимодействующие части), хотя, прежде всего, говорится о функциях системы «что делает система».

В общем случае, если использовать для анализа понятие внутренняя функциональная структура, по отношению и к организации и к информационной системе, то можно сказать, что архитектура организации и архитектура информационной системы сопоставимы по смыслу понятия. Архитектуры, прежде всего, рассматриваются как обобщенные структуры таких систем как организация, и информационная система. Поэтому для работы с архитектурами могут быть использованы методы структурного подхода, который при работе с архитектурами принимает форму **архитектурного подхода**.

Установив, что архитектура предприятия и архитектура информационной системы по смыслу сопоставимые понятия, поставим вопрос о рассмотрении единой архитектуры предприятия, которая включала бы в себя и архитектуру предприятия (бизнес архитектуру) и архитектуру информационной системы. О такой архитектуре, по сути, являющейся архитектурой компьютеризированного предприятия мы будем говорить как об *информационной архитектуре предприятия*.

Для более формализованного представления понятия архитектуры в среде разработчиков существуют различные схемы, называемые иногда фреймворками. В среде разработчиков существует несколько десятков таких схем. Наиболее известными схемами являются следующие схемы:

1. Схема Захмана (Zachman Framework)
2. NGOSS (New Generation Operation Systems and Software)
3. eTOM (enhanced Telecom Operations Map)
4. SID (Shared Information \ Data model)

Основы **архитектуры предприятия** (Enterprise Architecture) **в форме схемы Захмана** были заложены известными работами Дж. Захмана причем обобщенная схема (framework) архитектуры предприятия стала стандартом де-факто. Основой Enterprise Architecture является **архитектурный подход**, при котором в качестве различных архитектурных представлений единого целого (взаимодействующих компонентов системы) рассматриваются следующие аспекты:

- a) аспекты устройства и потребностей бизнеса (основной деятельности)
- b) прикладные и технические аспекты ИТ - систем
- c) аспекты некоторого мгновенного состояния архитектуры **и процессы разработки и реализации архитектуры**

В конце 90-х годов прошлого века и в начале 21 века прогресс общего понимания того, что такое Enterprise Architecture, привел к появлению базовых международных стандартов методологии архитектуры предприятия (в первую очередь, ISO 15704). Однако на практике наиболее широко и как концепция устройства архитектуры и как интегрирующая схема общего уровня используется обобщенная схема архитектуры предприятия Дж. Захмана. **В последние годы она используется и как верхний интегрирующий уровень CASE-инструментов для моделирования, анализа и конструирования систем.**

Прежде чем перейти к рассмотрению комплексной архитектуры дадим определение следующим понятиям:

- 1) **Аспект** – представление об объекте в рамках ограниченного набора отношений, которые являются существенными для решения конкретной задачи. Развитие представления об объекте осуществляется в рамках определенного аспекта объекта.
- 2) **Предмет** – аспект объекта, отраженный в некоторой концептуальной конструкции (понятийной конструкции). При значительном количестве аспектов, представленных в концептуальной конструкции понятие предмета преобразуется в **предметную область**.
- 3) **Функция** (от лат. functio исполнение, осуществление) – деятельность, обязанность, работа; внешнее проявление свойств какого-либо объекта в данной системе отношений
- 4) **Операционное время** – шкала времени, задаваемая последовательностью операций. Обычно под такой шкалой понимают процесс или структуру процессов
- 5) **УПРАВЛЕНИЕ** – процесс организованных систем различной природы (биологических, социальных, технических), обеспечивающий сохранение их оп-

ределенной структуры, поддержание режима деятельности, реализацию их программ (достижение цели). Рассмотрим, например, **управление знаниями** (knowledge management) формальный процесс, который состоит в оценке организационных процедур, людей и технологий и в создании системы, использующей взаимосвязи между этими компонентами с целью предоставления нужной информации нужным людям в нужное время, что приводит к повышению продуктивности (по определению IDC)

6) **Модель информационной системы (Information System Model)** - строка, относящаяся к точке зрения проектировщика. Это понятие соответствует строке схемы Захмана – проектной модели (или модели аналитика). В принятом сейчас представлении схемы эта строка включает (Таблица 1 Таблица 2):

- а) логическую модель данных (структура бумажного и электронного документооборота)
- б) архитектуру приложений (автоматизированные и ручные функций по обработке документов и связи между ними)
- в) распределение рабочих мест системы (потoki электронных и бумажных документов в местах обработки)

8) **Модель бизнес - объекта**

Описание объектов (изделий или активов), в которых заинтересовано данное предприятие. Этот перечень имеет высокий уровень агрегирования. Соответствующая модель определяет сферу или границы объектов, которые являются значимыми для данного предприятия

9) **Модель бизнес- процесса (Business Process Model)**

Модель фактической деятельности предприятия, которая осуществляется независимо от информационной системы и от каких-либо организационных ограничений. Эта модель может быть представлена как структурированная методо - ориентированная модель, отражающая не только бизнес-процессы, но и их входы и выходы.

10) **Данные (data, ГОСТ 34.321 - 96)** – информация, представленная в фор-

мализованном виде, пригодном для передачи, интерпретации или обработки с участием человека или автоматическими средствами

11) **Модель данных (Data Model)**

Модель данных (в широком смысле) - основные принципы организации и манипулирования данными (различают иерархические, сетевые, реляционные, объектно-ориентированные модели данных). На практике это понятие, чтобы отличать от своего омонима, часто заменяется понятием **«подход»** и говорят, например, о реляционном подходе проектирования логической модели данных

Модель данных (в узком смысле) - **схема базы данных** вместе с процедурами для манипулирования, являющаяся продуктом процесса моделирования данных (в технической литературе различают: концептуальную, логическую, физическую модели данных)

5.1 Концепция устройства комплексной архитектуры предприятия

Наиболее широко в качестве концепции устройства комплексной архитектуры предприятия продолжает использоваться обобщенная «плоская» схема архитектуры Дж. Захмана, в форме – двумерной таблицы, состоящей из шести столбцов (ключевых аспектов) и шести строк (моделей, представлений).

В данном конспекте лекций некоторые положения схемы интерпретируются не так, как это встречается в литературе, хотя в большинстве случаев основной смысл схемы соответствует общепринятой интерпретации. Такое отступление вызвано необходимостью отразить в учебном курсе проблематизацию предметной области на основе анализа модели предметной области и разработки первого представления информационной системы (концепции, образованной основными понятиями, функциональными и нефункциональными требованиями). Причем, решения, закладываемые в концепции (прежде всего, функциональные требования) должны решать полностью или частично выявленные проблемы путем замещения операций в реальном мире операциями компьютерной системы (точнее автоматизированной системы). В таблице о необходимости такого мыслительного действия напоминает пустая между строками «концептуальная модель предмет-

ной области» и «проектная модель».

При использовании схемы в учебных целях появилась необходимость более явно обозначить процессные отношения, заложенные в схеме. Суть предлагаемого изменения выражается следующими положениями:

1. ключевым (системообразующим) аспектом является аспект «цель, мотив»;
2. аспект «операционное время» понимается как отношение, задаваемое на множестве других аспектов каждой отдельной строки (модели представления) по принципу описания «дома, который построил Джек»;
3. остальные аспекты и их элементы рассматриваются так или иначе участвующими в процессах (управляющих, технологических, организационных и т.п.) для достижения ключевых целей;

В качестве примера рассмотрим вторую строку «бизнес-модель предприятия». Бизнес-модель предприятия может быть представлена следующими высказываниями:

1. Целью предприятия является такое состояние, которое описывается параметрами, заложенными в бизнес-план (сегмент рынка, прибыль и т.п.);
2. Для достижения требуемых параметров бизнеса необходимо обеспечить необходимый набор бизнес-процессов, выполняемых под управлением административного процесса;
3. Для достижения результатов каждого процесса необходимы кадровые ресурсы, организованные согласно определенной организационной схеме;
4. При выполнении операций каждого процесса необходима информационная поддержка (автоматизация соответствующей деятельности), которая раскрывается следующими аспектами:
 - а). Документооборота (структуры информационных объектов, чаще всего документов)
 - б). Функциями по обработке документов
 - в). Структуры потоков между рабочими местами, службами, подразделениями, филиалами и т.п.

Таблица 1

Аспекты \ представления	Данные	Функции	Сеть
Потребности и внешняя среда	Объекты и документы (постулированные единицы информации) важные для бизнеса	Выпуск документов Обмен документами (обмен информацией)	Потоки документов между распределенными по миру компаниями и организации
Бизнес - модель предприятия (модель топ-менеджеров)	Документы, составляющие документооборот предприятия и отношения между ними	Информационные модели бизнес-процессов предприятия <i>(Обычно представлены моделями документооборота, включая создание, удаление, изменение состояния, прием \ передачу, исполнение)</i>	Потоки документов между рабочими местами, подразделениями, службами, обеспечивающие управление предприятием
Проектная модель (аналитика)	Гибридная структура бумажного и электронного документооборота	Ручные и автоматизированные функции по обработке документов и связи между ними (создание, удаление, изменение состояния, прием \ передача, исполнение)	Потоки электронных и бумажных документов в местах обработки бумажной и электронной документации
Технологическая модель (разработчика)	Модель данных в контексте выбранной платформы разработки	Алгоритмы обработки ИС (системы приложений) необходимых электронных документов и информационных объектов	Системотехническая архитектура выбранной платформы для поддержки потоков электронных документов
Программистская модель (программист)	Реализация модели данных средствами выбранной платформы	Код программ (программная реализация алгоритмов)	Архитектура приложений (программного обеспечения ИС) и потоки информации
Пользовательская модель (пользователь)	Электронные документы пользователя	Пользовательская обработка документов (формирование, удаление, обмен)	Взаимодействие удаленных пользователей (интеграция), информационный обмен

Таблица 2

Аспекты \ представления	Люди	Операционное время	Цели, мотивы
Потребности и внешняя среда	Кадровый рынок	События и процессы, имеющие значения для бизнеса	Товары, услуги, конкуренты
Бизнес - модель предприятия (топ - менеджеров)	Организационная структура предприятия	Управляющий процесс и комплекс бизнес - процессов	Бизнес - план
Проектная модель (аналитика)	Измененные должностные обязанности и, возможно, организационная структура	Преобразованные бизнес-процессы (замещение ручных операций операциями ИС)	Обновленные бизнес-правила с учетом ИС
Технологическая модель (разработчика)	Пользовательское представление (пользовательский интерфейс)	Сценарий работы ИС (поток управления) для каждого бизнес-процесса	Правила и характеристики работы ИС
Программная модель (программист)	Система доступа к ИС (идентификация, аутентификация), код пользовательского интерфейса	Сценарий (алгоритм) работы программного обеспечения	Правила и характеристики работы программного обеспечения
Пользовательская модель (пользователь)	Умения и ответственность за выполнение операций в компьютерной среде	Сценарий использования ИС	Ожидаемые эффекты от использования ИС

Архитектурное представление - это ячейка таблицы, соответствующая пересечению выбранного столбца и выбранной строки. Например, с точки зрения разработчика (технологическая модель) информационное архитектурное представление (данные) - это проект структуры данных.

Взгляд заинтересованного лица - это совокупность ячеек в пределах одной строки, то есть совокупность архитектурных представлений с определенной точки зрения, соответствующая выбранным аспектам системы. Совокупности ячеек,

включающие все ячейки каждой строки, имеют конкретные названия. Названия отражают точки зрения менеджера, проектировщика, разработчика, программиста, пользователя и обычно называются **точками зрения**.

В схеме **архитектура определяется** как представление конечного продукта (например, информационной системы) **с точки зрения одного** из заинтересованных лиц. Таким образом, существует не одна архитектура, а некое множество архитектур (обычно это менеджер, проектировщик, разработчик, программист, пользователь). В зависимости от взгляда заинтересованного лица и аспекта фокусировки внимания, архитектура системы представляется по-разному. Но все вместе архитектуры дают представление **о комплексной архитектуре компьютеризированного предприятия**.

В практике проектирования архитектурный подход часто предстает в слишком сильно урезанной и даже искаженной форме. **Стандарты предшествующих десятилетий не уделяли достаточного внимания созданию архитектуры** не только предприятия, но даже отдельной автоматизированной системы. Положение изменилось с вводом в действие стандарта ГОСТ Р 34.320-96 (введен в действие с 1.07.2001), а также стандарта ГОСТ Р ИСО/МЭК 15288-2005 (введен в действие 29.12. 2005)

5.2 Основные понятия бизнес – модели предприятия

Согласно рассмотренной выше схеме комплексной архитектуры предприятия, прежде чем приступить к разработке информационной системы, необходимо получить бизнес-модель предприятия (в общем случае - модель предметной области). Бизнес-модель является результатом исследования и формализации бизнес-процессов предприятия как ключевых отношений связывающих отдельные части в целое. Например, бизнес-процессы делопроизводства. Организация работы с документами (будь то платежные или конструкторско-технологические документы) является важной составной частью процессов управления и принятия управлен-

ческих решений, существенно влияющей на оперативность и качество управления.

Процесс принятия управленческого решения состоит из следующих операций: получения информации, переработка информации, анализа, подготовки и принятия решения.

Все эти этапы самым тесным образом связаны с документационным обеспечением процессов управления, проектирования и производства. Если на предприятии отсутствует четкая организация работы с документами, то, как следствие этого, закономерно появление документов низкого качества, как в оформлении, так и в полноте и ценности содержащейся в них информации, увеличение сроков их обработки. Это приводит к ухудшению качества управления и увеличению сроков принятия решений и числу неверных решений. С ростом масштабов предприятия и численности его сотрудников вопрос об эффективности документационного обеспечения управления становится все более актуальным. При этом предприятие вынуждено решать следующие основные проблемы:

- a) Руководство теряет целостную картину происходящего
- b) Структурные подразделения, не имея информации о деятельности друг друга, перестают слаженно осуществлять свою деятельность. Неизбежно падает качество выполнения заказов и способность организации поддерживать внешние контакты
- c) Падение производительности и создает ощущение недостатка в ресурсах: людских, технических, коммуникационных и т.д.;
- d) Расширение штата сотрудников, вкладывание денег в оборудование новых рабочих мест, помещения, коммуникации, обучение новых сотрудников;
- e) Для производственных предприятий увеличение штата может повлечь изменение технологии производства, что потребует дополнительных инвестиций;

- f) В результате может оказаться, что штат увеличен, производительность упала, производство требует инвестиций, соответственно возникает потребность в увеличении оборотного капитала, что может потребовать новых кредитов и уменьшить плановую прибыль.

В итоге предприятие перестает расти интенсивно и дальнейшее расширение происходит чисто экстенсивным путем за счет ранее созданной прибыли.

Почему же сегодня, когда для организации документооборота (в дальнейшем под этим термином мы будем понимать документооборот любых документов: конструкторских, технологических, финансовых, организационных и т.п.) предлагается множество самых различных средств автоматизации, документооборот часто организован плохо, даже на относительно небольших предприятиях? Ответ, независимо от степени автоматизации предприятия и его типа, может быть один - **отсутствие или игнорирование модели организации документооборота** неизбежно приведет к тому, что старые проблемы останутся нерешенными. При этом, если по состоянию делопроизводства в организации был "ручной" хаос, то результатом автоматизации будет "компьютерный" хаос.

Без должной оценки возможностей пользователя, исследования бизнес - процессов его предприятия трудно ожидать эффекта от внедрения как систем документооборота класса docflow так и, тем более, workflow. При этом совсем неважно, как планируется или уже реализован документооборот: вручную или путем автоматизации с помощью мощных западных либо отечественных пакетов - всегда на первом месте должна быть четкая стратегия, направленная на упорядочение бизнес - процессов. Иначе говоря, прежде чем что-то делать, необходимо ответить на вопрос, кому и почему выгодно выполнять те или иные процессы, имеющие место на предприятии. **Проводя в жизнь программу модернизации делопроизводства, важно представлять, уровень достижений предприятия, и какое место ему отводится в модельном пространстве системы документооборота**

6. Моделирование информационных систем

6.1 Общие положения

В разделе 2 при описании признаков объекта проектирования отмечалось, что объектом проектирования является информация об объекте проектирования, информационная модель объекта. В случае проектирования ИС это информационная модель информационной системы. В общем случае модели проектируемых объектов являются знаниями о будущей технической системе и могут быть описаны с помощью следующих теоретических схем: **функциональных, поточных и структурных**. С помощью этих схем обычно раскрываются теоретические (мыслимые) аспекты научно-технические знания. Рассмотрим подробнее эти схемы.

Функциональная схема фиксирует общее представление о технической системе, независимо от способа ее реализации, и является результатом идеализации технической системы на основе принципов определенной технической теории. Функциональные схемы совпадают для целого класса технических систем. Блоки этой схемы фиксируют только те свойства элементов технической системы, ради которых они включены в нее для выполнения общей цели. Каждый элемент в системе выполняет определенную функцию. Совокупность такого рода свойств, рассмотренных обособлено от тех нежелательных свойств, которые привносит с собой элемент в систему, и определяют блоки (или функциональные элементы) таких схем. Как правило, они выражают обобщенные математические операции, а функциональные связи, или отношения, между ними - определенные математические зависимости.

Функциональные схемы, например, в теории электрических цепей представляют собой графическую форму математического описания состояния электрической цепи. Каждому функциональному элементу такой схемы соответствует определенное математическое соотношение, - скажем, между силой тока и напряжением на некотором участке цепи или вполне определенная математическая операция (дифференцирование, интегрирование и т.п.). Порядок расположения и характеристики функциональных элементов адекватны электрической схеме.

В классической технической науке функциональные схемы всегда привязаны к определенному типу физического процесса, т.е. к определенному режиму функционирования технического устройства, и всегда могут быть отождествлены с какой-либо математической схемой или уравнением. Однако, они могут быть и не замкнуты на конкретный математический аппарат. В этом случае они выражаются в виде простой декомпозиции взаимосвязанных функций, направленных на выполнение общей цели, предписанной данной технической системе. С помощью такой функциональной схемы строится алгоритм функционирования системы и выбирается ее конфигурация (внутренняя структура).

Поточная схема, или схема функционирования, описывает процессы, протекающие в технической системе и связывающие ее элементы в единое целое. Блоки таких схем отражают различные действия, выполняемые над естественным процессом элементами технической системы в ходе ее функционирования.

Для каждого вида процесса применяется наиболее адекватный ему математический аппарат, призванный обеспечить эффективный анализ поточной схемы технической системы в данном режиме ее функционирования. Заметим, что для разных режимов функционирования технической системы может быть построено несколько поточных и функциональных схем.

Поточные схемы в общем случае отображают не обязательно только физические процессы (электрические, механические, гидравлические и т.д.), но и химические, если речь идет о теоретических основах химической технологии и вообще любые (например, информационные) процессы. В предельно общем случае поточные схемы отображают не только естественные процессы, но и вообще любые потоки субстанции (вещества, энергии, информации). Причем в частном случае эти процессы могут быть редуцированы к стационарным состояниям, но последние могут рассматриваться как вырожденный частный случай процесса.

Структурная схема технической системы фиксирует те узловые точки, на которые замыкаются потоки (процессы функционирования). Это могут быть единицы оборудования, детали или даже целые технические комплексы, представляющие собой конструктивные элементы различного уровня, входящие в данную техниче-

скую систему, которые могут отличаться по принципу действия, техническому исполнению и ряду других характеристик. Такие элементы обладают, кроме функциональных свойств, свойствами второго порядка, т.е. теми, которые приносят с собой в систему определенным образом реализованные элементы, в том числе и нежелательными (например, усилитель - искажения усиливаемого сигнала). Структурная схема фиксирует конструктивное расположение элементов и связей (т.е. структуру) данной технической системы и уже предполагает определенный способ ее реализации. Такие схемы, однако, сами уже являются результатом некоторой идеализации, отображают структуру технической системы, но не являются ни ее скрупулезным описанием в целях воспроизведения, ни ее техническим проектом, по которому может быть построена такая система. Такая схема пока еще теоретический набросок структуры будущей технической системы, который может помочь разработать ее проект, то есть продуцированный технической теорией исходный пункт для последующей инженерной деятельности, или исходное теоретическое описание, *теоретическая* схема уже существующей технической системы с целью ее теоретического расчета и поиска возможностей для усовершенствования (или разработки на ее основе новой системы). Кроме того, часто эти схемы строятся на основе представлений более специализированных научно-технических дисциплин и решают теоретическими средствами возникшие в них задачи.

Таким образом, в технической теории на материале одной и той же технической системы строится несколько оперативных пространств, которым соответствуют различные теоретические схемы. В каждом таком "пространстве" используются разные абстрактные объекты и средства оперирования с ними, решаются особые задачи. В то же время их четкая адекватность друг другу и структуре реальной технической системы позволяет "транспортировать" полученные решения с одного уровня на другой, а также в сферу инженерной деятельности. Механизмы (методы) взаимодействия этих оперативных пространств могут быть раскрыты в результате методологического анализа функционирования технической теории (например, программной инженерии).

В случае, когда технической системой является информационная система, для разработки схем (оперативных пространств) и их взаимодействий применяют следующие методы программной инженерии: метод структурного программирования (СП) или метод объектно-ориентированного программирования (ООП).

Структурные методы появились раньше и на сегодняшний день имеют большую историю применения и инструментальную поддержку. Объектно - ориентированные методы появились позже, однако, они стремительно завоевывают признание среди разработчиков программных систем и вытесняют из этой области структурные методы.

Несмотря на существенную разницу в принципах и системе понятий, существующую в методах рассмотренных подходов, между ними нет прямого противоречия. Часто объектно-ориентированные методы содержат элементы структурного подхода или структурные методы развиваются в сторону базовых принципов ООП. Например, диаграммы потоков данных, будучи классическим элементом структурного подхода, применяются также и при объектно – ориентированном подходе. Например, диаграммы активности в UML (язык, применяемый в ООП) позволяют моделировать структуру деятельности (бизнес-процессов). Другой пример – классы (диаграммы классов), являются одним из основных понятий ООП и, в тоже время, представляют собой развитие диаграмм «сущность-связь» (ERD), предложенных П.Ченом и широко используемых в структурных методах. Таким образом, оба подхода и СП и ООП обязательно рассматривают функциональное представление системы, структурное представление системы и конечно же потоковое (представление функционирования). Весь вопрос насколько прост и удобен тот или иной метод для решения конкретного класса задач. При решении задач проектирования ИС, как уже отмечалось выше, ООП становится более популярным и уже имеет мощную инструментальную поддержку в виде линейки CASE-средств, которые в свою очередь обеспечивают простой переход от проектных решений по ИС к программным решениям.

6.2 Методы структурного моделирования

Основу методов структурного моделирования составляют принципы структурного подхода, основными из которых являются:

- а). Принцип «разделяй и властвуй» - принцип решения сложных проблем путем их разбиения на множество меньших, независимых задач;
- б). Принцип иерархического упорядочивания – принцип организации составных частей проблемы в иерархические древовидные структуры;
- в). Принцип абстрагирования – выделение существенных аспектов системы и отвлечение от несущественных;
- г). Принцип формализации – необходимость строго методического подхода к решению проблемы;
- д). Принцип непротиворечивости – обоснованность и согласованность элементов;
- е). Принцип независимости данных – модели данных должны быть спроектированы и проанализированы независимо от процессов их логической обработки;
- ж). Использование графических нотаций.

В методах структурного анализа наиболее часто применяют следующие виды графических нотаций:

- DFD (Data Flow Diagrams) – диаграммы потоков данных (ДПД) совместно со словарями данных и спецификациями процессов;
- ERD (Entity-Relationship Diagrams) – диаграммы «сущность-связь»;
- STD (State Transition Diagrams) – диаграммы переходов состояний

Методологической основой структурного моделирования является методология SADT (Structured Analysis and Design Technique). Несмотря на то, что полная методология подразумевает моделирование как функциональной (модели активности), так и информационной составляющей (модели данных), наибольшее распространение получила функциональная часть данной методологии. На основе

функциональной составляющей SADT разработана методология IDEF0, ставшая промышленным для предприятий многих стран мира. Используемые в ней нотации и техника функционального моделирования являются альтернативой DFD, которые преимущественно применяются в методах структурного анализа. Одной из важнейших особенностей методологии SADT является строгая типизация связей между функциями.

6.3 Методы объектно-ориентированного моделирования

Основу методов объектно-ориентированного моделирования составляют принципы объектно-ориентированного подхода, основными из которых являются:

- а). **Принцип онтологизации** системы декларирует представление системы посредством системы классов, отражающих понятийную структуру предметной области в виде моделей (например, модель здания, модель уровневой системы образования и т.п.). Классы определяются атрибутами (свойствами) и операциями (методами). Все множество экземпляров класса (объекты) также определяются через атрибуты и операции класса;
- б). **Принцип декомпозиции** декларирует представление системы (предметной области или ИС) совокупностью взаимодействующих между собой объектов. Объекты являются экземплярами классов и взаимодействуют посредством передачи сообщений;
- в). **Принцип инкапсуляции** – декларирует запрещение любого доступа к атрибутам объекта, кроме как через его операции (методы); в соответствии с этим принципом внутренняя структура объекта скрыта от пользователя, а любое его действие инициируется внешним сообщением, вызывающим выполнение соответствующей операции;
- г). **Принцип наследования** - декларирует создание новых классов от общего к частному; новые классы сохраняют все свойства классов-родителей, а также содержат дополнительные атрибуты и операции,

характеризующие их специфику;

- д). **Принцип полиморфизма** - декларирует возможность работы с объектом без информации о конкретном классе, экземпляром которого он является.

Информационные системы, разработанные на основе объектно - ориентированного подхода обладают следующими ключевыми преимуществами:

- а). возможность повторного использования (объектно - ориентированные системы могут быть легко собраны из ранее разработанных программных компонентов)
- б). расширяемость, (системы будут легко расширяться без какой либо модернизации повторно используемых компонентов)

В процессе становления объектно - ориентированного программирования интерес сместился к объектно - ориентированным методам проектирования и анализа (или описания). Преимущества повторного использования и расширяемости можно рассматривать в рамках анализа и проектирования, а не только программирования. Некоторые специалисты считают, что в общем контексте программной инженерии, чем выше уровень повторного использования, тем лучше. Это приводит к возникновению важных вопросов. Должно ли объектно - ориентированное проектирование реализовываться в объектно – ориентированном языке? Должны ли методы проектирования быть связаны с определенными языками? Одним из ответов на эти вопросы явилась разработка языка моделирования UML (Unified Modeling Language – Унифицированный Язык Моделирования). В методах объектно-ориентированного анализа в настоящее время международным стандартом (версия 1.4 стандарт ISO) и стандартом «де-факто» стал язык UML. В ходе принятия проектных решений, на практических занятиях, являющихся составной частью данного лекционного курса, рассматриваются вопросы более детально применения элементов языка UML и методов моделирования с его помощью.

7. Архитектура информационной системы

Нормативное понятие архитектуры информационной системы по ГОСТ Р 34.320-96. Архитектура клиент – сервер

Нормативное понятие архитектуры информационной системы (по ГОСТ Р 34.320-96)

В стандарте ГОСТ Р 15288-2005 не только особо выделен процесс создания архитектуры целевой системы (информационной системы или автоматизированной системы), но появились прямые указания на то, что архитектурные продукты бизнес архитектуры и логической архитектуры не являются принадлежностью только стадий создания ("концепция", "разработка") системы. Эти архитектуры **должны сопровождаться и развиваться на протяжении всего жизненного цикла системы**, отражать все изменения потребностей заинтересованных лиц, служить для принятия решений во всех важных точках, связанных с принятием решений о дальнейшем развитии системы.

Одно из популярных в среде разработчиков ИС формальных определений архитектуры приведено в стандарте ANSI \ IEEE Std 1471 - 2000 Института инженеров-электриков и электронщиков, который предоставляет метамодель для определения архитектуры. Этот стандарт определяет такие абстрактные элементы архитектуры, как представления, системы, среды, обоснования, заинтересованные стороны и т.д. в соответствии со схемой, показанной Рисунок 4



Рисунок 4 Рамочная модель разработки архитектуры по IEEE 1471

В соответствии с этим представлением система обладает некоторой архитектурой, которая может быть определенным образом описана с различных точек зрения в зависимости от интереса тех людей (заинтересованных лиц), которые рассматривают архитектуру системы. Каждой точке зрения на архитектуру системы соответствует определенное представление, основу которого составляет некоторый набор моделей. Однако этот стандарт не определяет структуру собственно архитектуры предприятия. Например, говорится о том, что необходимо иметь различные представления архитектуры, но при этом не указывается, какие это должны быть представления.

Рассматривая архитектуру ИС можно рассмотреть различные аспекты понятия архитектуры ИС. В частности, можно выделять такие подмножества, как системная архитектура (архитектура систем – System Architecture) и программная архитектура (архитектура программного обеспечения – Software Architecture). Напомним, что мы уже отмечали неоднозначность трактовки терминов. На практике, в зависимости от контекста, термин "системная архитектура" может относиться либо к архитектуре ИС предприятия (в дополнение к бизнес-архитектуре) или даже в

еще более узком смысле к технологической инфраструктуре информационной системы, либо – к архитектуре сложного продукта или семейства продуктов, выпускаемых предприятием.

В последнем случае понятие разработки системной архитектуры близко по смыслу понятию - системное проектирование, определение которого давалось выше.

Методика описания и проектирования архитектуры отдельных прикладных систем имеет много общего с подходами к описанию архитектуры предприятия в целом, тем не менее, архитектура программных систем является отдельной областью знаний, которой посвящено большое количество соответствующей литературы. Под "**программной архитектурой**", опять же в зависимости от контекста, может пониматься как архитектура взаимодействия приложений в рамках информационной системы предприятия (т.е. архитектура приложений), так и архитектура программных модулей или архитектура взаимодействия различных классов в рамках одного приложения. Каждая из отмеченных архитектур, в свою очередь, может рассматриваться с тем или иным уровнем детализации и под определенным углом зрения. Так, для программной архитектуры традиционными являются следующие перспективы или уровни описания архитектуры:

- **концептуальная архитектура** определяет компоненты системы и их назначения, обычно в неформальном виде. Это представление часто используется для обсуждения с нетехническими специалистами, такими как руководство, бизнес-менеджеры и конечные пользователи функциональных характеристик системы (что система должна уметь делать, в основном, с точки зрения конечного пользователя);
- **логическая архитектура** выделяет, прежде всего, вопросы взаимодействия компонент системы, интерфейсы и используемые протоколы. Это представление позволяет эффективно организовать параллельную разработку;

- **физическая реализация**, которая описывает привязку к конкретным узлам размещения, типам оборудования, характеристикам окружения, таким как, например, используемые операционные системы и т.п.

Рассмотренные выше положения ANSI \ IEEE Std 1471 – 2000 задают лишь рамочную модель разработки архитектуры, но тем не менее являются полезными для понимания основ архитектурного подхода вообще. Ниже рассмотрены положения стандарта ГОСТ 34.320-96, который интересен тем, что содержит конкретные знания об архитектуре ИС и, к тому же, является стандартом РФ. Эти факторы явились определяющими для рассмотрения положений этого стандарта.

В ГОСТ 34.320-96 дано описание архитектуры информационной системы, которая состоит из трех уровней: внешняя схема, внутренняя схема и уровень концептуальной схемы, информационной базы и информационного процессора. Ниже приведены (согласно ГОСТ 34.320-96) определения этих понятий.

Внешняя схема: Определение форм внешнего представления для возможных совокупностей предложений в пределах представления конкретного пользователя, а также аспектов манипулирования этими формами.

Внутренняя схема: Определение форм внутреннего представления в компьютере совокупностей предложений концептуальной схемы и информационной базы, а также аспектов манипулирования этими формами

Концептуальная схема: непротиворечивая совокупность предложений, выражающих необходимые высказывания, относящиеся к проблемной области.

Информационная база: совокупность предложений, выражающих высказывания, отличные от необходимых высказываний, согласующиеся друг с другом и с концептуальной схемой, а также истинные в некотором пространстве сущностей

Информационный процессор: процессор, который в ответ на команду выполняет действие над концептуальной схемой и/или информационной базой.

Прежде чем рассматривать следующие понятия архитектуры, вспомним определение формальной системы и формальной модели (22), которое вводилось при определении понятия системы, и сравним их с определением понятий концептуальной схемы и информационной базы. Из сравнения следует, что и

концептуальную схему и информационную базу можно **трактовать как модели предметной области, отображаемые в информационной системе.**

На Рисунок 5 приведена «трехсхемная» архитектура информационной системы, выполненная согласно описанию архитектуры по ГОСТ 34.320-96.

Информационная система включает, по меньшей мере, следующие типы интерфейсов:

1. Внешний интерфейс — это фактически интерфейс между пользователем в среде и информационной системой. Описывается во внешних схемах и по отношению к пользователям информационной системы, обрабатывает внешние формы представления, удобные для конкретного пользователя.
2. Внутренний интерфейс — это интерфейс между информационной системой и средствами управления данными в компьютере. Описывается внутренней схемой и связан с аспектами физического представления информации, эффективностью работы программ и механизмами эффективного доступа к хранимым данным, управление параллельным использованием, восстановлением после сбоев и т. д.

В информационную систему поступают сообщения (например, от пользователя), содержащие информацию, которая добавляется в информационную базу. Информационный процессор, управляемый правилами, описанными в концептуальной схеме и, возможно, другими предложениями, уже присутствующими в информационной базе, или вставит эту новую информацию, или проигнорирует сообщение, выдав соответствующее сообщение, извещающее о результате.

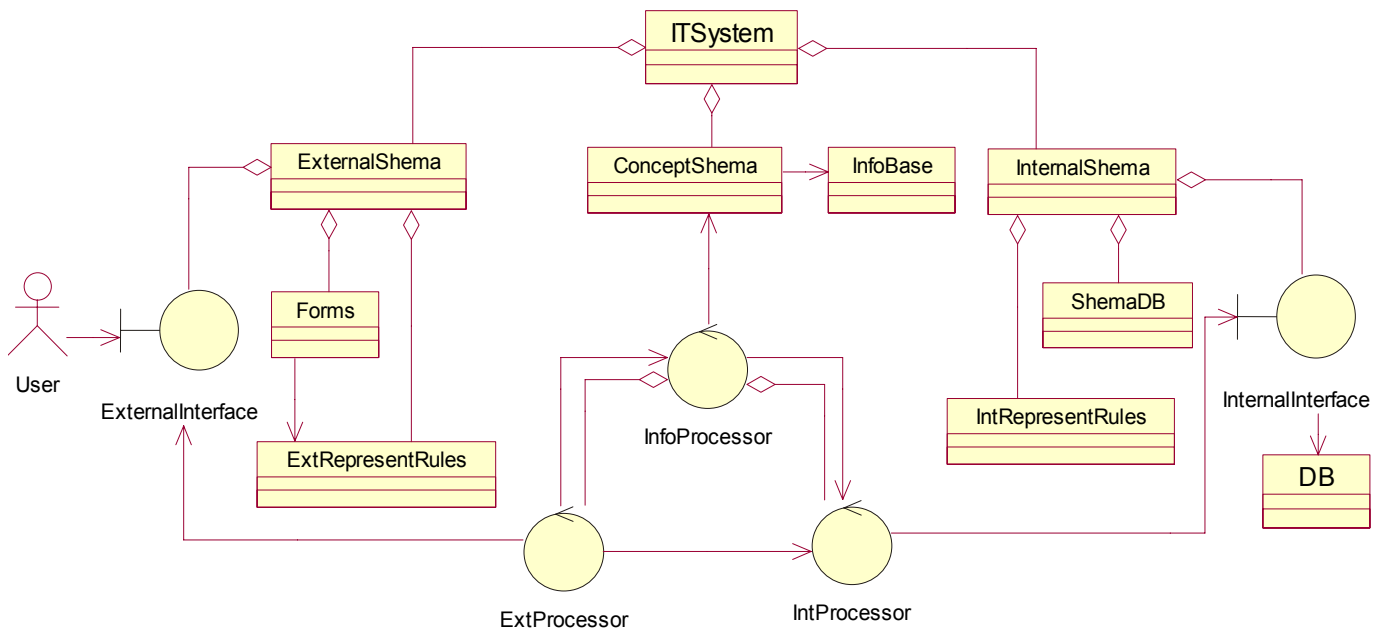


Рисунок 5

Прикладной процесс (внешний процессор), обрабатывающий сообщения пользователя, имеет дело с определенным внешним представлением данных (строками символов), составляющих его внешнюю базу данных и содержащую соответствующую информацию. Это конкретное внешнее представление пользователя описано во внешней схеме, соответствующей данному прикладному процессу. Такая внешняя база данных является виртуальной, отображаемой на информационную базу (или соответствующую ее часть). А это, в свою очередь, предполагает, что информационная система должна обрабатывать:

- «интеграцию» действий различных пользователей;
- отображение их внешних представлений в общее (концептуальное) представление, известное информационной системе.

Представления пользователей можно объединить в концептуальную подсхему. Концептуальная схема в информационной системе представляет собой «объединение» этих различных концептуальных подсхем. На концептуальном уровне формы представления несущественны.

На внешнем уровне определяются формы представления, удобные для поль-

зователя. Они описываются во внешних схемах. Каждое представление пользователя (концептуальная подсхема) отображается в одну или более внешних схем, определяющих соответствующие формы представления, каждая из которых описывает внешнюю базу данных, существующую в пределах представления этого пользователя, хотя и в виртуальной форме. Отображение (преобразование) внешней базы, представляемое посредством совокупности форм, на информационную базу выполняется внешним процессором по правилам отображения, содержащимися во внешних схемах.

В случае, когда внешнее представление является объединением нескольких представлений, полученная внешняя схема будет охватывать несколько отдельных внешних схем и описывать общую базу данных во внешней, но объединенной форме. Функция разложения внешней схемы на подмножества поддерживается и управляется внешним процессором

Сама информационная база по существу является виртуальной. Информация представлена в вычислительной системе во внутренних формах физических данных (записи, сегменты, поля и т. д.) во внутренней базе данных. Эти формы описаны во внутренней схеме. Отображение (преобразование), выполняемое внутренним процессором. Правила отображения для него также описаны во внутренней схеме. К самой внутренней базе данных доступ осуществляется с помощью средства управления внешней памятью компьютерной системы.

Согласно трехсхемной архитектуре и внутренняя, и внешняя схемы, и процессоры могут иметь множество уровней. Точно также внутренняя база данных может быть реализована как семейство внутренних баз данных, каждая из которых «хранит» часть информационной базы. Такие базы данных пересекаются.

Фактически внешние базы данных отображаются в физические базы данных. Несколько внешних баз данных могут отображаться в одну физическую базу данных; одна внешняя база данных может отображаться в несколько физических баз данных; возможно другое сочетание. Возможны и распределенные сети.

Физическая база данных определяется во внутренней схеме. Преобразование из внешней во внутреннюю форму главным образом осуществляется внутренним

процессором.

В распределенных базах данных взаимосвязь между внешней и внутренней базами данных может быть описана в схеме распределения, которая может быть объединенной частью внешней схемы (внешних схем), взаимодействующей с внутренней схемой (внутренними схемами).

Внешний процессор устанавливает связь непосредственно с пользователями и координирует их потоки информации. Внутренние схемы описывают внутреннее физическое представление информации. Отображение между внешними и внутренними формами выполняется, главным образом, внутренним(ми) процессором(ами). Поэтому внешний(ние) процессор(ры) устанавливает(ют) связь с внутренним(ними) процессором(ами). Отображение внешних схем во внутренние схемы должно сохранять смысл информации в соответствии с концептуальной схемой

Концептуальная схема рассматривается как описание необходимых высказываний для проблемной области и поэтому определяет, что описывается в информационной базе, а не как это описывается. Концептуальная схема управляет семантическим (смысловым) значением всех представлений — то есть определяет набор проверяющих, генерирующих и выводящих процедур, определенных на концептуальном уровне информационной системы. Но они не дают описания промежуточного состояния в процессе преобразования из внешней формы во внутреннюю форму

Задача обработки правил (высказываний концептуальной схемы), выполняемая информационным процессором, может быть реализована в виде набора процедур. Они не обязательно выполняются одним специальным (информационным) процессором, отличным от внешнего и внутреннего процессоров. В частности, в системах распределенных баз данных эти процедуры могут распределяться по соответствующим внешним и внутренним процессорам.

Примеры концептуальной схемы и информационной базы приведены в приложениях: Приложение 2. Пример описания предметной области (по ГОСТ 34.320-96) и Приложение 3. Концептуальная схема для подхода сущность-атрибут-связь (необходимые высказывания по ГОСТ 34.320-96)

Архитектурные решения при построении клиент – серверных систем

В настоящее время можно выделить три архитектурных решения, использующихся при построении клиент- серверных систем:

- а). удаленный доступ к данным
- б). хранимые процедуры
- в). трехзвенные системы

В системах с удаленным доступом к данным приложение располагается на клиентской машине и осуществляет обмен данными с СУБД (Рисунок 6). Такие системы создают неоправданно высокую нагрузку на вычислительную сеть, плохо масштабируются и чрезвычайно неудобны в обслуживании.



Рисунок 6 Удаленный доступ к данным

В системах с использованием механизма хранимых процедур приложение реализовано средствами самой СУБД (Рисунок 7) и, вследствие этого, страдает отсутствием переносимости, ограничением возможностей и производительности.

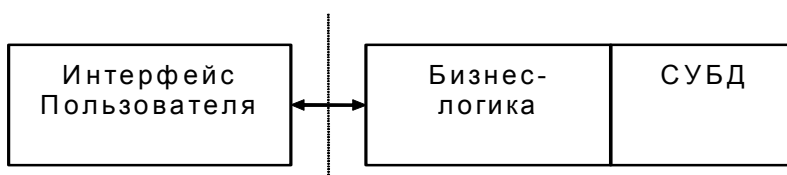


Рисунок 7 Хранимые процедуры

Трехзвенная архитектура, показанная Рисунок 8, поддерживающая манипулирование объектами, решает многие из возникающих проблем, которые связаны с разработкой высокопроизводительных, надежных приложений для архитектуры клиент-сервер.

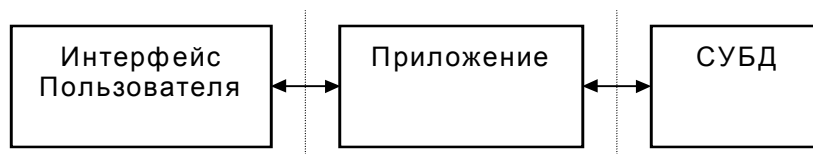


Рисунок 8 Трехзвенная архитектура

В трехзвенной архитектуре, приложение располагается в промежутке между СУБД и клиентом на так называемом сервере приложения. Непосредственными следствиями внедрения в систему промежуточного звена обработки информации являются:

- а). Переносимость
- б). Модульность
- в). Централизация управления
- г). Надежность
- д). Простота интеграции
- е). Производительность

Само приложение может быть, в свою очередь, разделено на части. Одна из этих частей отвечает за решение прикладной задачи, а вторая за взаимодействие этой задачи с остальными компонентами системы. Это программное обеспечение принято называть промежуточным, поскольку оно располагается в промежутке между прикладной программой пользователя и системным программным обеспечением, включающим в себя операционную систему и сетевое программное обеспечение.

Промежуточное программное обеспечение предоставляет разработчику приложения стандартный API доступ к ресурсам системы, позволяя создавать распределенные переносимые приложения, работающие в гетерогенных вычислительных сетях. Таким образом, промежуточное программное обеспечение помогает сделать разрозненные информационные ресурсы предприятия и его окружения доступными для оперативной обработки. Поддерживая широкий спектр

различных стандартов распределенной обработки, промежуточное программное обеспечение позволяет соединить между собой разрозненные информационные системы в единый комплекс эффективной обработки и хранения информации.

К функциям промежуточного программного обеспечения относятся также: предоставление разработчику унифицированного интерфейса к разнородным СУБД, надежная доставка запросов и ответов в гетерогенной сети, балансировка загрузки, централизация управления и многое другое.

Разработчики прикладного программного обеспечения и системные интеграторы могут рассматривать компоненты промежуточного программного обеспечения как отдельные блоки для построения сложных информационных систем для эффективной обработки распределенных данных. Такие системы позволяют пользователю получать необходимую информацию, в определенное время, в определенном месте и в требуемом ему формате.

8. Модели жизненного цикла информационных систем

Рассмотрев комплексную архитектуру и архитектуру информационной системы, рассмотрим модели жизненных циклов (ЖЦ) как процессов, создания, функционирования, модернизации и замены информационных систем как изделий. Описания моделей приводятся по ГОСТ Р ИСО/МЭК ТО 15271-2002.

Существуют множество моделей жизненного цикла, но три из них - фундаментальные: каскадная, инкрементная, эволюционная. Каждая из указанных моделей может быть использована самостоятельно или скомбинирована с другими для создания гибридной модели жизненного цикла. При этом конкретную модель жизненного цикла следует выбирать так, чтобы процессы, работы и задачи из ГОСТ Р ИСО \ МЭК 12207-99 были связаны между собой и определены их взаимосвязи с предшествующими процессами, работами (видами деятельности), задачами (заданиями). В настоящем документе с точки зрения достоинств (выгод) и недостатков (аргументов против их применения) описаны все три фундаментальных модели жизненного цикла программных средств. Эти достоинства и недостатки должны быть учтены при выборе модели жизненного цикла проекта.

Перед рассмотрением моделей жизненного цикла рассмотрим понятия системы и компьютерной системы по материалам ГОСТ Р ИСО \МЭК 15271-2002. На Рисунок 9 приведена схема понятия системы, применяемого в стандарте. Как следует из схемы, под системой понимается комплекс ручных и автоматизированных процессов. Причем, автоматизированные процессы составляют часть ручных процессов, то есть допускается, что в системе не все процессы автоматизированы. Автоматизированные процессы посредством программных и технических средств замещают часть ручных операций, выполняемых сотрудниками, а оставшаяся часть преобразуется и становится средством автоматизированных процессов. Технические и программные средства автоматизированных процессов образуют компьютерную систему и являются основой для преобразования бизнес-процессов с целью решения проблем бизнеса.

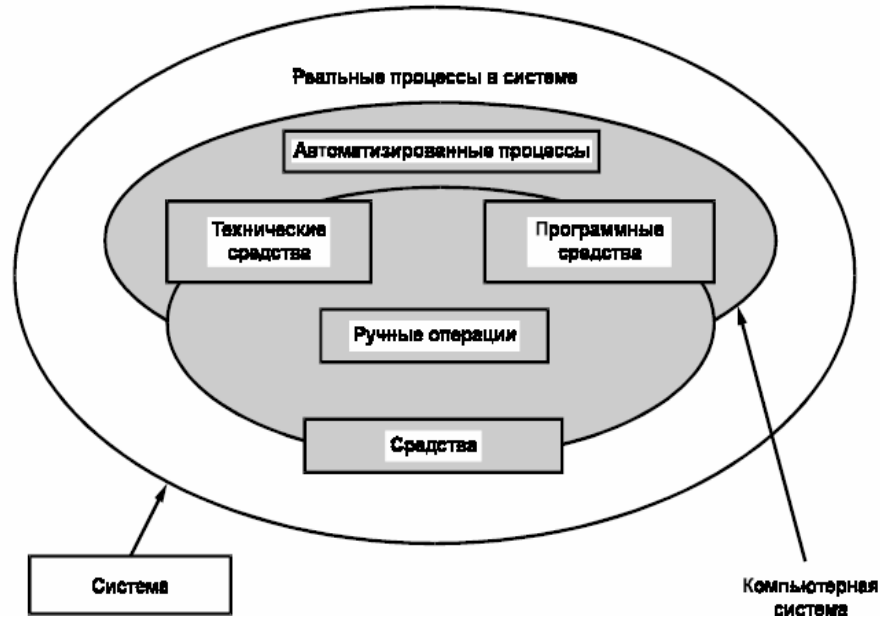


Рисунок 9

8.1 Каскадная модель

Каскадная модель жизненного цикла реализует принцип однократного выполнения каждого из следующих видов деятельности:

1. Установление потребностей пользователя
2. Определение требований
3. Проектирование системы
4. Изготовление системы
5. Испытание
6. Корректировка
7. Поставка или использование

При применении такого принципа разработки каждого программного объекта соответствующие работы и задачи процесса разработки обычно выполняют последовательно (Рисунок 10). Однако они могут быть частично выполнены параллельно в случаях перекрытия последовательных работ.

Когда несколько программных объектов разрабатывают одновременно, для всех этих объектов работы – задачи процесса разработки могут быть выполнены параллельно. Процессы сопровождения и эксплуатации обычно реализуют после

процесса разработки. Процессы заказа и поставки, а также вспомогательные и организационные процессы обычно выполняют параллельно с процессом разработки.

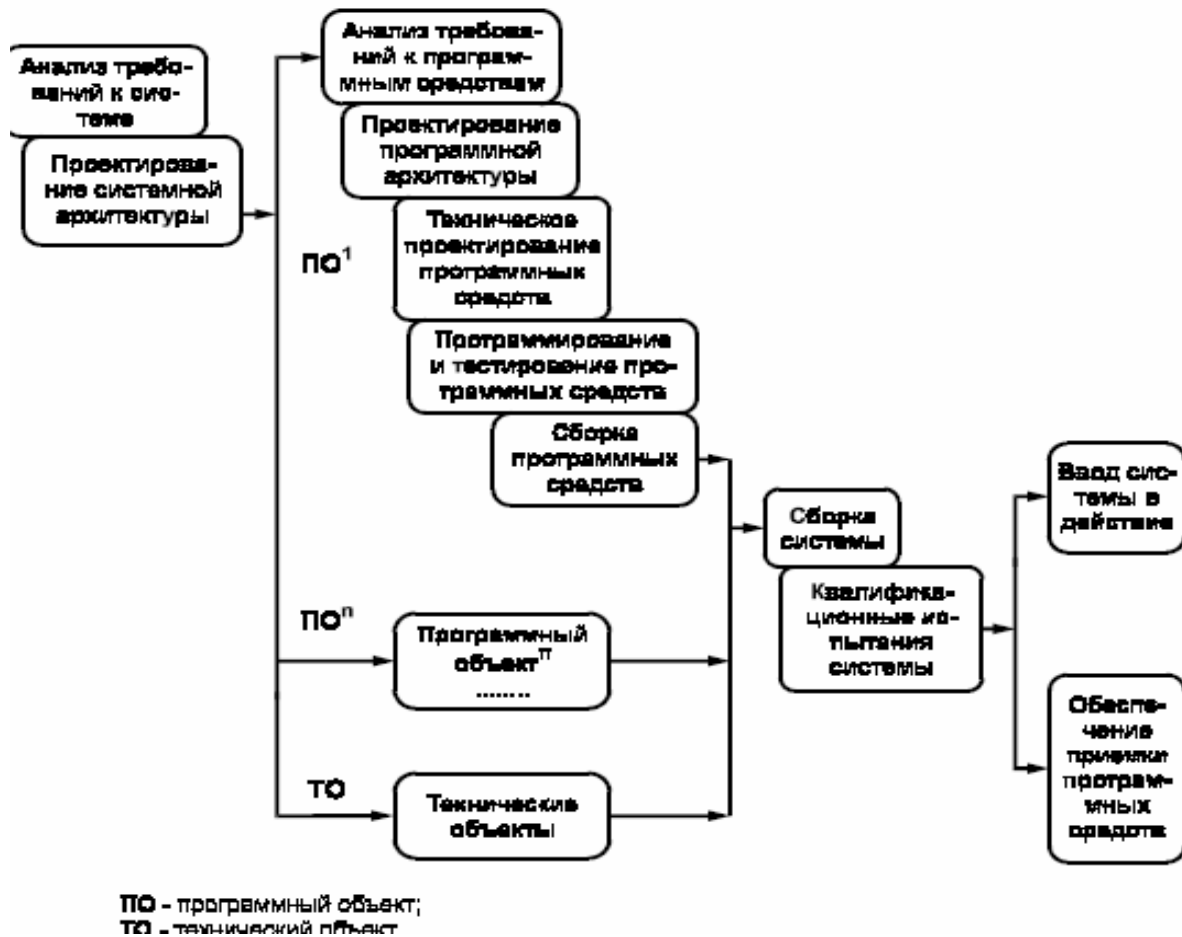


Рисунок 10

Недостатки

Данной модели присущи следующие недостатки, которые необходимо учитывать при оценке возможности ее применения:

1. Требования к объектам определены недостаточно четко
2. Система обычно слишком велика, чтобы все работы по ее созданию выполнять однократно
3. Предполагаемые скорые изменения в технологии работ

4. Ограниченность ресурсов, например средств или персонала
5. Промежуточный продукт может быть непригоден для использования

Преимущества

1. Однократное представление всех возможных характеристик системы
2. Необходимость только единственной фазы перехода от старой системы к новой

8.2 Инкрементная модель

Инкрементная модель жизненного цикла, называемая также запланированным усовершенствованием продукта (Рисунок 11), начинается с выдачи набора требований и реализует разработку последовательности конструкций. Первая конструкция содержит часть требований, в последующую конструкцию добавляют дополнительные требования и так далее до тех пор, пока не будет закончено создание системы. Для каждой конструкции выполняют необходимые процессы, работы и задачи, например, анализ требований и создание архитектуры могут быть выполнены сразу, в то время как разработку технического проекта программного средства, его программирование и тестирование, сборку программных средств и их квалификационные испытания выполняют при создании каждой из последующих конструкций.

В данной модели при разработке каждой конструкции работы и задачи процесса разработки выполняют последовательно или частично параллельно с перекрытием. При частично одновременной разработке последовательных конструкций работы и задачи процесса разработки могут быть выполнены параллельно для ряда конструкций

Недостатки

1. Требования к объектам определены недостаточно четко
2. Предусмотрены сразу все возможные системы
3. Предполагаемые скорые изменения в технологии работ
4. Возможные текущие изменения требований к системе
5. Привлечение ресурсов (средств или персонала) на длительный период ог-

раничено

Преимущества

1. Необходимость изначального использования характеристик системы
2. Пригодность для использования промежуточного продукта
3. Естественное разделение системы на наращиваемые компоненты (инкременты)
4. Возможности наращивания привлекаемого персонала и средств

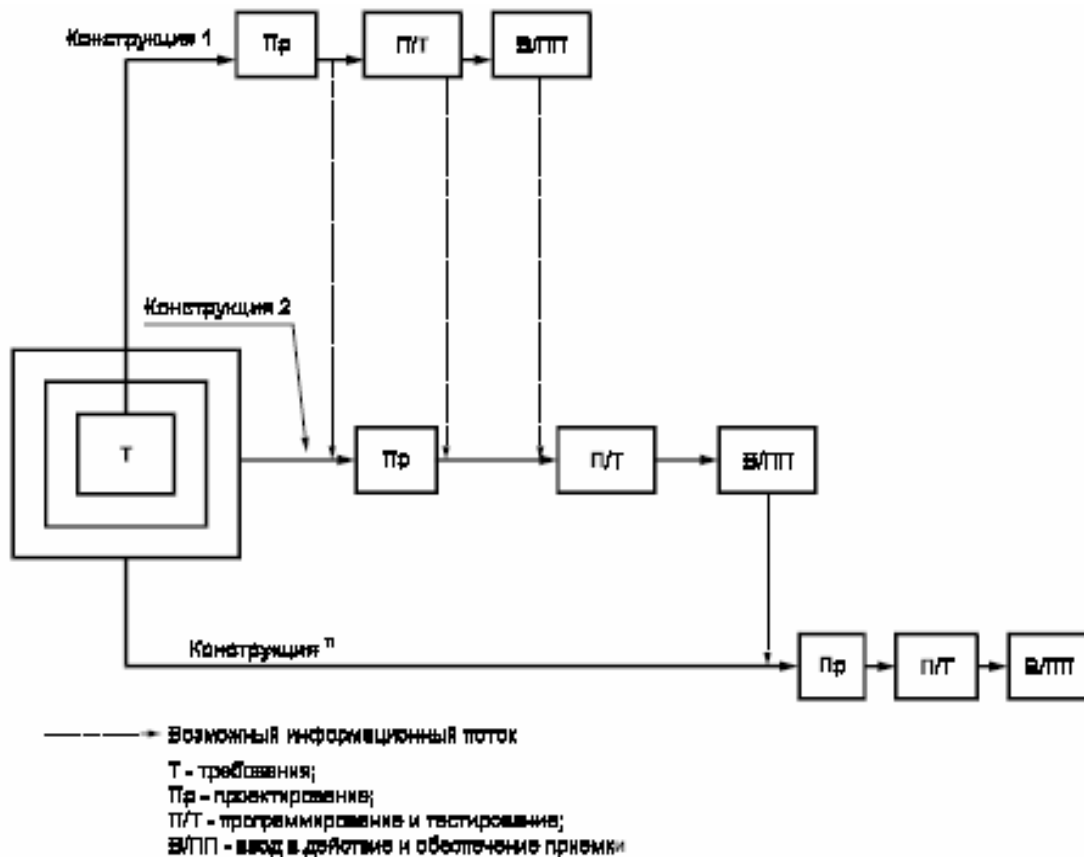


Рисунок 11

8.3 Эволюционная модель

В эволюционной модели жизненного цикла систему также разрабатывают в виде отдельных конструкций, но в отличие от инкрементной модели требования изначально не могут быть полностью осознаны и установлены. В данной модели

требования устанавливаются частично и уточняются в каждой последующей конструкции (Рисунок 12)

При таком методе для каждой конструкции работы и задачи процесса разработки выполняются последовательно или параллельно с частичным перекрытием.

Работы и задачи процесса разработки обычно выполняются многократно в той же последовательности для всех конструкций. Процессы эксплуатации и сопровождения могут быть реализованы параллельно с процессом разработки. Процессы заказа и поставки, а также вспомогательные и организационные процессы обычно выполняются параллельно с процессом разработки.

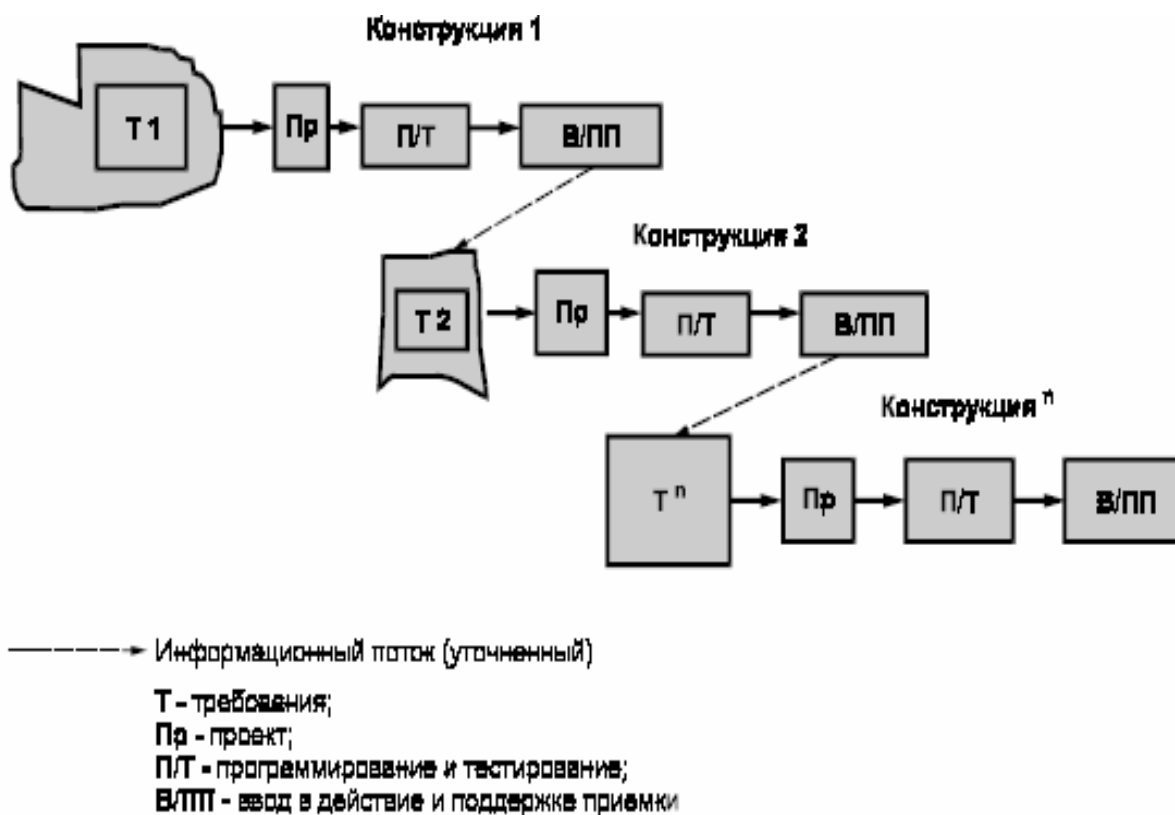


Рисунок 12

Недостатки

1. Все возможности системы predetermined изначально
2. Ограниченные возможности длительного привлечения ресурсов (средств или персонала)

Преимущества

1. Изначальное определение возможностей системы
2. пригодность для использования промежуточного продукта
3. Естественное разделение системы на наращиваемые компоненты (инкременты)
4. Привлечение персонала и средств по мере необходимости
5. Необходимая обратная связь с пользователем для полного понимания требований
6. Упрощение надзора за изменением технологии

9. Ключевые концепции унифицированного процесса

Ключевые концепции унифицированного процесса: управляемый вариантами использования, архитектурно-ориентированный, итеративный и инкрементный

Введем необходимые понятия.

Артефакт - это общее название для любых видов информации, создаваемой, изменяемой, или используемой сотрудниками при создании системы. Наиболее интересный тип артефактов, используемых в унифицированном процессе это модели

Модель – это абстракция (знания), описывающая моделируемую систему с определенной точки зрения и на определенном уровне абстрагирования. Под точкой зрения будем понимать представление аналитика требований или проектировщика.

Процесс разработки в бизнесе разработки программного обеспечения – набор деятельности, необходимых для переработки требований заказчика в согласованный набор артефактов, представляющих собой программное обеспечение, а позднее для переработки изменений в этих требованиях в новые версии программного обеспечения.

Понятие «процесс» трактуется по-разному и употребляется в различных смыслах для различных контекстов, например, бизнес-процесс, процесс разработки, программный процесс. В контексте унифицированного процесса (RUP), **понятие процесс рассматривается как шаблон (комплекс знаний)**, который может быть неоднократно использован для создания его экземпляров. Такое понимание сравнимо с пониманием класса и объекта в объектно-ориентированном проектировании. Экземпляр процесса – **синоним проекта**.

Проблемы качественного программного обеспечения (ПО) сводятся к проблемам разработчиков, вынужденных преодолевать в ходе разработки множество преград. **Результаты такой работы обычно не прогнозируемы по качеству и что не маловажно по срокам**. Поэтому необходим подход к организации процесса разработки, который бы объединил все множество аспектов разработки программ. Наличие хорошо определенного и хорошо управляемого процесса дает гарантию качественных проектов. Такой процесс должен отвечать на следующие

вопросы:

1. Как **управлять** деятельностью команды и **проектом** в целом
2. Какие поставить (определить) **задачи** для отдельного разработчика и команды в целом
3. Какой перечень **артефактов** следует разработать
4. Какие необходимы **критерии** для отслеживания и измерения **продуктов** и функционирования проекта

Унифицированный процесс в контексте приведенных выше требований определяется как **управляемый вариантами использования, архитектурно-ориентированный, итеративный и инкрементный**.

В следующем разделе рассмотрим основные составляющие унифицированного процесса более подробно.

9.1 Унифицированный процесс – управляемый вариантами использования

Вариант использования – это часть функциональности системы, необходимая для получения пользователем значимого и измеримого результата.

Сумма всех вариантов использования составляет модель вариантов использования, которая описывает полную функциональность системы. Эта модель заменяет традиционное описание функций системы (функциональной структуры). Описание варианта использования отвечает на вопрос, что **система может сделать для каждого пользователя**. Такое понимание варианта использования очень важное положение методологии унифицированного процесса и побуждает разработчиков мыслить в понятиях результата для пользователя, а не в понятиях функций, которые необходимо иметь в системе. Описание варианта использования моделирующего функциональность бизнес-процесса приведено в Приложение 1. Пример текстового описания варианта использования.

Определение варианта использования содержит понятие функции (функциональной возможности). Рассмотрим происхождение этого понятия для технических систем. В контексте данного курса, функции уже рассматривались как аспект информационной части схемы архитектуры предприятия, и как комплекс решений,

замещающий часть операций бизнес-процесса, операциями разрабатываемой информационной системы. Аналогично по смыслу, но в других терминах, определяется понятие технической функции для изделий (технических систем). Техническая функция определяется как основное действие системы (выражаемое в назначении), реализующее определенную потребность. То есть техническая функция определяется через потребность, которую она удовлетворяет. Различие между технической функцией и технически реализуемой потребностью, состоит в том, что последняя характеризует человека, у которого есть желание удовлетворить потребность, а техническая функция характеризует техническую систему (изделие), с помощью которой данная функция удовлетворяется. Множество технических функций в процессе развития техники непрерывно расширяется и структурно меняется. Такие изменения происходят под воздействием процессов (закономерностей) возникновения и развития потребностей. В данном курсе понятие потребностей рассматривается в форме высказываний - требований (функциональных и нефункциональных) к информационной системе (раздел отчета: “ проблемы предметной области и концепция ИС”)

Вариант использования не только является выражением функциональных требований, структурой состоящей из функций, но и структурой, которая призвана направлять проектирование, реализацию и тестирование, то есть она управляет процессом разработки и выступает во второй своей форме – в форме **управляющего разработкой процесса**.

Руководствуясь знанием (моделями) вариантов использования разработчик создает серию моделей проектирования и реализации, которые воплощают варианты использования в работу программного обеспечения

Тестеры испытывают реализацию для того, чтобы гарантировать, что разработанные программные компоненты правильно выполняют варианты использования. Процесс разработки, **управляемый вариантами использования** означает, что в процессе разработки выполняются серии **рабочих процессов** (отрабатываются управляющие воздействия), **порожденные вариантами использования**

Поскольку варианты использования управляют процессом разработки, то они

разрабатываются в паре с архитектурой системы (?). Таким образом, варианты использования управляют архитектурой, а архитектура оказывает влияние на варианты использования. Причем, и варианты использования и архитектура развиваются в процессе жизненного цикла

9.2 Унифицированный процесс - ориентирован на архитектуру

Архитектура - это представление всего проекта с выделением ключевых составляющих и затухивание деталей. Архитектура вырастает из требований к результату, в том виде, как их понимает пользователь и другие заинтересованные лица.

А требования, как отмечалось ранее, отражаются в вариантах использования. Как связаны варианты использования и архитектура?

Начнем с того, что постулируем, что каждый продукт имеет функции и форму, причем одно без другого не существует. В нашем случае функции, как мы ранее отмечали, соответствуют вариантам использования, а форма – архитектуре. Согласно знаниям, заложенным в методологии (унифицированного процесса), **сначала должны быть разработаны варианты использования**, то есть функции, а потом для того чтобы обеспечить выполнение этих функций разрабатывается архитектура системы. **С другой стороны архитектура должна обеспечить реализацию необходимых сейчас и в будущем функций**, то есть вариантов использования. Реально архитектура и варианты использования разрабатываются параллельно.

Таким образом, архитектор придает системе форму и архитектор, проектируя форму, должен заложить такие решения, которые бы позволили системе **развиваться не только в момент начальной разработки, но и в будущих поколениях системы**. Чтобы найти такие формы архитектор должен хорошо понимать ключевые варианты использования системы, которые составляют ядро функционирования системы.

Архитектор выполняет следующие работы:

1. Создает грубый набросок архитектуры (эскиз), начиная с той части, которая

не связана с вариантами использования (**платформа, ядро и т.п.**). Выделяет ключевые (конституирующие) варианты использования

2. Приступает к работе с выделенными ключевыми вариантами использования. Каждый такой вариант использования описывается и реализуется в понятиях **подсистем, классов и компонентов, на основе которых архитектор создает различные модели**

Архитектура определяется в виде представлений всех **моделей** системы, объединенных (сконфигурированных) в систему. Существуют архитектурные представления модели вариантов использования, модели анализа, модели проектирования, модели развертывания. Модель реализации включает в себя компоненты, доказывающие то, что архитектура выполнима. Результатом этой фазы является базовый уровень архитектуры.

3. На основе базового варианта архитектуры, разрабатывает другие варианты использования.
4. Процесс разработки носит циклический характер, так как при разработке очередных вариантов использования архитектору, возможно, потребуется внести изменения в архитектуру и на базе измененной архитектуры продолжить разработку вариантов использования. Процесс разработки продолжается до тех пор, пока архитектура не будет **признана стабильной**

9.3 Унифицированный процесс - итеративный и инкрементный

Разработка проекта информационной системы, может продолжаться от нескольких месяцев до нескольких лет. В таком случае, практически было бы разделить работу на небольшие части, которые назовем итерациями. Результатом выполнения каждой итерации будет приращение (инкремент). Для максимальной эффективности итерации должны быть управляемыми, то есть они должны быть запланированы и выполняться по плану. В таком случае, итерации имеют все признаки проекта, но поскольку их объемы работ, закладываемые в итерации сравнительно небольшие, то их можно назвать мини-проектами.

Задачи, которые образуют итерации, выбираются под воздействием двух

факторов:

1. В ходе итерации следует работать с группой вариантов использования, которая повышает применимость продукта в ходе дальнейшей разработки
2. В ходе разработки итерации следует заниматься серьезными рисками

Определив итерацию как мини – проект, тем самым мы определили и последовательность разработки этих мини-проектов – анализ, проектирование, реализация и тестирование. Приращения, получаемые в результате итерации не всегда аддитивны, особенно на начальных этапах разработки, когда некоторые решения имеют нестабильное состояние

Выполняя каждую итерацию, разработчики описывают необходимые варианты использования, создают мини-проект, используя в качестве каркаса, направляющего разработку – архитектуру системы, реализуют проект в компоненты и проверяют соответствие компонентов вариантам использования. Если итерация достигла своей цели, то процесс разработки переходит на следующую итерацию, в противном случае разработчики должны пересмотреть свои решения и попробовать другой подход

Достоинства управляемого итеративного процесса:

1. Управляемая итерация ограничивает финансовые риски затратами на одно приращение, так как если разработчикам потребуется повторить итерацию, то затраты будут на одну итерацию, а не стоимость всего продукта
2. Управляемая итерация снижает риски не поставки продукта заказчику в запланированные сроки
3. Управляемая итерация ускоряет темпы процесса разработки, так как для разработчиков короткий и точный план предпочтительнее длинного и вечно сдвигающегося
4. Управляемая итерация признает часто отвергаемый факт, что желания и требования пользователей не могут быть определены в начале разработки. Они обычно уточняются в последовательных итерациях. Такой подход облегчает адаптацию к изменениям требований

Подведем итоги:

Все три концепции, которые рассмотрены выше, одинаково важны для целостности процесса разработки. Архитектура предоставляет нам компонентную структуру, определяющую нашу работу в итерациях, в каждой из которых варианты использования определяют цели и направляют работу разработчиков. Удаление одной из этих частей сильно уменьшит ценность унифицированного процесса

Кроме того, архитектура дает представление о проектируемой системе, как наборе моделей и связей между ними. Каждая модель содержит элементы модели связи между элементами и ограничения. Разработка каждой модели важный комплекс решений разработчиков.

9.4 Жизненный цикл в унифицированном процессе

Жизненный цикл информационной системы - последовательность циклов унифицированного процесса. В процессе жизни информационной системы унифицированный процесс циклически повторяется. Каждый цикл состоит из четырех фаз (**анализ и проектирование, требований, проектирование, построение, внедрение**), каждая фаза подразделяется на **итерации**. Каждая итерация включает следующие рабочие процессы: требования, анализ, проектирование, реализация, тестирование

В ходе фазы анализа и проектирования требований идея системы превращается в концепцию готового продукта и создается бизнес-план разработки продукта. В концепции продукта должны быть отражены:

1. Основные понятия, которые будет использовать система
2. Функции, которыми должна обладать система, решая проблемы пользователей
3. План проекта по разработке системы и стоимость разработки

В ходе фазы проектирования разрабатываются варианты использования и архитектура системы. **Архитектура определяется** в виде представлений всех **моделей** системы, которые, будучи объединены (сконфигурированы) представляют систему целиком. Это значит, что существуют архитектурные представления модели вариантов использования модели анализа, модели проектирования, мо-

дели развертывания. Модель реализации включает в себя компоненты для доказательства того, что архитектура выполнима. Результатом этой фазы является **базовый уровень архитектуры**.

В ходе фазы **построения** происходит создание продукта – к архитектуре добавляются законченные программы. Базовый уровень архитектуры разрастается до продукта, готового к передаче пользователям.

В ходе фазы **внедрения** происходит работа различных пользователей с бета-версией продукта, исправление обнаруженных дефектов, тренинг сотрудников заказчика, поддержка работы пользователей по горячей линии

Результатом каждого цикла является новый выпуск системы, а каждый выпуск **это продукт готовый к поставке**.

9.5 Продукт унифицированного процесса

Готовый к поставке продукт включает: исходный код, воплощенный в компоненты, которые могут быть откомпилированы и проверены, руководство пользователя и дополнительные компоненты поставки. Но готовый продукт, согласно представлениям унифицированного процесса должен быть приспособлен не только для нужд пользователей, но и всех заинтересованных лиц, то есть всех лиц, которые будут работать с продуктом. Чем обусловлено такое понимание продукта? Ответ на этот вопрос – **это изменения** и, первую очередь, это **изменения требований**. Известно, что при разработке программ единственное, что **постоянно – это изменение требований**. В итоге, разработчики вынуждены начинать новый цикл разработки, а руководство вынуждено ее финансировать. Для того, чтобы эффективно выполнять новый цикл, разработчикам необходимо иметь полное представление о программном продукте и той среде, в которой будет функционировать этот продукт:

1. Модель предметной области (модель среды окружения)
2. Модель вариантов использования системы (функциональная модель)
3. Модель анализа (концептуальная модель и первичное распределение поведения)

4. Модель проектирования
5. Модель реализации, которая включает в себя компоненты (представленные исходным кодом) и раскладку классов по компонентам
6. Модель тестирования
7. Модель развертывания, которая определяет физические компьютеры- узлы сети и размещение компонентов по этим узлам

Все эти модели связаны между собой и являются результатами выполнения соответствующих рабочих процессов

9.6 Унифицированный процесс – методология разработки

Мы обсудили, ключевые идеи унифицированного процесса (процесс - управляемый вариантами использования, архитектурно-ориентированный, итеративный, инкрементный). Для эффективного применения этих идей необходимо, чтобы они образовывали единый многоплановый процесс, поддерживающий циклы, фазы, рабочие процессы, снижение рисков, контроль качества, управление проектом и конфигурацией. Унифицированный процесс создает каркас, объединяющий все эти аспекты. Такой комплекс знаний, называют **методологией разработки**.

10. Анализ. Концептуальная (аналитическая) модель

Концептуальная (аналитическая) модель

Этапу анализа предшествует этап определения требований к информационной системе, в процессе которого выявляются проблемы предметной области (производится проблематизация предметной области) и формируются предложения по решению выявленных проблем посредством информационной системы. Предложения по решению проблем относятся, прежде всего, к функциональным возможностям, которым должна обладать информационная система, чтобы успешно решать выявленные проблемы. После формирования списка функциональных возможностей (в контексте концепции – функциональных требований) и основных понятий, которыми будут оперировать функциональные возможности, переходят к выполнению работ этапа анализа, а затем и проектирования. Результаты этапа определения требований являются входными данными этапа анализа.

Перед тем как перейти к изложению принципов анализа (построения концептуальной модели) необходимо сделать несколько замечаний, касающихся рассматриваемой в данном курсе последовательности выполнения действий по определению требований, анализу и проектированию. Дело в том, что традиционно в курсах лекций и технических статьях по методологии RUP, функциональные требования к информационной системе оформляются в виде диаграммы прецедентов (use case). При выполнении лабораторной работы, выполняемой в рамках данного курса лекций (практической работы по проектированию), не требуется разрабатывать диаграммы прецедентов информационной системы, хотя и не запрещается. Посредством диаграмм прецедентов (а детальнее и диаграмм активности или состояний) при выполнении лабораторной работы моделируется предметная область, причем в аспектах комплексной архитектуры (схемы Захмана). Поэтому входными данными для анализа служит перечень функциональных возможностей и основных понятий из раздела концепции. При желании, используя знания объектно-ориентированного программирования, студенты сами могут описать посредством диаграмм прецедентов (use case), функциональные требования к информационной системе и далее использовать вместе с другими данны-

ми в качестве входной информации этапа анализа.

А теперь, определим понятие **анализа**.

Под **анализом** будем понимать процесс **анализа требований** (и прежде всего, функциональных требований) к информационной системе, сформированных на этапе определения требований. В процессе анализа решается задача анализа, результаты которой: **декомпозиция системы на более мелкие элементы и определение свойств системы или среды**, окружающей систему.

Целью анализа может быть определение закона преобразования информации, задающего поведение системы (например, преобразования функциональных требований в поведение системы). При этом система предстает как один элемент, на вход которого поступают входные данные (сообщения), а на выходе, в зависимости от закона поведения, формируются выходные данные, **удовлетворяющие пользователя**.

Например, в контексте методологии RUP результат анализа это **концептуальная схема** (диаграмма классов, моделирующих основные понятия и решающая задачу декомпозиции) и реализация варианта использования в виде диаграммы взаимодействия, моделирующая поведение системы посредством взаимодействия объектов анализа. При условии рассмотрения системы в виде одного элемента входными данными будут сообщения пользователя, а выходными сообщения системы об удовлетворении (или отказе) выполнения запросов пользователя. Выходные сообщения системы в таком случае (моделировании поведения) называют **списком ответственности** системы. Система как бы отвечает за эти сообщения (в последствие эти сообщения могут быть интерпретированы как экранные формы документов, рисунки и т.п.)

Концептуальная (аналитическая) модель системы предназначена:

1. Представления более детальной спецификации требований к системе (более точную, чем в спецификациях на этапе определения требований и разработки вариантов использования) и может рассматриваться как первый шаг к модели проектирования, то есть служить входными данными

для проектирования системы

2. Представления системы с использованием языка разработчиков, поэтому позволяет вводить больше формализма и может использоваться для анализа внутренних механизмов системы
3. **Решения задачи анализа**, которая заключается в **декомпозиции** системы на более мелкие элементы и определение свойств системы или среды, окружающей систему. Так, согласно рекомендациям RUP, варианты использования (если они разрабатывались) в модели анализа преобразуются в диаграммы классов анализа (при необходимости) и диаграммы взаимодействующих объектов анализа. Основным элементом декомпозиции в них является класс. Представление вариантов использования посредством диаграмм взаимодействующих объектов класса анализа называется «анализом реализации варианта использования» (вариант поведения системы). Такое представление содержит описание реализации варианта использования как в терминах взаимодействующих объектов классов анализа, так и в терминах естественного языка (Приложение 1. Пример текстового описания варианта использования).

Обобщенная модель анализа представлена на Рисунок 13

Модель анализа представлена **системой анализа**, которая, в свою очередь, может быть представлена **подсистемами (пакетами анализа)**. Такое представление удобно с точки зрения управления процессом анализа и управления проектом в целом. Посредством **пакетов анализа** могут быть представлены не только абстракции подсистем, но и абстракции уровней.

Базовыми элементами (артефактами) модели анализа являются **классы анализа**. На основе классов анализа, как уже отмечалось выше, образованы реализации вариантов использования, подсистемы, и система в целом

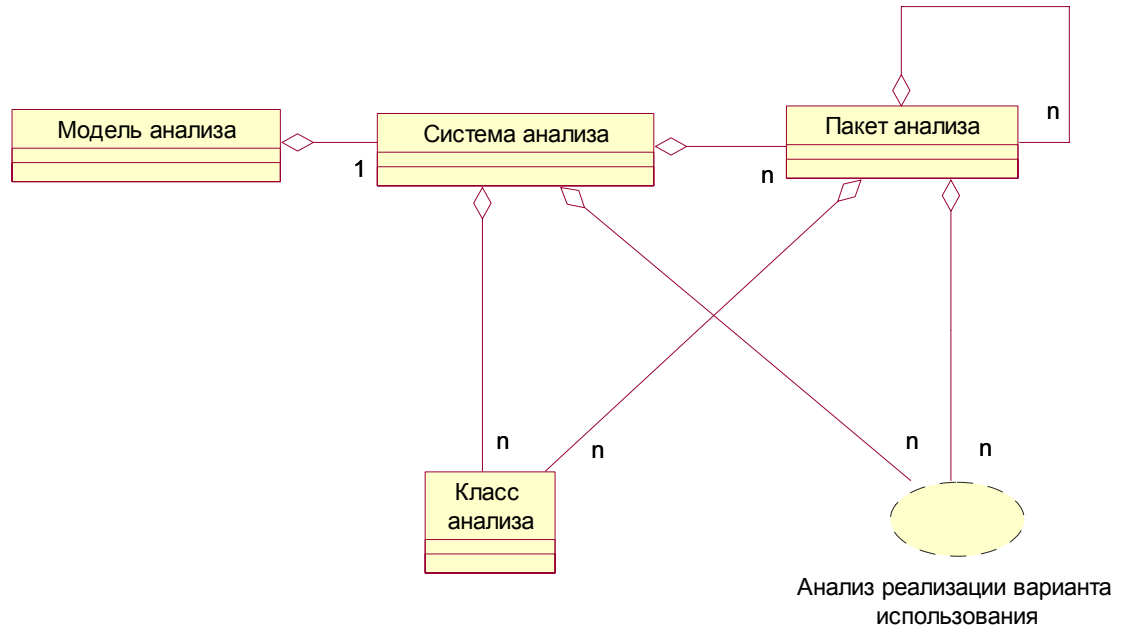


Рисунок 13

Анализ реализации варианта использования – организованность (кооперация) внутри концептуальной (аналитической) модели, описывающая реализацию и выполнение варианта использования в понятиях классов анализа и взаимодействующих объектов анализа.

Рассмотрим детальнее понятие класса анализа.

Класс анализа представляет собой абстракцию одного или нескольких классов (и/или подсистем) в проекте системы. Эта абстракция имеет следующие характеристики:

1. Класс анализа сосредоточен на представлении **функциональных** требований и откладывает не функциональные требования на последующие стадии – проектирование и реализацию. Такое положение определяет класс анализа как концептуальный класс.
2. Класс анализа редко поддерживает, какие либо интерфейсы в понятиях операций. Вместо этого его поведение определено в соответствии с ответственностями (обязанностями) верхнего менее формального уровня. Ответственность в контексте концептуальной модели это текстовое описание

поведения для рассматриваемого класса. Класс анализа определяет атрибуты высокого уровня

3. На множестве классов анализа задаются отношения, которые также как и классы анализа относят к концептуальному уровню представления. Например, направленность отношения не слишком важна в анализе, но существенна для проектирования.
4. Класс анализа всегда можно отнести к одному из типов: **граничный, управляющий, сущности**. Каждый из типов подразумевает определенную семантику (смысловую нагрузку), которая приводит к мощному и логичному методу обнаружения и описания классов анализа и способствует созданию качественной модели объекта.
5. Классы анализа и заданные на них отношения, образуют **концептуальные схемы** (необходимые высказывания). Концептуальные схемы понимаются в терминологии стандарта: ГОСТ Р 34.320-96. Концептуальные схемы представленные, с помощью CASE-средств на каком – либо формальном языке, (например, на языке UML), будем называть **концептуальными моделями**.
6. Классы анализа для концептуальной схемы (ГОСТ Р 34.320-96) определяются на основании следующих принципов:
 - а). описания классов (типов) сущностей проблемной области, а не отдельных экземпляров
 - б). описания понятий, менее подверженных изменениям
 - в). включение правил или ограничений, имеющих широкое воздействие на поведение предметной области (и поэтому на поведение концептуальной схемы и информационной базы)
 - г). В любом случае должны соблюдаться общие принципы концептуальной схемы:

П р и н ц и п 100%

Принцип, согласно которому все общие аспекты, т. е. все правила, законы и т. д., проблемной области должны быть описаны в концептуальной схеме, причем информационная система не может нести ответ-

ственность за несоблюдение правил и законов, описанных не в концептуальной схеме.

П р и н ц и п к о н ц е п т у а л и з а ц и и

Принцип, согласно которому концептуальная схема должна включать статические и динамические аспекты проблемной области только концептуального уровня, не касаясь внешних и внутренних аспектов представления и организации данных (физической организации данных и доступа к ним, аспектов представления, касающихся отдельных пользователей)

Рассмотрим типы классов анализа: граничный, управляющий, сущности. В процессе рассмотрения, в качестве примеров, отобразим на языке UML элементы **концептуальной схемы** (перечень необходимых высказываний). Высказывания приводятся перед каждой диаграммой и служат словесным описанием диаграммы.

10.1 Граничные классы

Граничные классы используются для моделирования взаимодействия между системой и ее актантами (пользователями и внешними системами). Взаимодействие часто включает в себя получение (и передачу) информации, запросы пользователей и внешних систем. Граничные классы часто представляют собой абстракции окон, форм, панелей, коммуникационных интерфейсов, интерфейсов принтера, датчиков, терминалов. Граничные классы находятся на высоком концептуальном уровне и не должны описывать каждую мелочь (например, интерфейса пользователя) Вполне достаточно, если граничные классы опишут только то, что происходит при взаимодействии с актантом (запросы, ответы)

Пример (Рисунок 14). Пользователь просматривает счета, подлежащие оплате, посредством интерфейса запросов на оплату и **дает команду** системе «оплатить сумму по **документу** счет. Но вместе с тем, интерфейс запросов на оплату **позволяет** покупателю **отклонить оплату** по документу счет, который покупатель не желает оплачивать.

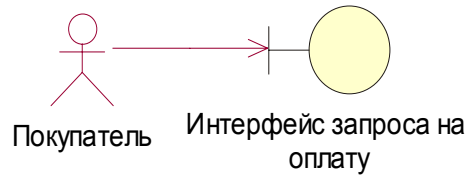


Рисунок 14

10.2 Классы сущностей

Классы сущностей используются для моделирования долгоживущей, устойчивой информации. Классы сущностей моделируют информационное наполнение и связи явлений и концепций – человека, объекта или события реального мира. В большинстве случаев классы сущностей являются моделями бизнес - классов сущностей. Отличие между ними заключается в том, что классы сущностей обрабатываются проектируемой системой, а бизнес - классы сущностей моделируют объекты предметной области.

Пример (Рисунок 15). Класс сущностей с названием **счет** используется для моделирования документа **счет**. Вместе с граничным классом «интерфейс запроса на оплату» они позволяют пользователю **просматривать** документы **счет**. Документы кредитовые счета используются для оплаты покупок (кредитовый счет – расход по банковскому счету). Документы дебетовые счета используются для регистрации прихода средств на банковский счет. **Вычисляемая разность** между суммами дебетовых и кредитовых счетов **дает сумму банковского счета**.

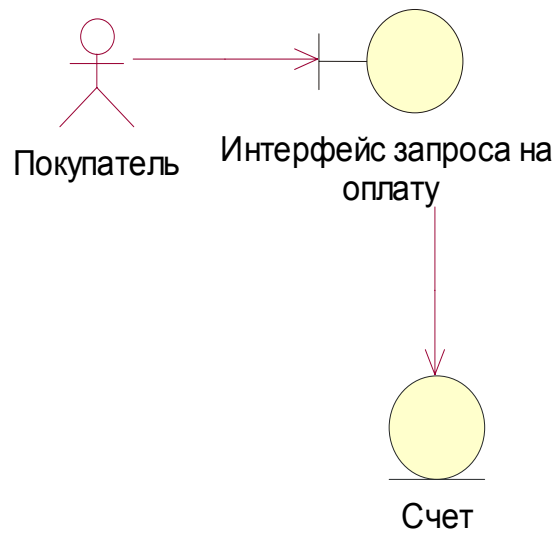


Рисунок 15

10.3 Управляющие классы

Управляющие классы отвечают за координацию, порядок последовательности, взаимодействия и управления другими объектами и часто используются для управления процессом информационной поддержки некоторого варианта использования. Классы управления используются для представления сложных переходов и вычислений, например, бизнес – логики, которая не связана с классами сущностей.

Динамика системы (поведение системы) моделируется посредством управляющих классов, объекты которых координируют основные действия и потоки управления и делегируют работу объектам других классов (граничным и сущностей)

Пример (Рисунок 16). Планировщик оплат **принимает запрос** на платеж (например, запрос на оплату документа счет и дату оплаты счета). Позднее (если платеж с отсрочкой), в день платежа, **планировщик** оплат **осуществляет платеж**, проводя перевод денег между соответствующими счетами.

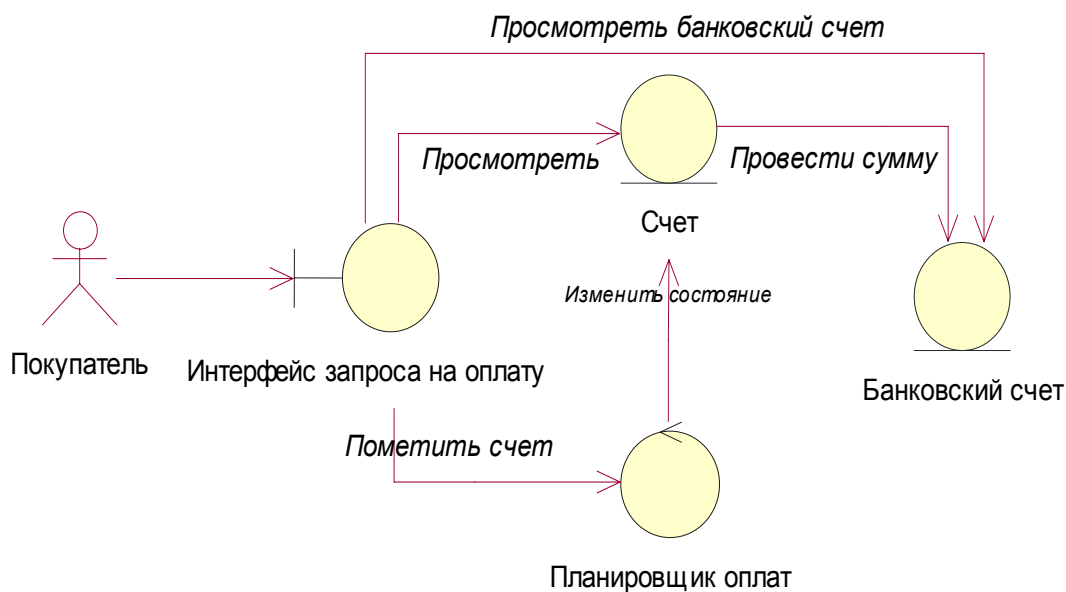


Рисунок 16

Рассмотрев основные типы классов анализа, перейдем к **рассмотрению понятия анализ реализации варианта использования** (слово анализ применено с целью зафиксировать этап проектирования).

В определении этого понятия говорится, что это организованность внутри концептуальной (аналитической) модели, описывающая реализацию и выполнение варианта использования в понятиях взаимодействующих объектов классов анализа (объектов анализа). Другими словами (понятиями системного анализа) это структура, элементами которой являются объекты анализа, а связи представлены операциями взаимодействия. Такие структуры в терминах объектно-ориентированного языка UML называются диаграммами взаимодействия. В UML существуют два вида диаграмм взаимодействия: диаграммы последовательности и диаграммы кооперации.

Далее рассмотрим проектирование реализации варианта использования как части концептуальной модели. Но прежде чем перейти к проектированию, уточним назначение этого артефакта. Реализация варианта использования предназначена для моделирования посредством **взаимодействующих объектов анализа** варианта использования, а точнее сценариев поведения, описанных в вари-

антах использования. **Причем, в данном конспекте лекций**, как и в методологии RUP, различаются варианты использования, отражающие **процессы предметной области** (в контексте данного курса концептуальная модель предметной области – варианты использования, детализированные посредством диаграмм активности) и варианты использования, с помощью которых **моделируют функциональные возможности проектируемой информационной системы**. Основным элементом структуры варианта использования является **сценарий**, задающий процессы модели предметной области, либо функциональные возможности информационной системы.

Сценарии (как предметной области, так и информационной системы), отраженные в вариантах использования обычно моделируются диаграммами активности. Сценарии, модели которых представлены **взаимодействующими объектами анализа** на диаграммах взаимодействия, называются **реализациями вариантов использования** и моделируют всю ту же функциональность, но уже в терминах **объектов анализа взаимодействующих** между собой.

Преобразование варианта использования в реализацию варианта использования называется **трассировкой**. Результаты трассировки имеют важное значение для моделирования функциональных возможностей информационной системы.

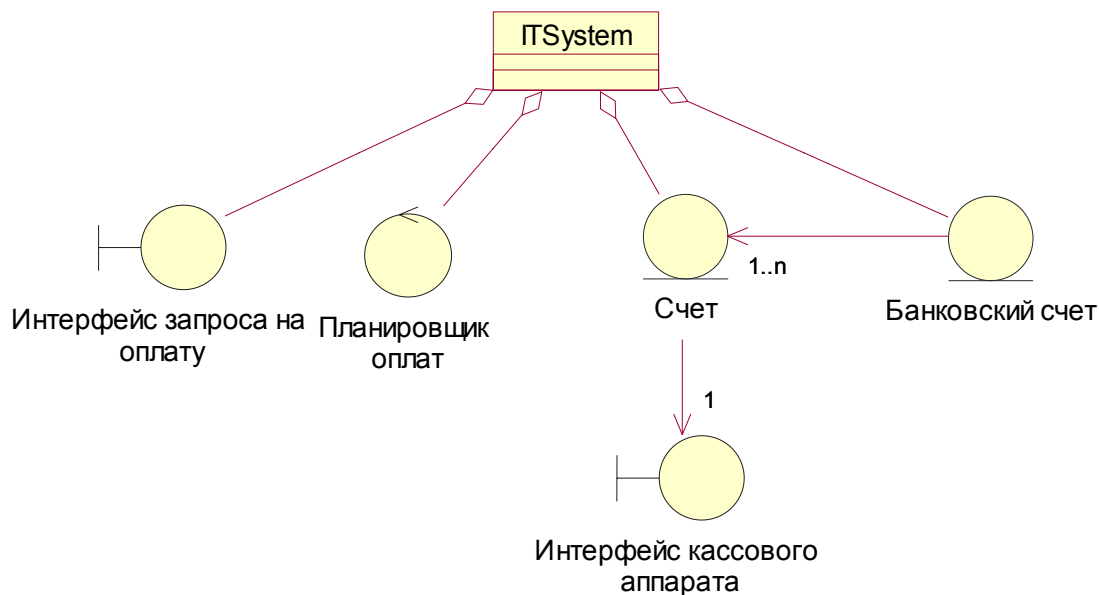


Рисунок 17

Рассмотрим процесс создания реализации варианта использования.

Выше отмечалось, что в случае, когда целью анализа является **определение закона преобразования информации**, задающего поведение, система предстает как **один элемент**, на вход которого поступают входные данные (сообщения), а на выходе, в зависимости от закона поведения, формируются выходные данные (Рисунок 18). Список названий сообщений или документов таких выходных данных будем называть **списком «ответственности»**. Название этого списка, видимо, происходит от выражения - информационная система несет **ответственность** за те или иные действия (за определенное поведение).

Сформируем список ответственности посредством диаграмм языка UML. (Рисунок 18). Но перед тем как перейти к анализу списка ответственности поясним посредством диаграммы (Рисунок 17) происхождение объекта ИС (информационная система)

Данная диаграмма является первым решением на пути проектирования поведения информационной системы в терминах взаимодействующих объектов. Перед началом проектирования этой диаграммы мы предполагали определить закон преобразования информации, задающий поведение системы. Другими словами – нам необходимо знать - как должна быть организована информационная

система внутри, чтобы она смогла реализовывать требуемое поведение (требуемые функциональные возможности). Такая постановка задачи является основной задачей проектирования и наиболее часто представлена в следующих формах:

1. обратная задача проектирования
2. прямая задача проектирования

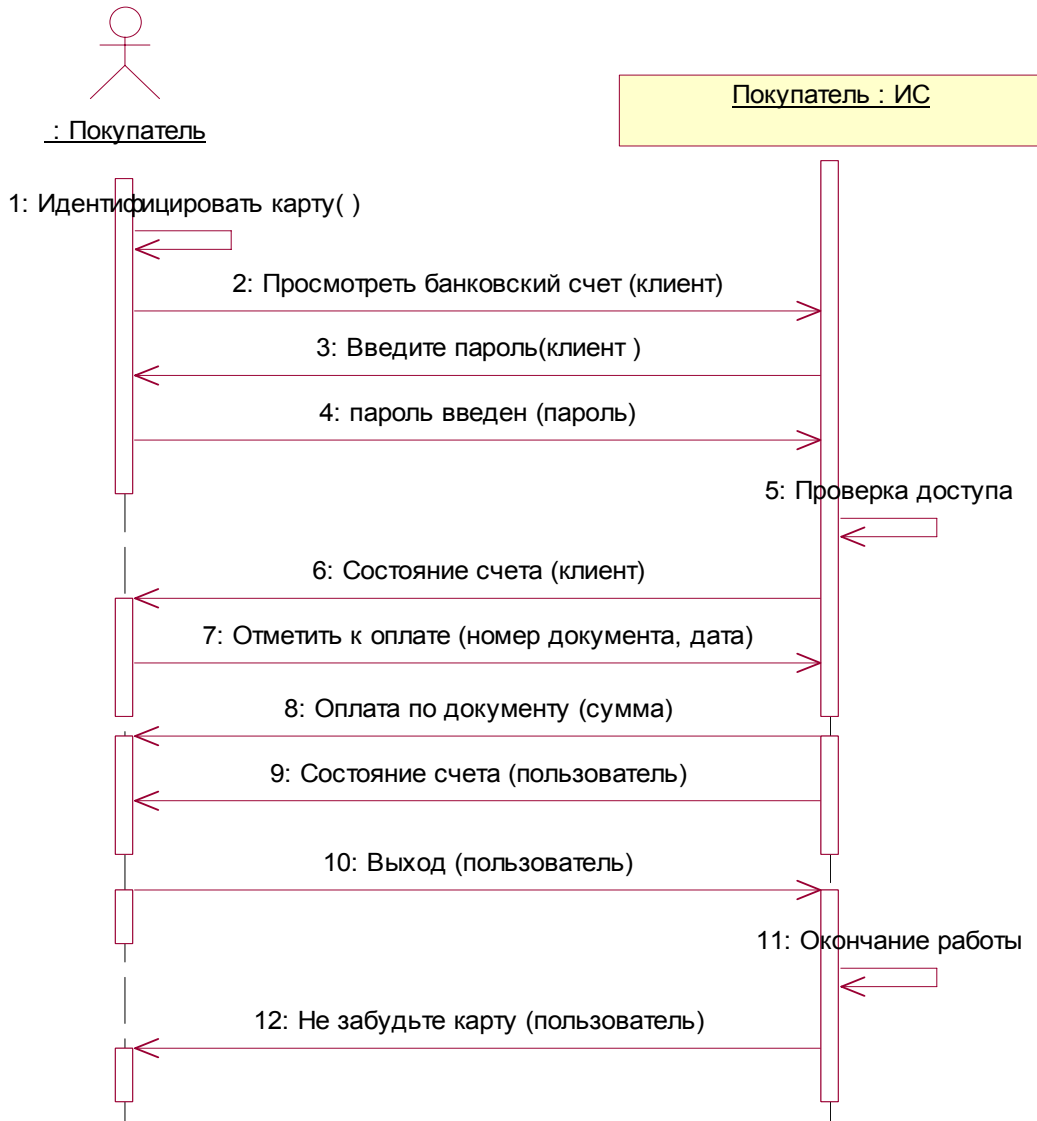


Рисунок 18

11. Проектирование. Модель проектирования (логическая модель)

Модель проектирования (в рамках данного курса логическая модель)

Модель проектирования – это объектная модель, которая описывает процесс проектирования (по RUP конструирования) системы и используется в качестве исходных данных для процесса реализации системы. Обобщенная модель проектирования представлена на Рисунок 19

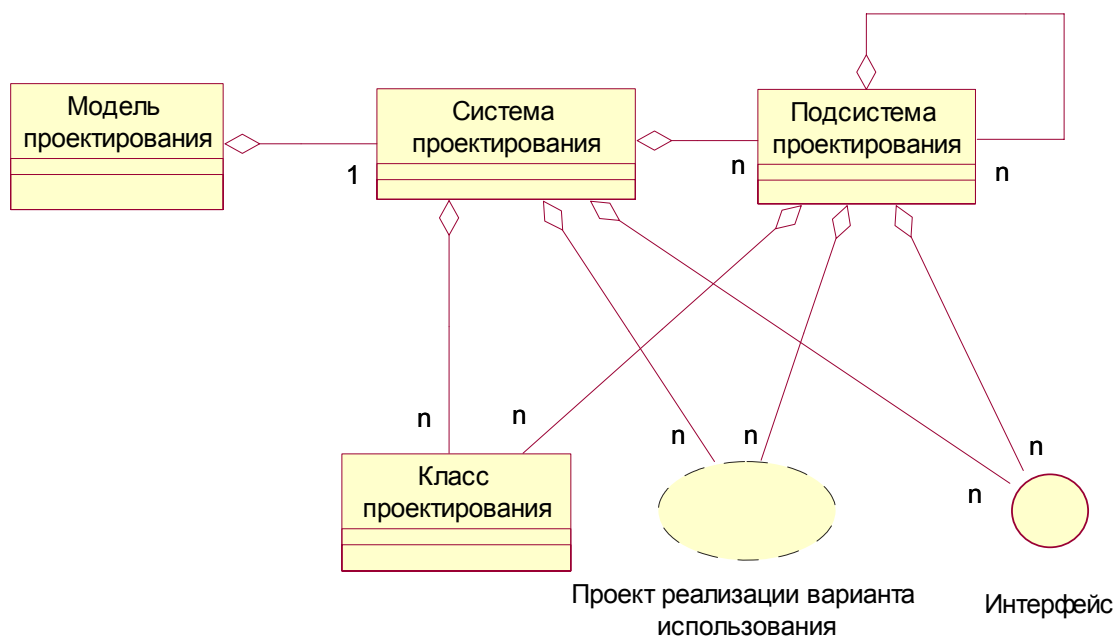


Рисунок 19

Модель проектирования представлена **системой проектирования**, которая, в свою очередь, может быть представлена **подсистемами (пакетами проектирования)**. Такое представление удобно с точки зрения управления процессом проектирования и управления проектом в целом. Посредством **пакетов проектирования** могут быть представлены не только абстракции подсистем, но и абстракции **уровней**.

Рассмотрим составные части модели проектирования.

Базовыми элементами (артефактами) модели проектирования являются **классы проектирования**. На основе классов проектирования образованы проек-

ты реализации вариантов использования, интерфейсы, подсистемы, и система в целом. Класс проектирования наиболее приближенная к реальности абстракция по следующим причинам:

1. Язык для описания класса проектирования тот же, что и язык программирования для реализации. Соответственно, операции, параметры, атрибуты, типы и другие подробности определяются с использованием синтаксиса выбранного языка программирования
2. Обычно задается видимость атрибутов и операций класса проектирования (public, protected, private).
3. Отношения, в которых участвует класс проектирования, обычно получают явное выражение при реализации этого класса. Например, обобщение имеет семантику, которая соответствует обобщению (или наследованию) в языке программирования. Таким образом, обобщение и агрегация часто отображаются на соответствующие переменные (атрибуты) реализации, соответствующие ссылкам на объекты.
4. Методы (реализации операций средствами языка программирования) класса проектирования прямо отображаются на соответствующие методы классов реализации (тексты программ)
5. Класс проектирования может переложить обработку некоторых требований на последующую реализацию, передав ей требования к реализации класса. В результате появляется возможность отложить принятие решений, которые невозможно принять на основе модели проектирования, например, касающиеся вопросов кодирования класса.
6. Класс проектирования часто задается стереотипом, который напрямую отображается в конструкцию соответствующего языка программирования (так для Visual Basic следующие стереотипы – «модуль класса», «форма», «элемент управления» и т.д.).
7. Класс проектирования может быть реализован в виде **интерфейса**, если это понятие существует в выбранном языке программирования. Например, класс проектирования, представляющий класс языка Java может быть

реализован в виде **интерфейса**.

8. Класс проектирования может быть активным. Это означает, что объекты класса будут использовать свою собственную нить управления, работая параллельно с другими активными объектами. Однако, обычно, классы проектирования не активны

Проект реализации варианта использования – организованность (кооперация) внутри модели проектирования, описывающая реализацию и выполнение варианта использования в понятиях классов проектирования и их взаимодействующих объектов проектирования. Проект реализации вариантов использования может непосредственно трассироваться из анализа реализации варианта использования модели анализа.

Проект реализации вариантов использования содержит текстовое описание **потока событий, диаграммы классов и диаграмму взаимодействий**, которые отображают поток событий конкретного сценария варианта использования, но посредством взаимодействий между объектами проектирования.

В данном лекционном курсе и лабораторной работе, выполняемой в его рамках, проект реализации варианта использования содержит **диаграмму классов и диаграмму взаимодействий**, (последовательности) которые дают представление о **поведении системы**.

Модель поведения – система взаимодействующих посредством сообщений объектов, которые после их (сообщений) определения должны быть представлены в модели классов в качестве операций взаимодействия.

Интерфейсы предназначены для задания операций, выполняемых классом проектирования или подсистемой.

Интерфейсы предоставляют способ отделения спецификации функциональности в **терминах операций** от ее реализации в **терминах методов**. Это отделение делает клиентов, которые зависят от интерфейса или используют его, независимыми от реализации интерфейса. Отдельная реализация интерфейса (класс проектирования или подсистема) может быть заменена другой реализацией, при этом никаких изменений в клиенте делать не потребуется.

Большинство интерфейсов между подсистемами считается архитектурно значимыми, поскольку они определяют способы взаимодействия подсистем. В некоторых случаях они также используются для создания стабильных интерфейсов в начале жизненного цикла системы (еще до того как в подсистеме будет реализована объявленная функциональность)

11.1 Подходы к разработке модели проектирования

Разработка модели проектирования (или в рамках данного лекционного курса – логической модели) с **определения классов проектирования**, которая может состоять из следующих работ:

1. Определение на основе признака вовлеченности класса в реализацию варианта использования. Классы не участвующие в реализации варианта использования не являются необходимыми
2. Определение на основе классов анализа. Некоторые классы проектирования могут быть получены из архитектурно – **значимых классов анализа**. Причем, отношения между классами анализа могут быть использованы для получения пробного набора отношений между соответствующими классами проектирования
3. Определение на основе признака активности класса. Активные классы, обеспечивая возможность параллельной работы с другими классами, решают ряд проблем, стоящих перед системой (производительность, распределение по узлам и т.п.).

После определения значимых для системы классов проектирования приступают к разработке **диаграмм взаимодействия модели поведения**.

Последовательность действий на диаграмме взаимодействия обычно начинается с того, что актант инициирует вариант использования, посылая системе некоторое сообщение. Внутри системы некий объект проектирования принимает от актанта сообщение. Затем этот объект проектирования вызывает другие объекты, которые, в свою очередь, вызывают другие объекты. Взаимодействие объектов происходит с **целью** выполнить вариант использования. В ходе проектирования обычно изображают этот процесс при помощи **диаграмм последователь-**

ности, которые фокусируют внимание на получение детальной хронологической последовательности взаимодействий. Диаграммы последовательности иллюстрируют взаимодействие объектов при помощи обмена сообщениями между линиями жизни объектов или подсистем. Имя сообщения должно указывать на операцию внутреннего объекта или интерфейса, предоставляемого объектом.

После разработки диаграммы взаимодействий (в данном случае последовательности) **определение** участвующих во взаимодействии **интерфейсов** и включение их в состав классов проектирования

В итоге, на основе классов анализа, взаимодействий объектов проектирования и интерфейсов получаем систему классов проектирования. Каждый объект классов проектирования такой системы исполняет свои роли в реализациях варианта использования и отвечает предъявляемым к нему нефункциональным требованиям.

Каждый класс анализа и проектирования структурно представлен множеством объектов этого класса, то есть набором экземпляров класса, которые отличаются значениями атрибутов. Понятие класс в методологии объектно-ориентированного анализа и проектирования является обобщением понятия объекта. **Объектом** называется все то, что можно мысленно выделить из окружающей его среды, путем указания свойств и признаков, существенных для данного понятия.

Рассмотрев, структуру элементов, образующих класс констатируем, что каждый класс проектирования как обобщающее множество объектов может быть охарактеризован:

1. Операциями
2. Атрибутами
3. Отношениями, в которые он вступает
4. Методами, которые реализуют его операции
5. Состояниями, в которых он находится
6. Зависимостями от обобщенных механизмов проектирования
7. Требованиями, относящихся к его реализации

8. Правильной реализацией всех интерфейсов, которые он должен предоставить

Перед тем как рассмотреть определение характеристик класса проектирования дадим **более подробное описание** обработки классов проектирования в зависимости от стереотипов классов анализа (**граничный, сущности, хранения**)

1. Проектирование граничных классов определяется спецификой технологии программирования. Например, граничные классы проектируемые под VB, должны включать класс проектирования «Форма», а также классы проектирования, которые представляют собой управляющие элементы пользовательского интерфейса, возможно элементы ActiveX.
2. Проектирование классов сущности, содержащих информацию длительного хранения, часто определяется спецификой используемой технологии базы данных. Например, могут появиться классы проектирования соответствующие таблицам реляционной модели данных.
3. Проектирование управляющих классов является сложным и тонким делом, поскольку они обеспечивают сценарии, координацию с другими объектами, а иногда и чистую бизнес-логику. Для понимания проектирования управляющих классов рассмотрим следующие аспекты (проблемы):
 - а). При размещении сценария и управляющей последовательности в различных узлах сети – управляющие классы проектирования должны быть разработаны различные классы для различных узлов (проблема размещения)
 - б). При разработке классов проектирования, реализующих управляющий процесс (систему управления) принято не создавать без необходимости несколько отдельных различных классов. Напротив, прежде всего, должна рассматриваться возможность интеграции (совмещения) управляющего класса с граничным классом или классом сущности (проблема производительности)
 - в). Поскольку управляющие классы иногда для организации процесса управления используют транзакции, то соответствующие классы проек-

- тирования должны включать в себя использование какой-либо из существующих технологий управления транзакциями (проблема транзакций)
- г). Классы проектирования, определяемые на этом этапе, должны быть связаны трассировкой с соответствующими классами анализа, от которых они порождаются. Важно помнить «природу» классов проектирования, так как мы будем уточнять их на последующих этапах.

11.2 Определение операций класса проектирования

Операциями называются действия, которые должен выполнять объект класса, участвуя в реализации сценариев вариантов использования под воздействием множества внутренних и внешних факторов.

Исходными данными для определения операций класса проектирования служат:

1. Ответственность, определяемая в концептуальной модели (модель анализа), которые, как правило, трассируются в одну или несколько операций класса проектирования
2. Специальные требования каждого из классов анализа, которые трассируются в класс проектирования и должны быть обработаны в модели проектирования. Такая обработка может быть учтена, например, посредством включения общеизвестных технологий проектирования, таких как технология баз данных.
3. Интерфейсы (наборы операций), которые необходимы для взаимодействий объектов класса проектирования, при реализации вариантов использования.
4. Проекты реализаций вариантов использования, в которых участвует класс проектирования

Методом, который лежит в основе создания проекта реализации варианта использования является организация выполнения сценария.

Категория **«организация»** многозначна. В широком смысле это свойство систем, в узком – **действия, направленные на создание чего-либо** (например, ор-

ганизация собственного «дела», организация необходимого поведения системы). В то же время организацией можно назвать группу людей, объединившихся ради достижения общих целей. Это могут быть производственные, общественно-политические и спортивные организации, организации любителей пива и т. д.

Организация выполнения сценария является действием, направленным на достижение вариантом использования поставленных целей. Для того, чтобы организовать достижение целей необходимо выполнить следующие действия:

1. Идентифицировать (определить) объекты классов проектирования, которые будут участвовать в достижении целей
2. Распределить каждому из объектов те действия (операции), за выполнение которых они будут отвечать (распределить ответственность)
3. Используя операции объектов, задать на множестве объектов последовательность действий (сценарий), который через взаимодействие (обмен сообщениями) реализует некоторое поведение системы для достижения цели

Пример реализации варианта использования (уровня проекта) приведен на Рисунок 20. Вариант использования предназначен для моделирования взаимодействия объектов системы при **отображении состояния банковского счета, создании документа счет по данным кассового аппарата и оплаты документа счет**. На диаграмме заданы следующие сценарии: просмотр банковского счета, прием данных об оплате, оплата покупки.

Операции класса проектирования должны поддерживать **все роли**, которые исполняет этот класс в различных реализациях вариантов использования. **Роль определяются посредством анализа диаграмм взаимодействия**, описывающих реализации вариантов использования, и выявление в последовательности операций и потоках событий (сценариях) некоторого списка операций выполняемых объектом класса. Такой список операций образует роль, выполняемую объектом класса в рассматриваемом варианте использования

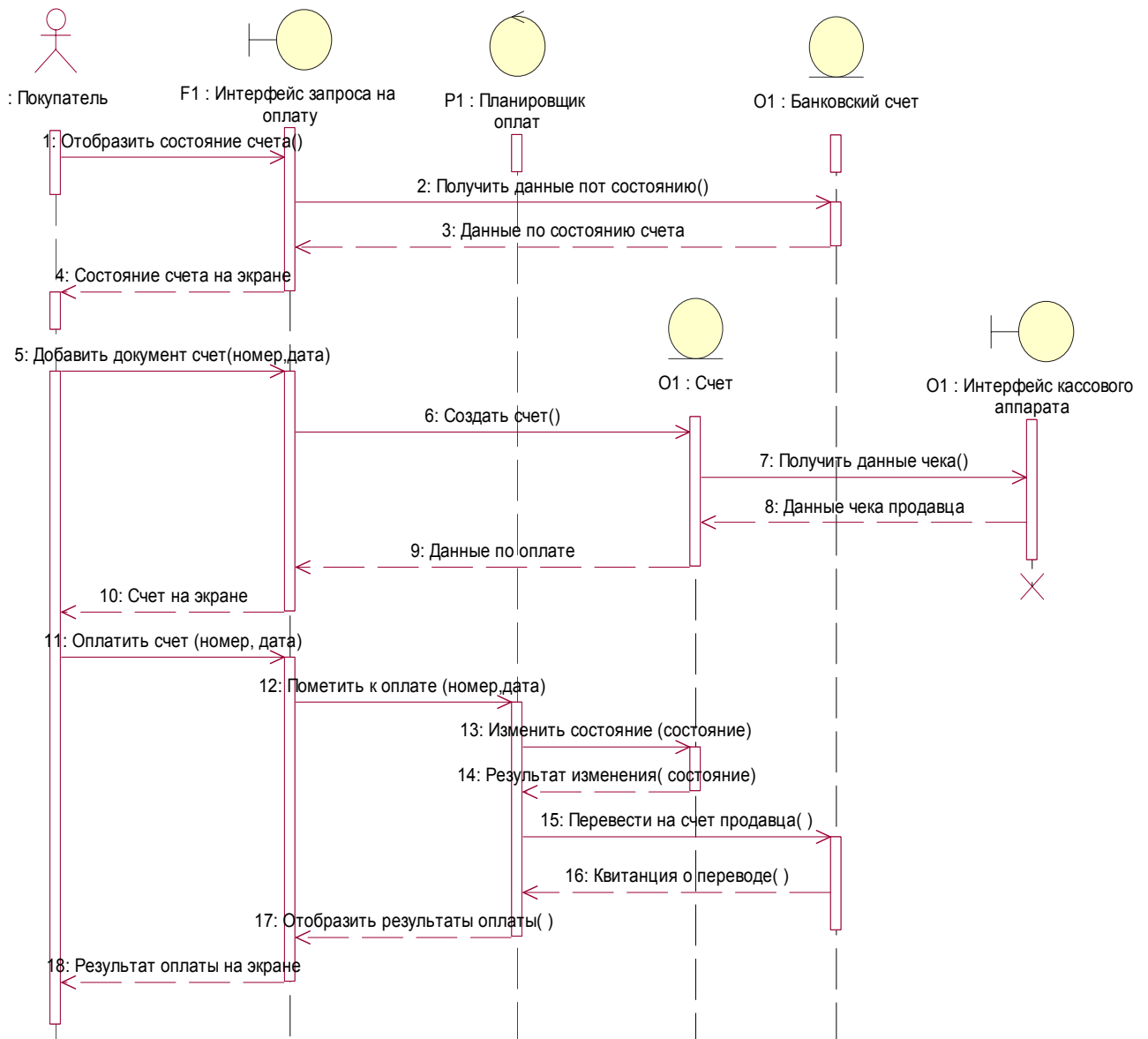


Рисунок 20

11.3 Шаблоны проектирования

При распределении ответственности и разработке способов взаимодействия объектов разработчику предоставляется достаточная свобода действий. Неудачный выбор метода распределения может привести к тому, что системы и их отдельные компоненты окажутся непригодными для поддержки, понимания, повторного использования и резервирования. Неудачи можно избежать, если при рас-

пределении ответственности применять принципы объектно-ориентированного проектирования. Некоторые из этих принципов систематизированы в шаблонах GRASP и применяются при разработке диаграмм взаимодействия, то есть при распределении ответственности между объектами и разработке способов их взаимодействия.

В объектно-ориентированном проектировании шаблоном называют описание проблемы по распределению ответственности и ее решения. Результат решения проблемы это операции назначенные объекту для выполнения успешного (а в общем случае требуемого) сценария взаимодействия. В идеале шаблон должен содержать советы по поводу его применения в различных ситуациях. С этой точки зрения шаблоны являются основным механизмом для накопления и повторного использования полезных принципов разработки программного обеспечения.

11.3.1 Шаблон MVC (*Model-View-Controller*)

Шаблон проектирования Model-View-Controller (далее просто MVC) использован в основе архитектурного решения первой среды программирования с графическим интерфейсом пользователя – Smalltalk-80. Согласно шаблону при проектировании следует разделять данные приложения, пользовательский интерфейс и управляющую логику на три отдельных компонента: *модель*, *представление* и *контроллер* – таким образом, что модификация одного из компонентов оказывает минимальное воздействие на остальные. *Модель* (Model) предоставляет данные предметной области *представлению* и реагирует на команды *контроллера*, изменяя свое состояние. *Представление* (View) отвечает за отображение данных предметной области пользователю, реагируя на изменения *модели*. *Контроллер* (Controller) интерпретирует действия пользователя, оповещая *модель* о необходимости изменений.

Приведем простой пример MVC. *Модель* может представлять собой объект, реализующий переключатель. В простейшем случае данная *модель* характеризуется состоянием: выключен или включен – и, кроме того, позволяет изменять его – объект *модели* имеет метод изменения состояния: выключить и включить. *Пред-*

ставление отображает на дисплее пользователя состояние переключателя с помощью определенной текстовой или графической формы. Например, *представление* может отображать текстовую метку, которая при изменении состояния *модели* переключателя отобразит соответствующий текст: «выключен» или «включен». Помимо отображения *представление* позволяет пользователю изменять состояние переключателя с помощью графических примитивов, к примеру, двух кнопок с надписями: «Включить» и «Выключить». *Представление* умеет только отображать состояние *модели* переключателя, для изменения *представление* обращается к *контроллеру*. *Контроллер* представляет собой объект, который в нашем случае умеет только изменять состояние *модели* переключателя.

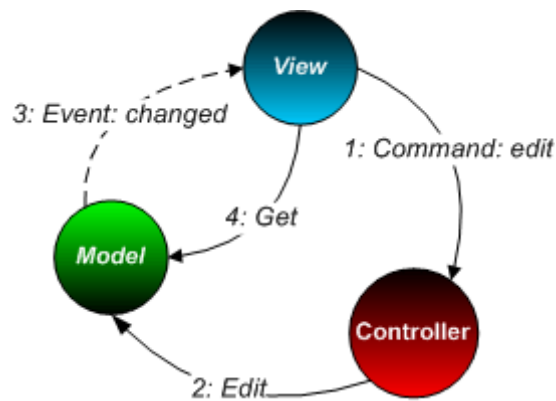


Рисунок 21

Полный цикл работы данной MVC-триады: *модели, представления и контроллера* переключателя – можно описать следующим образом. При инициализации *представления* пользователем оно обращается к *модели* и устанавливает текст метки в соответствии с текущим состоянием переключателя. Пользователь инициирует изменение переключателя, нажимая на определенную кнопку. При этом *представление* отправляет соответствующую команду *контроллеру*: включить или выключить. *Контроллер* интерпретирует команду и изменяет *модель*.

Представление регистрирует изменение *модели*: по этому событию оно изменяет текст метки для соответствия новому состоянию *модели* переключателя.

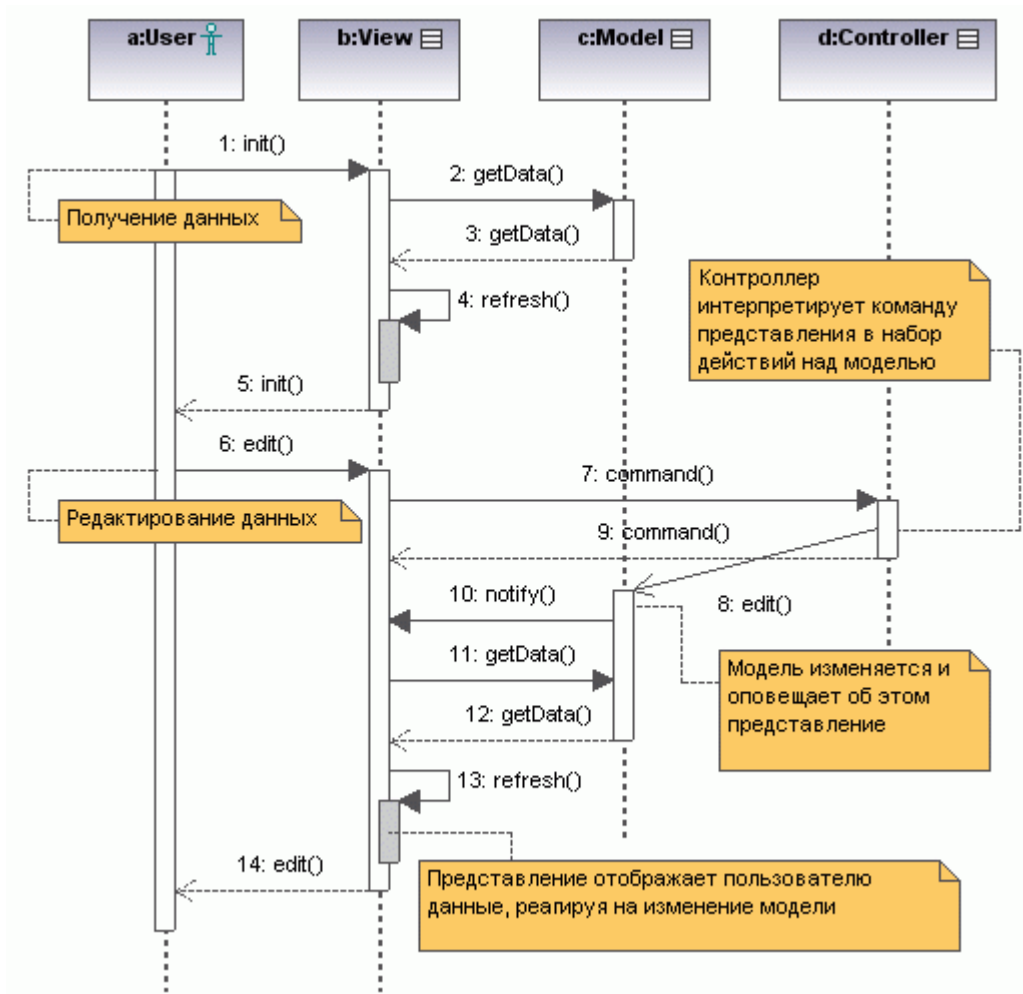


Рисунок 22

Предложенное программистами Smalltalk-80 решение оказалось настолько эффективным, что по прошествии уже почти 30 лет с момента своего появления шаблон проектирования MVC до сих пор является стандартом настольных и Интернет-приложений. В этом легко убедиться – достаточно рассмотреть, насколько MVC представлен в популярных платформах программирования.

11.3.2 Шаблон *Expert*

Шаблон *Expert*, как указывалось выше, может быть представлен описанием проблемы распределения ответственности и описанием ее решения.

Проблема: каков наиболее общий принцип распределения ответственности между объектами при объектно-ориентированном проектировании?

Решение: назначить ответственность информационному эксперту – классу, у которого имеется информация, требуемая для выполнения ответственности.

При распределении ответственности шаблон *Expert* используется гораздо чаще любого другого шаблона. В нем определены основные принципы, которые давно используются в объектно-ориентированном проектировании. Шаблон не содержит неясных или запутанных идей и отражает обычный интуитивно понятный подход. Он заключается в том, что объекты выполняют действия, связанные с имеющейся у них информацией.

Пример: в информационной системе розничной торговли некоторому классу необходимо знать общую сумму продажи. Согласно шаблону *Expert* нужно определить объекты каких классов содержат информацию для вычисления общей суммы. Рассмотрим пример модели поведения, приведенный на Рисунке 23. Для вычисления общей суммы необходимо знать стоимость всех проданных товаров (*CheckLineItem*) и далее просуммировать эти промежуточные суммы. Такой информацией обладает лишь экземпляр объекта *Check*. Поэтому согласно шаблону *Expert* объект *Check* может быть информационным экспертом.

Но в то же время, для вычисления общей суммы (*total*) необходимо вычислить промежуточные суммы (*subtotal*), а для этого необходимо знать *quantity* и *price* каждого товара. Поэтому промежуточную сумму в соответствии с шаблоном *Expert* должен вычислять объект *CheckLineItem*

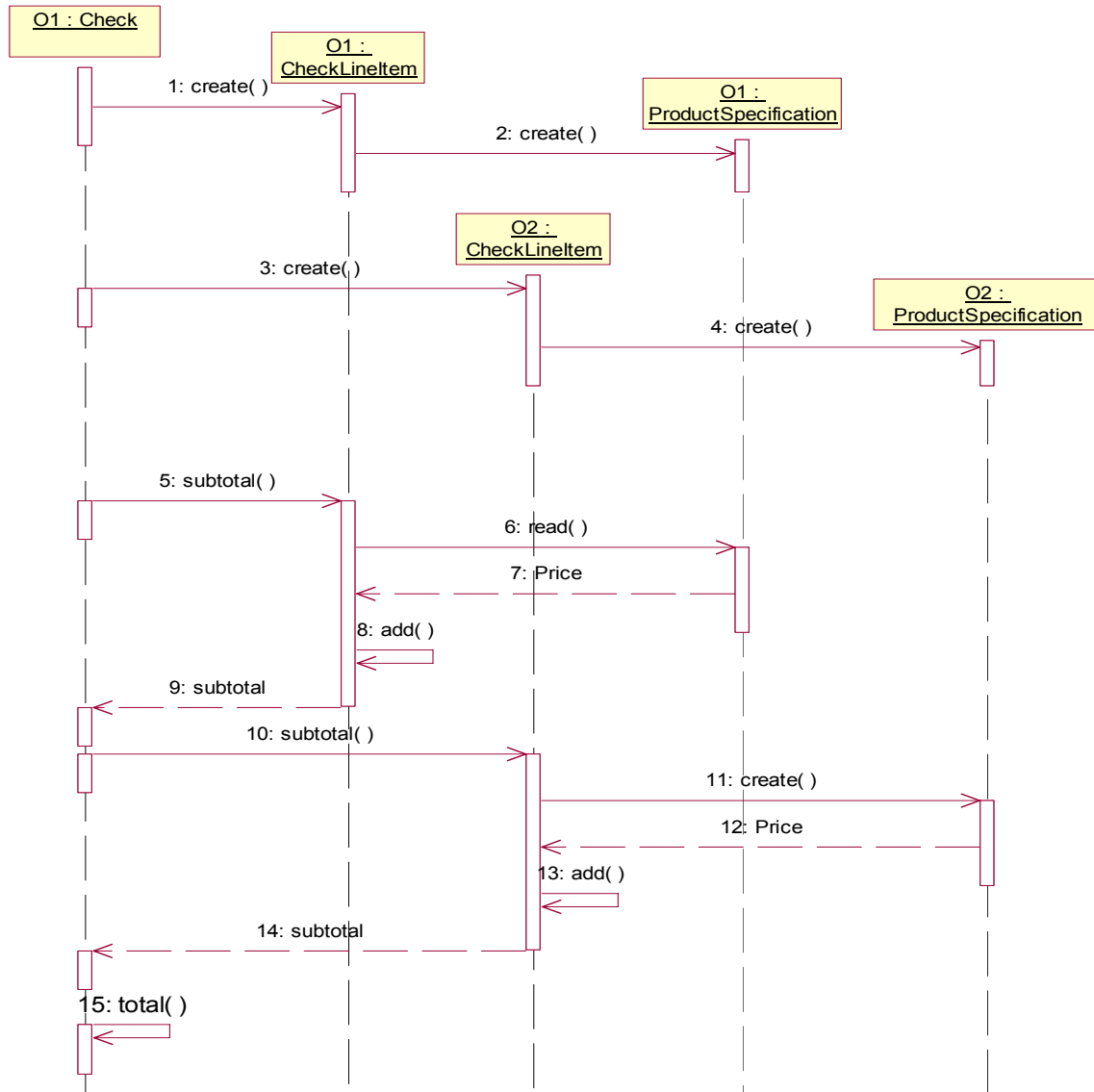


Рисунок 23

11.3.3 Шаблон Controller

Шаблон Controller может быть представлен описанием проблемы распределения ответственности и описанием ее решения.

Проблема: Как распределить ответственность при обработке информационной системой системных событий.

Системное событие это событие высокого уровня, генерируемое внешним исполнителем (событие с внешним входом). Системные события связаны с системными операциями, то есть операциями, выполняемыми системой в ответ на

события. Например, когда кассир нажимает кнопку OutCheck, он генерирует системное событие, свидетельствующее о завершении торговой операции.

Решение: Делегировать обработку системных событий классу, удовлетворяющему одному из следующих условий:

1. Класс представляет всю систему в целом (внешний контроллер)
2. Класс представляет всю организацию (внешний контроллер)
3. Класс представляет активный объект из реального мира (например, роль человека), который может участвовать в решении задачи (контроллер роли)
4. Класс представляет искусственный обработчик всех системных событий в рамках некоторого варианта использования и обычно называется – контроллер прецедента. Для всех системных событий в рамках одного варианта использования применяется один и тот же контроллер

Контроллер это объект, не относящийся к интерфейсу пользователя и отвечающий за обработку системных событий. Он определяет операции для выполнения системных операций.

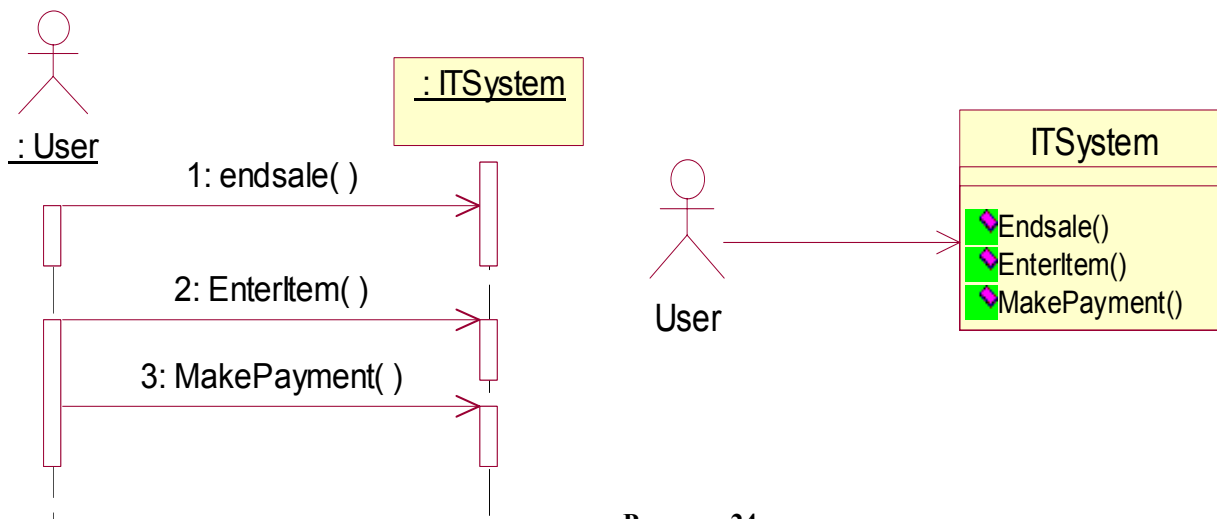


Рисунок 24

11.3.4 Шаблон Polymorphism

Шаблон Polymorphism может быть представлен описанием проблемы распределение обязанностей и описанием ее решения.

Проблема: Как обрабатывать альтернативные варианты поведения на основе типа? Как создавать подключаемые программные компоненты?

Решение: При изменении поведения одного типа (или класса), ответственность распределяется для различных вариантов поведения с помощью полиморфных операций для этого класса.

Пример: Распределить ответственность за санкционирование различных видов платежей в системе розничной торговли.

Поскольку поведение системы авторизации зависит от типа платежа (наличные, чеком, по кредитной карте), согласно шаблону Polymorphism необходимо распределить ответственности по авторизации каждого из типов платежа. Для этого можно использовать полиморфную операцию `authorize()`. Реализации каждой такой операции должны быть различны. Например, объект `CreditPayment` должен взаимодействовать со службой авторизации кредитных платежей.

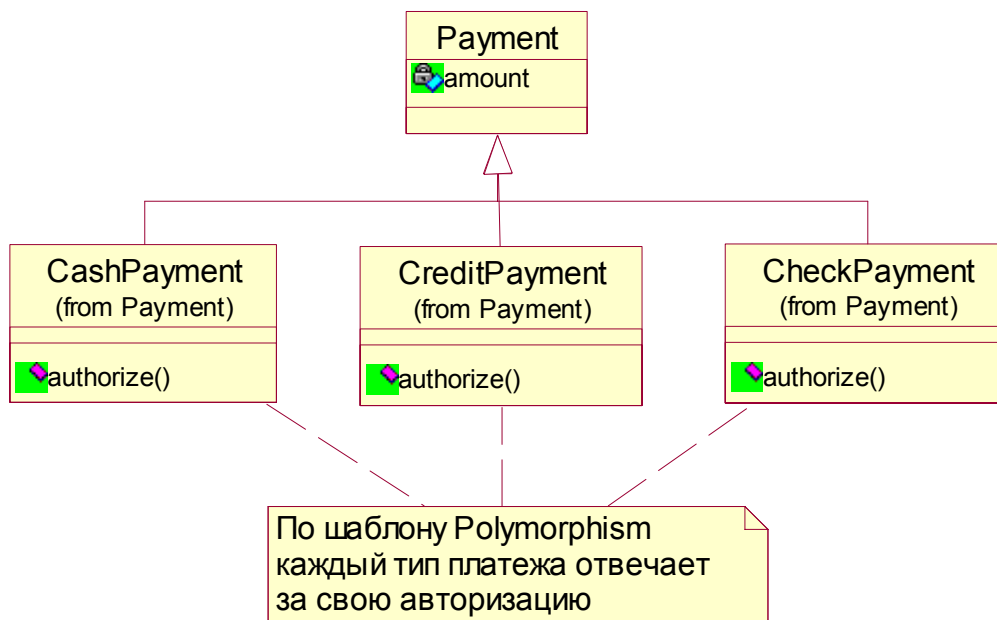


Рисунок 25

11.4 Определение атрибутов класса проектирования

Атрибутом класса проектирования называется представление свойства или признака объекта (класса) в знаковой форме в процессе мысленного выделения из окружающей его среды.

При определении атрибутов следует учитывать следующие положения:

1. Рассмотрите атрибуты классов анализа, трассируемых в классы проектирования. Иногда они соответствуют одному или нескольким атрибутам класса проектирования
2. Доступные типы атрибутов выбираются из предполагаемого к использованию языка программирования
3. Выбирая тип, постарайтесь использовать ранее применяемые типы
4. Одиночный экземпляр атрибута не может совместно использоваться несколькими объектами проектирования. При необходимости атрибут переопределяют в виде отдельного класса
5. При уменьшении понимания класса проектирования по причине сложности его атрибутов, некоторые из его атрибутов можно выделить и переопределить в виде отдельных классов
6. При необходимости рассмотреть все множество атрибутов класса проектирования, можно проиллюстрировать его структуру отдельной диаграммой, указав на ней только атрибуты класса.

11.5 Определение ассоциаций и агрегаций класса проектирования

Ассоциацией называется модель отношения между объектами по какому либо свойству (признаку)

При определении и уточнении ассоциаций следует учитывать следующие общие положения:

1. Ассоциация между классами анализа могут соответствовать одной или нескольким ассоциациям в модели проектирования между классами проектирования
2. Уточните сложность ассоциаций, название ролей, классы ассоциаций, упорядоченные роли, именованные роли и n-арные в плане поддержки этих структур используемым языком программирования. Так, например, имена ролей при генерации кода могут стать атрибутами класса проектирования, зафиксировав таким образом форму имен ролей. Или класс ассоциации может стать новым классом, соединяющим два класса, что потребует новых ассоциаций.
3. Уточните направление просмотра ассоциаций. Воспользуйтесь диаграммой взаимодействий, в которых участвуют эти ассоциации. Направление передачи сообщений между объектами проектирования будет соответствовать направлению просмотра ассоциаций между ассоциированными классами.
4. Взаимодействие объектов классов проектирования требует ассоциаций между соответствующими классами. Сообщения, которыми обмениваются объекты, должны быть внимательно рассмотрены с целью определения необходимых ассоциаций. Экземпляры ассоциаций могут быть использованы для хранения ссылок на другие объекты и группирования объектов в агрегации для отправки им сообщений
5. Количество отношений между классами в модели стремятся свести к минимуму. Ассоциациями и агрегациями моделируются не первоначальные отношения, взятые из реального мира, а отношения созданные в ответ на требования различных реализаций вариантов использования. Кроме того, в ходе проектирования решаются проблемы производительности, следует провести моделирование оптимальных маршрутов поиска среди ассоциаций и агрегаций.

11.6 Определение обобщений класса проектирования

Обобщения используются с той же семантикой, которая определена в применяемом в проекте языке программирования. Если язык программирования не поддерживает обобщений (или наследования), то для посылки сообщений от объекта специализированного класса объектам более общих классов следует использовать ассоциации

11.7 Определение методов класса проектирования

Методы в проектировании используются для реализации операций. Например, метод может определить алгоритм, необходимый для реализации операции. Но в большинстве случаев методы в ходе проектирования не определяются. Они создаются в ходе реализации на выбранном языке программирования.

12. Экстремальные методологии

Как правило, разработка программного обеспечения представляет собой довольно хаотическую деятельность, которую нередко можно охарактеризовать фразой "code and fix" ("пишем и правим"). Именно так работали довольно продолжительное время. Впрочем, всегда была альтернатива – использовать тяжеловесную методологию. Методология превращает создание программного продукта в упорядоченный процесс, с помощью которого можно сделать работу программиста более прогнозируемой и эффективной. Для этого создается детальное описание процесса создания системы, особое место в котором занимает планирование (аналогично другим инженерным дисциплинам).

Основной недостаток методологий «бюрократизм» - чтобы следовать такой методологии, нужно выполнять так много различных предписаний, что замедляется весь темп работ. Именно поэтому их называют тяжеловесными методологиями, (монументальными).

Альтернативой являются экстремальные методологии. Привлекательность этих методологий для многих заключается в отсутствии бюрократизма, присущего монументальным методологиям. Такие методологии представляют собой попытку достичь необходимого компромисса между слишком перегруженным процессом разработки и полным его отсутствием. Иначе говоря, объем процесса разработки в них должен быть достаточен, чтобы получить разумную отдачу (заданное качество).

Основное отличие экстремальных методологий ориентированность на код, то есть основная предпосылка состоит в том, что ключевая часть документации - это исходный код. Но не оно является главным отличием экстремальных методологий от монументальных. Отсутствие документации - это следствие куда более существенных различий.

Проблема состоит в том, что нотация на языке UML может выглядеть очень хорошо на бумаге и при этом содержать в себе серьезные дефекты, которые проявятся только тогда, когда начнется работа по непосредственному программированию. В гражданском строительстве создатели моделей опираются на многолет-

ний опыт, зафиксированный в целой системе правил. Кроме того, базовые положения модели (например, проектирование взаимодействия различных сил) поддаются математическому анализу. В отличие от таких моделей, UML и подобные им диаграммы можно проверить только одним образом - с помощью экспертной оценки. Даже если считать тестирование частью конструирования, то все равно проектирование займет не менее 50% работ.

Отсюда вопрос о сущности проектирования при разработке программных продуктов и его отличиях от проектирования в других областях инженерной деятельности. В процессе решения подобных вопросов было сделано предположение о том, что на самом деле проектным документом может **являться исходный код**, а вся фаза конструирования (разработки) сводится к использованию компилятора и компоновщика. И в самом деле, все, что можно отнести к конструированию, может и должно быть автоматизировано.

Впрочем, не стоит полагать, что без предсказуемого процесса вы погрузитесь в неконтролируемый хаос. Нет, вам просто нужен процесс, который дает контроль над непредсказуемостью. И тут мы вплотную подходим к понятию **адаптивности монументальных технологий под конкретные условия разработки**, категории процессов разработки и т.п.

13. Перечень использованных источников

1. ГОСТ Р ИСО \ МЭК 12207-99
Информационная технология. Процессы жизненного цикла программных средств.
2. ГОСТ Р ИСО \ МЭК 15271-99
Информационная технология. Руководство по применению ГОСТ Р ИСО \ МЭК 12207-99 (процессы жизненного цикла программных средств)
3. ГОСТ Р 34.320-96
Информационная технология. Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы
4. ГОСТ Р 34.321-96
Информационная технология. Система стандартов по базам данных. Эталонная модель управления данными
5. ГОСТ Р ИСО \ МЭК 10746-3-2001
Информационная технология. Взаимосвязь открытых систем. Управление данными и открытая распределенная обработка. Часть 3. Архитектура
6. ГОСТ 34. *. * Информационная технология.
7. А. Якобсон, Г. Буч, Дж. Рамбо
Унифицированный процесс разработки программного обеспечения
Изд-во «Питер», 2002
8. Крэг Ларман
Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование
Изд-во «Вильямс», 2001
9. Описание стандарта 1471-2000 по адресу <http://www.enterprise-architecture.info/Images/Documents/IEEE%201471-2000.pdf>.

14. Приложения

14.1 Приложение 1. Пример текстового описания варианта использования

Прецедент 1. Заказ оборудования сотрудниками отдела крупной компании

Комментарий.

*В этом примере в качестве системы рассматривается компания в целом, то есть мы описываем **бизнес-прецедент**. Пример предназначен в качестве образца заполнения шаблона. В нем развернуты все необходимые разделы.*

Главное действующее лицо. Руководитель отдела

Внешний контекст. Руководитель отдела решает приобрести необходимое оборудование. Оплату оборудования осуществляет компания.

Масштаб. Бизнес-процессы компании. В описании рассматривается общий механизм заказа оборудования, с помощью компьютерной системы или без нее, с точки зрения сотрудников компании.

Уровень. Обобщенный

Заинтересованные лица.

Руководитель отдела: хочет получить необходимое оборудование с минимальными бюрократическим проволочками; хочет пользоваться удобной компьютерной системой в процессе работы с заказом.

Компания: хочет контролировать расходы, но вынуждена приобретать действительно необходимое оборудование.

Производитель оборудования: хочет своевременно получать оплату за отгруженное оборудование.

Исходные условия. Отсутствуют

Минимальный результат. Запрос руководителя фиксируется в планах компании.

Результат успешного завершения. Руководитель отдела получает требуемое оборудование, расходы компании не выходят за согласованный уровень.

Триггер. Руководитель отдела принимает решение о приобретении оборудования.

Основной успешный сценарий.

1. Руководитель отдела: составляет запрос, в котором указывает список необхо-

димого оборудования

2. Заместитель директора: проверяет наличие свободных средств, направляет запрос в отдел снабжения.

3. Отдел снабжения: ищет подходящего поставщика, уточняет цены.

4. Отдел снабжения: оформляет заказ для внешнего поставщика.

6. Производитель: поставляет заказанное оборудование, получает оплату за них (подробности выходят за рамки разрабатываемой системы).

6. Отдел снабжения: оприходует полученное оборудование, передает его руководителю отдела

7. Руководитель отдела: отмечает, что запрос удовлетворен.

Расширения.

1а. Руководитель отдела не знает цену и(или) производителя требуемого оборудования: оставляет соответствующие поля в запросе пустыми.

1b. В любой момент до получения оборудования руководитель отдела может изменить или отменить запрос. Отмена приостанавливает обработку на любой стадии (вопрос: нужно ли удалять данные из системы?). Изменение запроса в сторону уменьшения стоимости оборудования не влияет на процесс обработки запроса, изменение в сторону увеличения стоимости приводит к необходимости утвердить запрос у заместителя директора (шаг 2 основного сценария).

2а. Заместитель директора не знает цену и(или) производителя требуемого оборудования: оставляет соответствующие поля в запросе пустыми, направляет запрос в отдел снабжения.

2b. Заместитель директора не является ответственным за отдел. Может принять решение на себя или отменить запрос.

2с. Нет достаточных свободных средств: запрос возвращается руководителю отдела для изменения или отменяется.

3а. Отдел снабжения обнаруживает часть оборудования на складе компании: передает его руководителю отдела, сокращает соответствующие позиции в запросе.

3b. Отдел снабжения определяет поставщика и цены: возврат к шагу 2.

4а. Требуемое оборудование не может быть произведено одной компанией: отдел

снабжения генерирует несколько заказов, по количеству производителей.

4b. Отдел снабжения объединяет несколько запросов от разных отделов: к заказу составляется приложение, в котором приводится разбиение по отделам.

5a. Производитель не может поставить оборудование в срок: что делать в этом случае?

6a. Частичная поставка: отдел снабжения составляет новый заказ на недопоставленное оборудование

6b. Частичная поставка по объединенному заказу: отдел снабжения составляет новый заказ на недопоставленное оборудование и распределяет полученное между отделами (в спорных случаях может участвовать заместитель директора).

7a. Неверная номенклатура оборудования или плохое качество: руководитель отдела отказывается от получения (что делать в этом случае?)

7b. Руководитель отдела успел уволиться, не дождавшись получения оборудования, отдел снабжения консультируется с руководством, оборудование передается в другой отдел, на склад компании или возвращается производителю.

Варианты изменения технологий и данных. Отсутствуют

Дополнительная информация

Приоритет: средний

Количество вариантов реализации прецедента: несколько

Время ответа: не определяется

Частота: 3 раза в день

Связь с основным действующим лицом: Браузер, E-mail, и их эквиваленты.

Другие действующие лица: производитель продукции

Связь с другими действующими лицами: факс, телефон, автомобиль.

Вопросы для размышления:

Когда информацию об отвергнутом запросе можно полностью удалить из системы?

Какие полномочия необходимы для отмены запроса?

Кто может изменять номенклатуру запроса?

Как обрабатывается история запроса?

14.2 Приложение 2. Пример описания предметной области (по ГОСТ 34.320-96)

В ГОСТ 34.320-96 предметная область называется проблемной областью.

В подразделе 14.2.1 дано словесное описание классификаций и правил для примера проблемной области. Это описание можно рассматривать как **неформальную концептуальную схему**.

В подразделе 14.2.2 приведены примеры сущностей и событий в проблемной области. Их можно рассматривать как **неформальную и неполную информационную базу**, описывающую рассматриваемое пространство сущностей.

14.2.1 Правила и требования

Описываемая проблемная область связана с регистрацией автомобилей и рассматривается с точки зрения Органа Регистрации.

Назначение Органа Регистрации заключается в следующем:

1. знать, кто является или являлся зарегистрированным владельцем автомобиля в любой момент времени от изготовления до уничтожения автомобиля;
2. управлять определенными законами, например связанными с потреблением топлива автомобилями и с передачей права владения ими

Изготовители автомобилей

Существует некоторое число изготовителей, имеющих уникальные имена. Изготовители могут начать функционирование с разрешения Органа Регистрации (которое нельзя отменить). Одновременно могут работать не более пяти изготовителей. Изготовитель может прекратить работу при условии, что он не владеет ни одним автомобилем. В этом случае разрешение на функционирование теряет силу.

Автомобили

Автомобиль является определенной моделью; изготовитель присваивает ему серийный номер, уникальный для машин, сделанных этим изготовителем. Изготовитель регистрируется как владелец автомобиля немедленно. В этот момент присваивается регистрационный номер, уникальный для всех автомобилей и на

все времена. Регистрируется также год выпуска. Только в январе автомобиль может быть объявлен произведенным в предыдущем году. В конечном счете автомобиль уничтожается и регистрируется дата его уничтожения. История автомобиля должна храниться до конца второго календарного года после его уничтожения.

Модели автомобилей

Модель автомобиля имеет одно уникальное имя. Автомобили одной модели производятся только одним изготовителем. Новые модели могут вводиться без ограничений. Считается, что все автомобили одной модели расходуют то же количество топлива.

Расход топлива

Расход топлива — это некоторое число литров углеводородного топлива на 100 км пути, которое находится в пределах 4—25 л. Усредненный расход топлива зарегистрированных автомобилей, произведенных одной фирмой-изготовителем в определенном году, не должен превышать максимального значения, которое устанавливается для всех изготовителей и может меняться от года к году. В конце января каждого года Орган Регистрации посылает соответствующее сообщение каждой фирме-изготовителю, которая не выполнила этого требования.

Гаражи

Существует некоторое число гаражей, каждый с уникальным именем. Могут открываться новые гаражи. Гаражи могут владеть автомобилями, но в любое время автомобили, которыми они владеют, должны поступать от не более чем трех изготовителей (какие три — не имеет значения, и они со временем могут меняться). Гараж не может прекратить торговлю, пока у него есть автомобили.

Лица

Существует ряд лиц, которые могут владеть одним или несколькими автомобилями. Каждый человек имеет уникальное имя. Интерес представляют только те лица, которые владеют или когда-то владели автомобилем, до сих пор известным Органу Регистрации.

Владение автомобилем

В любое время автомобилем может владеть или его фирма-изготовитель,

или торгующий гараж, или некоторое лицо, или группа лиц. Если автомобилем владеет группа лиц, каждое из них считается владельцем.

Передача права владения

Право владения автомобилем передается при регистрации фактической передачи. Фирма-изготовитель может передавать свое право только гаражам и не может выступать в качестве получателя такого права. Гараж может передавать свое право только лицам. После уничтожения автомобиля он не может более никому передаваться. Однако могут регистрироваться более ранние передачи.

Кроме перечисленных, никаких других правил нет. В данном примере неизбежно делается некоторое число упрощающих допущений, например причины, цены и обстоятельства передачи права владения не рассматриваются.

14.2.2 Некоторые факты и события в пространстве сущностей

Ниже описаны некоторые объекты и события в предлагаемом пространстве сущностей.

Форд, Дженерал Моторз, Рено, Фольксваген и Джовет являются фирмами — изготовителями автомобилей, имеющими разрешение выпускать машины. Модели Форда — Мустанг и Гранада. Дженерал Моторз среди прочих производит модель Импала.

История автомобиля модели Мустанг фирмы Форд, серийный номер РСХХ999 такова: он изготовлен в 1975 г. и представлен на регистрацию 21 января 1975 г. Он получил регистрационный номер GMF117. 29 января 1975 г. он был направлен в гараж Смита, который продал его 15 марта 1975 г. г-ну Джонсону. Г-н Бейкер купил автомобиль у г-на Джонсона 24 мая 1978 г. Машина была уничтожена 13 января 1980 г.

Автомобиль модели Импала фирмы Дженерал Моторз, серийный номер QGTM783F, был зарегистрирован под номером ABC 653 9 апреля 1978 г. и направлен в гараж Джоунс Бразерс. Этот автомобиль был куплен г-ном Джонсоном 26 мая 1978 г. Автомобиль был уничтожен 14 августа 1979 г.

Дженерал Моторз выпустила автомобиль модели Импала, серийный номер QAVP864, в 1977 г. Он был зарегистрирован 21 января 1978 г., получив регистра-

ционный номер PQR456. 14 февраля 1978 г. он был продан в гараж RN CARS, который уже владел другими автомобилями фирм Джeneral Моторз, Рено и Фольксваген. Г-н и г-жа Дж. Соуп купили этот автомобиль 31 марта 1978 г., но не смогли продать свой Датсун как часть этой сделки.

В 1978 г. новый изготовитель PSC (PRETTY SMALL CAPS) запросил разрешение на производство, но получил отказ. После разрешения фирмы Джовет 1979 г. запрос был повторен, и на этот раз PSC получил разрешение функционировать с 1 января 1980 г.

Первой выпущенной моделью была Гесмайзер. Первая партия этой модели с серийными номерами GAM1001, GAM1002 и GAM1003 была зарегистрирована 4 января 1980 г. Они получили регистрационные номера XYZ101, XYZ102, XYZ103 соответственно.

Автомобиль XYZ101 был передан в гараж South Station 25 января 1980 г., но был случайно уничтожен в тот же день. Последние два автомобиля были переданы в гараж North Station 20 января 1980 г. Обе машины были проданы г.г. Гедель, Эшер и Бах 26 января 1980 г. Г-н Бах погиб в катастрофе на автомобиле XYZ103 2 марта 1980 г. Автомобиль был зарегистрирован как уничтоженный 5 марта 1980 г. 5 марта 1980 г. г-да Гедель и Эшер были зарегистрированы как владельцы оставшегося автомобиля XYZ102. Они продали этот автомобиль в гараж Смита 15 марта 1980 г., а у него купили Мустанг, серийный номер PCXXX010, который был зарегистрирован 5 января 1980 г. как выпущенный в 1979 г. Регистрационный номер этого автомобиля был XYZ109. XYZ102 был уничтожен путем разборки, т. к. в гараже Смита был дефицит запчастей. Поэтому в конце 1982 г. Орган Регистрации может удалить все сведения об автомобилях XYZ101—XYZ103. 1 декабря 1980 г. фирма PSC дала объявление о свертывании деятельности.

В 1979 г. средний расход топлива был установлен в размере 12 л на 100 км. В 1980 г. этот показатель был 10/100 км, который сохранился и на 1981 г.

14.3 Приложение 3. Концептуальная схема для подхода сущность-атрибут-связь (необходимые высказывания по ГОСТ 34.320-96)

Концептуальная схема представлена списком, который представлен в Таблица 3.

К таблице дается комментарий который приведен ниже.

Таблица 3

№	Необходимое высказывание	Примечание
1.	Проблемная область связана с регистрацией автомобилей и ограничена сферой интересов Органа Регистрации.	
2.	Каждый изготовитель автомобилей имеет уникальное имя.	
3.	Новые изготовители автомобилей могут начать производство при условии, что они имеют разрешение Органа Регистрации.	
4.	Орган Регистрации не может отменить разрешение.	
5.	Одновременно могут работать не более пяти автономных изготовителей.	
6.	Изготовитель может прекратить работу при условии, что он больше не владеет автомобилями.	
7.	Каждый изготовитель автомобилей производит автомобили нескольких моделей	
8.	Автомобиль относится к определенной модели.	
9.	Изготовитель автомобилей присваивает серийный номер каждому выпускаемому автомобилю.	
10.	Этот серийный номер является уникальным для всех автомобилей одного изготовителя.	
11.	Вновь произведенный автомобиль регистрируется Органом Регистрации, как только это окажется возможным	
12.	В этот момент автомобиль регистрируется как принадлежащий изготовителю. Поэтому первым владельцем автомобиля будет его изготовитель	
13.	Только Орган Регистрации может назначить регистрационный номер каждому регистрируемому автомобилю	
14.	Этот регистрационный номер уникален для всех автомобилей в течение всего времени	(E)
15.	Автомобиль имеет год выпуска	
16.	Автомобиль может регистрироваться как произведенный в предыдущем году только в течение января.	
17.	Автомобили могут быть уничтожены, после чего записывается дата уничтожения	
18.	История автомобиля должна храниться до конца второго календарного года с момента его уничтожения. После этого она удаляется	

19.	Имя модели автомобиля уникально для моделей автомобилей в течение всего времени.	(E)
20.	Любая определенная модель автомобиля производится только одним изготовителем	
21.	Время от времени вводятся новые модели.	
22.	Все автомобили одной и той же модели потребляют одинаковое количество топлива.	
23.	Расход топлива должен быть известен Органу Регистрации	(E)
24.	Расход топлива может находиться в пределах 4—25 л на 100 км	
25.	Расход топлива, усредненный по всем автомобилям, произведенным соответствующим изготовителем в определенный год, не должен превышать максимального значения, Одинакового для всех изготовителей	
26.	Максимальный расход топлива может меняться от года к году	
27.	В конце января изготовителю, не выполнившему требование по расходу топлива в предыдущем году, посылается уведомление.	
28.	Каждый гараж имеет уникальное имя	
29.	Могут открываться новые гаражи	
30.	Гаражи могут владеть автомобилями	
31.	В любое время гараж может владеть автомобилями, произведенными не более чем тремя изготовителями (какими именно, не имеет значения, и они могут меняться со временем).	
32.	Существующий гараж может быть закрыт при условии, что в нем нет зарегистрированных за ним автомобилей	
33.	Определенное лицо может иметь один или более автомобилей, зарегистрированных как принадлежащие ему.	
34.	Два или более человека могут иметь в одновременном владении один или несколько автомобилей	
35.	Люди имеют уникальные имена	
36.	Люди известны Органу Регистрации только в том случае, если они имеют или имели один или более автомобилей, которые известны Органу Регистрации.	
37.	В любой момент времени автомобилем владеет его изготовитель или гараж, или некое лицо, или группа лиц, но не вместе две или более из этих категорий.	
38.	Передача права на владение регистрируется, включая дату передачи предыдущего владельца (владельцев) и нового владельца (владельцев).	
39.	Передача права на владение может регистрироваться	

	после уничтожения автомобиля	
40.	Но передача права на владение может регистрироваться после уничтожения автомобиля при условии, что передача права на владение произошла до уничтожения автомобиля	
41.	Каждый изготовитель распределяет новые автомобили нескольким независимым гаражам, каждый из которых может получить автомобили от нескольких изготовителей.	
42.	Поэтому гараж всегда будет вторым владельцем автомобиля	(E)
43.	Изготовители не распределяют автомобили другим изготовителям или непосредственно людям	
44.	Каждый гараж может продавать (т. е. производить передачу зарегистрированного права владения) новые или бывшие в употреблении автомобили и может покупать (т. е. производить передачу зарегистрированного права владения) автомобили у людей	
45.	Гаражам не разрешается продавать автомобили другим гаражам	
46.	Гаражам не разрешается продавать автомобили изготовителям	
47.	Люди могут продавать автомобили друг другу или покупать их друг у друга	
48.		

Замечания по применению подхода сущность—атрибут—связь

- 1) Отметка «(E)» в круглых скобках означает, что концептуальная схема обеспечивает описание утверждения в информационной базе, но это утверждение не может быть выражено в виде правила. Ограничения уникальности в пунктах 14 и 19 поддерживаются, но не на все время. Так, отсутствует правило, которое запрещало бы иметь двум разным автомобилям один и тот же регистрационный номер, но в различные моменты времени.
- 2) Правила установления полномочий не включены (например, пункты 3, 4, 9, 11 и 13).
- 3) Правила проверки достоверности, хотя они и являются статическими, не включены (например, пункты 5, 10, 24, 25, 31, 36 и 39).

- 4) Взаимное исключение связей не включено (например, пункт 37).
- 5) Не включены динамические правила или ограничения, поэтому пункты 6, 20 и 32 не применимы. Заметим, что в пункте 20 не имеется в виду статическое ограничение «любая определенная модель автомобиля производится одним изготовителем в один момент времени».
- 6) Предписывающие правила взаимодействия не являются частью концептуальной схемы (например, пункты 16, 18, 21, 29, 40).

14.4 Приложение 4. Содержание отчета по лабораторной работе

Номер раздела	Название раздела	Содержание раздела
1	Введение	<p>Актуальность темы и описание структуры документа как системы задач, решение которых приводит к результату (логической модели ИС в электронной форме)</p> <p>Актуальность темы раскрывается проблемами, решению которых способствует разрабатываемая информационная система.</p>
2	Описание предметной области	<p>Приводится текстовое описание предметной области. Допускается включение иллюстраций, схем и т.п., раскрывающих текстовое описание. В описание включается описание внешней среды и описание предприятия взаимодействующего с внешней средой. В описании отражаются основные аспекты предметной области: документы, функции, сеть, люди, операционное время и цели (в минимальном составе: бизнес-процессы и отношения между объектами предметной области)</p>
3	Концептуальная модель предметной области	<p>Привести текст необходимых высказываний и затем представить их в виде UML- диаграмм. Набор UML - диаграмм, состоит из следующих диаграмм:</p> <ul style="list-style-type: none"> ▪ Диаграмм вариантов использования, моделирующих функциональную (процессную) структуру предметной области посредством вариантов использования и отношений между ними

		<ul style="list-style-type: none"> ▪ Диаграмм активности, моделирующих алгоритмы ключевых процессов предметной области средствами вариантов использования ▪ Диаграмм классов, моделирующих отношения ключевых объектов средствами диаграмм классов. Причем количество атрибутов у классов должно быть минимально
4	Проблемы предметной области и концепция ИС	<p>В разделе 4.1. приводится текстовое описание основных проблем предметной области. Причем, описывают только те проблемы, которые можно решить с помощью ИС.</p> <p>Концепция ИС – представлена текстовым описанием модели требований к ИС (прообраз технического задания на разработку ИС). Концепция приводится в разделе 4.2. (4.2.1, 4.2.2, 4.2.3)</p>
4.1	Проблемы предметной области	<p>Текстовое описание основных проблем предметной области, которые следуют из анализа диаграмм модели предметной области. Причем, описывать необходимо только те проблемы, которые решаются с помощью ИС</p>
4.2	Концепция ИС	<p>Концепция ИС содержит модель требований, состоящей как минимум из трех подразделов:</p> <p>4.2.1 - Основные понятия, которые должна моделировать система</p> <p>4.2.2 – Функциональные требования (или функциональные возможности), которыми должна удовлетворять ИС для того, чтобы</p>

		<p>успешно решать проблемы</p> <p>4.2.3 - Нефункциональные требования</p> <p>Концепцию ИС иногда называют моделью требований или моделью желаемого результата</p>
4.2.1	Основные понятия ИС	<p>Текстовое описание основных понятий, которые должна моделировать информационная система</p>
4.2.2	Функциональные требования	<p>Текстовое описание функциональных возможностей, которыми должна обладать ИС для того, чтобы успешно решать проблемы</p> <p>Функциональные требования являются ключевым компонентом модели требований. Если следовать положениям RUP, то модель требований есть результат процесса разработки требований, в ходе которого разрабатывается бизнес-модель (предметной области), определяются роли бизнес - акторов и только затем формируются требования к системе в виде вариантов использования ИС.(прецедентов ИС)</p> <p>В отчете, который необходимо составить в результате лабораторной работы, модель требований определяется декларативно в форме высказываний сгруппированный в разделы концепции. Поэтому в отчете не требуется приводить диаграммы вариантов использования для ИС. Они даны декларативно, например, в разделе «функциональные требования» концепции ИС. Но если кто - то из обучаемых приведет их, то это действие будет только поощряться</p>

4.2.3	Нефункциональные требования	Текстовые описания ограничений среды или реализации, производительность, зависимость от платформы, надежность, расширяемость, режим работы и т.п.
5	Концептуальная модель ИС	<p>Набор UML - диаграмм, разработанных согласно обобщенной концептуальной модели (Рисунок 13). UML-диаграммы разрабатываются на основе высказываний концептуальной схемы и информационной базы. Поэтому необходимо привести текст высказываний и затем UML- диаграммы.</p> <p>Рекомендуется разработать диаграммы, отражающие структурный и поведенческий аспект системы, например: диаграмма классов и диаграмма последовательности (Рисунок 18). Причем, диаграмма классов должна содержать классы с минимумом атрибутов.</p> <p>Диаграмма последовательности должна моделировать перечень ключевых функций системы (т.н. ответственностей), которые суть действия, замещающие в предметной области ручные действия, то есть автоматизирующие труд пользователей.</p>
6	Логическая модель ИС	<p>Набор UML - диаграмм, разработанных согласно обобщенной модели проектирования (Рисунок 19).</p> <p>UML-диаграммы разрабатываются на основе концептуальной модели. Причем, сначала разрабатывается модель поведения (модель взаимодействия объектов), с помощью UML диа-</p>

		грамм последовательности, а затем, используя операции, полученные в ходе разработки модели поведения, разрабатывается модель структуры (модель связи классов). Рекомендуется использовать в разработке шаблоны проектирования
6.1	Модель поведения ИС	Модель поведения (модель взаимодействия объектов), с помощью UML диаграмм последовательности. Модель поведения разрабатывается для каждой ключевой функции ИС (варианта использования)
6.2	Модель структуры ИС (взаимосвязи классов)	Является целевой моделью лабораторной работы, разрабатывается на основе концептуальной модели классов. Должна содержать атрибуты и операции, полученные в ходе разработки модели поведения..
7	Реализация модели в среде CASE-средства	Кратко описывается CASE- средство, с помощью которого осуществлялось моделирование. Описание процесса моделирования и получения файла (файлов) модели информационной системы
8	Заключение	В заключении подводится итог, определяются основные достижения.