МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«Казанский (Приволжский) федеральный университет»
Елабужский институт (филиал) КФУ

**УТВЕРЖДАЮ**
Заместитель директора по
образовательной деятельности
_____ С.Ю. Бахвалов
«19» мая 2025 г.
МП

**Программа дисциплины (модуля)**
*Проектирование информационных систем на транспорте*

Направление подготовки/специальность: 15.03.06 Мехатроника и робототехника

Направленность (профиль) подготовки (специальности): Физические основы мехатроники и робототехники

Квалификация: бакалавр

Форма обучения: очная

Язык обучения: русский

Год начала обучения по образовательной программе: - 2025

Содержание

Программу дисциплины разработал(а)(и) доцент, к.н., Галимуллина Э.З. (Кафедра математики и прикладной информатики), EZGalimullina@kpfu.ru

**1. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения ОПОП ВО**

Обучающийся, освоивший дисциплину (модуль), должен обладать следующими компетенциями:

| Шифр компетенции | Расшифровка приобретаемой компетенции |
|---|---|
| ОПК-11 | Способен разрабатывать и применять алгоритмы и современные цифровые программные методы расчетов и проектирования отдельных устройств и подсистем мехатронных и робототехнических систем с использованием стандартных исполнительных и управляющих устройств, средств автоматики, измерительной и вычислительной техники в соответствии с техническим заданием, разрабатывать цифровые алгоритмы и программы управления робототехнических систем |
| ОПК-11.1 | Знать способы разработки и применения алгоритмов и современных цифровых программных методов расчета и проектирования отдельных устройств, и подсистем мехатронных и робототехнических систем с использованием стандартных исполнительных и управляющих устройств, средств автоматики, измерительной и вычислительной техники в соответствии с техническим заданием, основы разработки цифровых алгоритмов и программ управления робототехнических систем |
| ОПК-11.2 | Уметь применять основы разработки и применения алгоритмов и современных цифровых программных методов расчета и проектирования отдельных устройств, и подсистем мехатронных и робототехнических систем с использованием стандартных исполнительных и управляющих устройств, средств автоматики, измерительной и вычислительной техники в соответствии с техническим заданием, основы разработки цифровых алгоритмов и программ управления робототехнических систем |
| ОПК-11.3 | Владеть навыками разработки и применения алгоритмов и современных цифровых программных методов расчета и проектирования отдельных устройств, и подсистем мехатронных и робототехнических систем с использованием стандартных исполнительных и управляющих устройств, средств автоматики, измерительной и вычислительной техники в соответствии с техническим заданием, разработки цифровых алгоритмов и программ управления робототехнических систем |
| ПК-3 | Способен участвовать в научно-исследовательских и опытно-конструкторских работах, проводить теоретические исследования и вычислительные эксперименты с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники |
| ПК-3.1 | Знать методы проведения научно-исследовательских и опытно-конструкторских работ, теоретических исследований и вычислительных экспериментов с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники |
| ПК-3.2 | Уметь применять методы проведения научно-исследовательских и опытно-конструкторских работ, теоретических исследований и вычислительных экспериментов с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники |
| ПК-3.3 | Владеть навыками применения методов проведения научно-исследовательских и опытно-конструкторских работ, теоретических исследований и вычислительных экспериментов с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники |

Обучающийся, освоивший дисциплину (модуль):

Должен знать:
- основные способы разработки и применения информационных систем и цифровых программных методов расчета

и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем;
- основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований и экспериментов с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники.

Должен уметь:
- применять основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем;
- применять основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники.

Должен владеть:
- основными навыками разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем;
- навыками применения основных методов проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники.

**2. Место дисциплины (модуля) в структуре ОПОП ВО**

Данная дисциплина (модуль) включена в Блок 1 "Дисциплины (модули)" Б1.О.05. основной профессиональной образовательной программы 15.03.06 «Мехатроника и робототехника» (Физические основы мехатроники и робототехники)" и относится к обязательной части.
Осваивается на 2 курсе в 3 и 4 семестрах.

**3. Объем дисциплины (модуля) в зачетных единицах с указанием количества часов, выделенных на контактную работу обучающихся с преподавателем (по видам учебных занятий) и на самостоятельную работу обучающихся**

Общая трудоемкость дисциплины составляет 6 зачетных(ые) единиц(ы) на 216 часа(ов).
Контактная работа - 84 часа(ов), в том числе лекции - 42 часа(ов), практические занятия - 0 часа(ов), лабораторные работы - 42 часа(ов), контроль самостоятельной работы - 0 часа(ов).
Самостоятельная работа - 96 часа(ов).
Контроль (зачёт / экзамен) - 36 часа(ов).
Форма промежуточного контроля дисциплины: зачет в 3 семестре, экзамен в 4 семестре.

**4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий**

**4.1 Структура и тематический план контактной и самостоятельной работы по дисциплине (модулю)**

| N | Разделы дисциплины / модуля | Семестр | Виды и часы контактной работы, их трудоемкость (в часах) | | | Самостоятельная работа |
|---|---|---|---|---|---|---|
| | | | Лекции | Практические занятия | Лабораторные работы | |
| 1. | Тема 1. Теоретические основы проектирования информационных систем | 3 | 6 | 0 | 6 | 16 |
| 2. | Тема 2. Методологические основы проектирования | 3 | 6 | 0 | 6 | 16 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | информационных систем | | | | | |
| 3. | Тема 3. Каноническое и типовое проектирование информационных систем | 3 | 6 | 0 | 6 | 18 |
| 4. | Тема 4. Структурные методы анализа и проектирования информационных систем | 3 | 6 | 0 | 6 | 10 |
| 5. | Тема 5. Проектирование документальных баз данных | 4 | 4 | 0 | 4 | 6 |
| 6. | Тема 6. Проектирование фактографических баз данных | 4 | 4 | 0 | 4 | 8 |
| 7. | Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем | 4 | 4 | 0 | 4 | 10 |
| 8. | Тема 8. Обеспечение совместного доступа к базам данных и программам | 4 | 6 | 0 | 6 | 12 |
| | Итого: 216 часов (из них 36 часов контроль) | | 42 | | 42 | 96 |

**4.2 Содержание дисциплины (модуля)**

**Тема 1. Теоретические основы проектирования информационных систем**

Понятие информационной системы. Классификация информационных систем. Архитектура информационных систем. Функциональные и обеспечивающие подсистемы информационной системы. Жизненный цикл информационных систем. Понятия и структура проекта информационных систем. Требования к эффективности и надежности проектных решений.

**Тема 2. Методологические основы проектирования информационных систем**

Основные исторические подходы в проектировании информационных систем, методология построения информационных систем, этапы создания информационных систем. Основные задачи методологии проектирования и области охвата проектирования информационных систем. Технология проектирования информационных систем, основные компоненты технологии проектирования информационных систем. Этапы создания информационных систем. Модели жизненного цикла информационных систем. Методы и средства проектирования информационных систем. Характеристика применяемых технологий проектирования информационных систем, требования к выбираемой технологии проектирования.

**Тема 3. Каноническое и типовое проектирование информационных систем**

Стадии и этапы процесса проектирования информационных систем. Состав работ на предпроектной стадии, стадии технического и рабочего проектирования, стадии внедрения, эксплуатации и сопровождения проекта. Понятие типового элемента. Технологии параметрически-ориентированного и модельно-ориентированного проектирования.

**Тема 4. Структурные методы анализа и проектирования информационных систем**

Проблема сложности и подходы к ее решению при проектировании информационных систем. Общие принципы проектирования информационных систем (2 основных подхода к декомпозиции систем). Визуальное моделирование (ERD, DFD, STD, SADT и др. нотации). Виды моделей. CASE-технологии и CASE-средства. Структурные методы анализа и проектирования информационных систем. Характеристики методов. Принципы структурного метода. Метод функционального моделирования IDEF0 (I cam DEFinition). Особенности топологии описания системы. Границы и связи. Области применения IDEF0. Методология описания бизнес-процессов IDEF3. Единица работы (действий). Связи. Соединения. Области применения IDEF3. Структурный анализ потоков данных DFD. Области применения DFD - диаграмм.

**Тема 5. Проектирование документальных баз данных**

Документальные базы данных. Проектирование документальных баз данных: анализ предметной области, разработка состава и структуры баз данных, проектирование логико-семантического комплекса. Анализ особенностей документальных баз данных. Сложные запросы. Поиск по словарной близости. Использование функции спецификации полей при поиске.

**Тема 6. Проектирование фактографических баз данных**

Фактографические базы данных. Основные функции фактографических баз данных. Проектирование фактографических баз данных: методы проектирования; концептуальное, логическое и физическое проектирование. Моделирование структур данных. Метод IDEF1X. Базовые понятия. Диаграмма "сущность-связь (ERD)" и области ее применения.

**Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем**

Базовые принципы и понятия технологии разработки объектно-ориентированных информационных систем на основе UML. Возможности и достоинства UML. Инструментальные средства визуального моделирования. Архитектурные представления UML. Понятие архитектуры информационной системы. Виды представлений. Обзор диаграмм UML. Средства UML. Диаграммы вариантов использования. Диаграммы классов. Диаграммы взаимодействия (Диаграммы последовательности, кооперативные диаграммы). Диаграммы состояний. Диаграммы деятельностей. Диаграммы реализации (диаграммы пакетов, компонентов и размещения). Рациональный унифицированный процесс (RUP). Дисциплины RUP. Ключевые понятия и принципы RUP. Общее представление, динамический и статический аспекты RUP.

**Тема 8. Обеспечение совместного доступа к базам данных и программам**

Межсистемные интерфейсы и драйверы; интерфейсы в распределенных системах. Стандартные методы совместного доступа к базам данных и программам в сложных информационных системах (драйверы ODBC (Open Data Base Connectivity), программная система CORBA (Common Object Request Broker Architecture - общедоступная архитектура с брокером при запросе объекта).

**5. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)**

Самостоятельная работа обучающихся выполняется по заданию и при методическом руководстве преподавателя, но без его непосредственного участия. Самостоятельная работа подразделяется на самостоятельную работу на аудиторных занятиях и на внеаудиторную самостоятельную работу. Самостоятельная работа обучающихся включает как полностью самостоятельное освоение отдельных тем (разделов) дисциплины, так и проработку тем (разделов), осваиваемых во время аудиторной работы. Во время самостоятельной работы обучающиеся читают и конспектируют учебную, научную и справочную литературу, выполняют задания, направленные на закрепление знаний и отработку умений и навыков, готовятся к текущему и промежуточному контролю по дисциплине.

Организация самостоятельной работы обучающихся регламентируется нормативными документами, учебно-методической литературой и электронными образовательными ресурсами, включая:

Порядок организации и осуществления образовательной деятельности по образовательным программам высшего образования – программам бакалавриата, программам специалитета, программам магистратуры (утвержденный приказом Министерства науки и высшего образования Российской Федерации от 6 апреля 2021 года № 245)

Порядок организации и осуществления образовательной деятельности по образовательным программам высшего образования - программам бакалавриата, программам специалитета, программам магистратуры (утвержден приказом Министерства образования и науки Российской Федерации от 5 апреля 2017 года №301)

Письмо Министерства образования Российской Федерации №14-55-996ин/15 от 27 ноября 2002 г. "Об активизации самостоятельной работы студентов высших учебных заведений"

Устав федерального государственного автономного образовательного учреждения "Казанский (Приволжский) федеральный университет"

Правила внутреннего распорядка федерального государственного автономного образовательного учреждения высшего профессионального образования "Казанский (Приволжский) федеральный университет"

Локальные нормативные акты Казанского (Приволжского) федерального университета

**6. Фонд оценочных средств по дисциплине (модулю)**

Фонд оценочных средств по дисциплине (модулю) включает оценочные материалы, направленные на проверку освоения компетенций, в том числе знаний, умений и навыков. Фонд оценочных средств включает оценочные средства текущего контроля и оценочные средства промежуточной аттестации.

В фонде оценочных средств содержится следующая информация:

- соответствие компетенций планируемым результатам обучения по дисциплине (модулю);
- критерии оценивания сформированности компетенций;
- механизм формирования оценки по дисциплине (модулю);
- описание порядка применения и процедуры оценивания для каждого оценочного средства;
- критерии оценивания для каждого оценочного средства;
- содержание оценочных средств, включая требования, предъявляемые к действиям обучающихся, демонстрируемым результатам, задания различных типов.

Фонд оценочных средств по дисциплине находится в Приложении 1 к программе дисциплины (модулю).

**7. Перечень литературы, необходимой для освоения дисциплины (модуля)**

Освоение дисциплины (модуля) предполагает изучение учебной литературы. Литература может быть доступна обучающимся в одном из двух вариантов (либо в обоих из них):

- в электронном виде - через электронные библиотечные системы на основании заключенных КФУ договоров с правообладателями;
- в печатном виде - в Научной библиотеке Елабужского института КФУ. Обучающиеся получают учебную литературу на абонементе по читательским билетам в соответствии с правилами пользования Научной библиотекой.

Электронные издания доступны дистанционно из любой точки при введении обучающимся своего логина и пароля от личного кабинета в системе "Электронный университет". При использовании печатных изданий библиотечный фонд должен быть укомплектован ими из расчета не менее 0,25 экземпляра на каждого обучающегося из числа лиц, одновременно осваивающих данную дисциплину

Перечень литературы, необходимой для освоения дисциплины (модуля), находится в Приложении 2 к рабочей программе дисциплины. Он подлежит обновлению при изменении условий договоров КФУ с правообладателями электронных изданий и при изменении комплектования фондов Научной библиотеки Елабужского института КФУ.

**8. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)**

Проектирование информационной системы - https://finswin.com/projects/proektirovanie/informacionnyh-sistem.html

Проектирование информационных систем - https://compress.ru/article.aspx?id=11764

Проектирование информационных систем (ИНТУИТ) - https://www.intuit.ru/studies/courses/2195/55/info

**9. Методические указания для обучающихся по освоению дисциплины (модуля)**

| Вид работ | Методические рекомендации |
|---|---|
| лекции | Лекционные занятия проводятся с использованием интерактивных технологий и предполагают активное участие студентов. Для подготовки к занятиям рекомендуется выделять в материале проблемные вопросы, затрагиваемые преподавателем в лекции, и группировать информацию вокруг них. Желательно выделять в используемой литературе постановки вопросов, на которые разными авторам могут быть даны различные ответы. На основании постановки таких вопросов следует собирать аргументы в пользу различных вариантов решения поставленных проблем. |
| лабораторные работы | Лабораторные занятия - это одна из разновидностей практического занятия, являющаяся эффективной формой учебных занятий в организации высшего образования. Лабораторные занятия имеют выраженную специфику в зависимости от учебной дисциплины, углубляют и закрепляют теоретические знания. На этих занятиях студенты осваивают конкретные методы изучения дисциплины, обучаются экспериментальным способам анализа, умению работать с приборами и современным оборудованием. Лабораторные занятия дают наглядное представление об изучаемых явлениях и процессах, студенты осваивают постановку и ведение эксперимента, учатся умению наблюдать, оценивать полученные результаты, делать выводы и обобщения. Отчёт по итогам выполненных лабораторных работ выполняется на листах белой бумаги формата А4 в печатном или рукописном виде. При оформлении отчёта используется сквозная нумерация страниц, считая титульный лист первой страницей. Номер страницы на титульном листе не ставится. Номера страницы ставятся по центру вверху. При оформлении отчёта в печатном виде желательно соблюдать следующие требования. Для заголовков: полужирный шрифт, 14 пт, центрированный. Для основного текста: нежирный шрифт, 14 пт, выравнивание по ширине. Во всех случаях тип шрифта - Times New Roman, отступ абзаца 1.25 см, полуторный междустрочный интервал. Поля: левое - 3 см, правое - 1 см, верхнее и нижнее - 2 см. Отчет должен содержать следующие элементы: 1) Титульный лист с обязательным указанием варианта; 2) Цель работы; 3) Задание; 4) Основная часть; 5) Вывод. |
| самостоя-тельная работа | Самостоятельная работа студентов по дидактической сути представляет собой комплекс условий обучения, организуемых преподавателем и направленных на самоподготовку учащихся. Учебная деятельность протекает без непосредственного участия преподавателя и заключается в проработке лекционного материала, подготовке к лабораторным занятиям; изучении учебной литературы из основного и дополнительного списка. |
| зачет | Зачет является формой оценки качества освоения студентом образовательной программы по дисциплине. По результатам зачета студенту выставляется оценка "зачтено" или "не зачтено". Зачет может проводиться в форме устного опроса по билетам (вопросам) или без билетов, с предварительной подготовкой или без подготовки, по усмотрению кафедры. Преподаватель может проставить зачет без опроса или собеседования тем студентам, которые активно участвовали на практических занятиях. |
| экзамен | Экзамен нацелен на комплексную проверку освоения дисциплины. Экзамен проводится в устной или письменной форме по билетам, в которых содержатся вопросы (задания) по всем темам курса. Обучающемуся даётся время на подготовку. Оценивается владение материалом, его системное освоение, способность применять нужные знания, навыки и умения при анализе проблемных ситуаций и решении практических заданий. |

**10. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения и информационных справочных систем (при необходимости)**

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения и информационных справочных систем, представлен в Приложении 3 к рабочей программе дисциплины (модуля).

**11. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)**

Материально-техническое обеспечение образовательного процесса по дисциплине (модулю) включает в себя следующие компоненты:

Учебная аудитория для проведения учебных занятий лекционного типа, семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации № 61

Комплект мебели для преподавателя – 1 шт., посадочные места для обучающихся – 30 шт., одноместные столы – 12 шт., компьютерные столы – 18 шт., компьютеры – 19 шт., интерактивная панель – 1 шт., меловая доска настенная – 1 шт., выход в интернет, внутривузовская компьютерная сеть, доступ в электронную информационно-

образовательную среду.

Помещение для самостоятельной работы № 10
Посадочные места для пользователей – 28 шт., металлические двусторонние стеллажи для книг – 11 шт., книжный шкаф открытый – 5 шт., проектор – 1 шт., ноутбуки для пользователей – 11 шт., шкаф каталожный – 8 шт., шкаф для одежды – 1 шт., ксерокс – 1 шт., рабочий стол библиотекаря – 1 шт., компьютер библиотекаря – 1 шт., вешалка для одежды – 1 шт., жалюзи рулонные «Омега» с фотопечатью – 4 шт., стенд настенный (бронированное стекло) – 4 шт., шкаф-витрина встроенный в арку – 2 шт., шкаф-витрина стеклянный – 2 шт., стеллаж трубчатый с деревянными полками – 2 шт., рабочий стол для инвалидов и лиц с ОВЗ – 2 шт., стол СИ-1 рабочий для инвалидов-колясочников – 1 шт., компьютер– 2 шт., наушники – 2 шт., устройство «Говорящая книга» (тифлоплеер) – 2 шт., видеоувеличитель – 2 шт., радиокласс – 1 шт., портативный тактильный дисплей - 1 шт., сканирующая читающая машина - 1 шт., сканер – 1 шт., веб-камера – 1 шт., выход в интернет, внутривузовская компьютерная сеть, доступ в электронную информационно-образовательную среду.

**12. Средства адаптации преподавания дисциплины к потребностям обучающихся инвалидов и лиц с ограниченными возможностями здоровья**

При необходимости в образовательном процессе применяются следующие методы и технологии, облегчающие восприятие информации обучающимися инвалидами и лицами с ограниченными возможностями здоровья:

- создание текстовой версии любого нетекстового контента для его возможного преобразования в альтернативные формы, удобные для различных пользователей;

- создание контента, который можно представить в различных видах без потери данных или структуры, предусмотреть возможность масштабирования текста и изображений без потери качества, предусмотреть доступность управления контентом с клавиатуры;

- создание возможностей для обучающихся воспринимать одну и ту же информацию из разных источников - например, так, чтобы лица с нарушениями слуха получали информацию визуально, с нарушениями зрения - аудиально;

- применение программных средств, обеспечивающих возможность освоения навыков и умений, формируемых дисциплиной, за счёт альтернативных способов, в том числе виртуальных лабораторий и симуляционных технологий;

- применение дистанционных образовательных технологий для передачи информации, организации различных форм интерактивной контактной работы обучающегося с преподавателем, в том числе вебинаров, которые могут быть использованы для проведения виртуальных лекций с возможностью взаимодействия всех участников дистанционного обучения, проведения семинаров, выступления с докладами и защиты выполненных работ, проведения тренингов, организации коллективной работы;

- применение дистанционных образовательных технологий для организации форм текущего и промежуточного контроля;

- увеличение продолжительности сдачи обучающимся инвалидом или лицом с ограниченными возможностями здоровья форм промежуточной аттестации по отношению к установленной продолжительности их сдачи:

- продолжительности сдачи зачёта или экзамена, проводимого в письменной форме, - не более чем на 90 минут;

- продолжительности подготовки обучающегося к ответу на зачёте или экзамене, проводимом в устной форме, - не более чем на 20 минут;

- продолжительности выступления обучающегося при защите курсовой работы - не более чем на 15 минут.

Программа составлена в соответствии с требованиями ФГОС ВО и учебным планом по направлению 15.03.06 «Мехатроника и робототехника» и профилю подготовки "Физические основы мехатроники и робототехники".

*Приложение №1*
*к рабочей программе дисциплины (модуля)*
*Б1.О.05.05 Проектирование информационных систем на транспорте*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
"Казанский (Приволжский) федеральный университет"
Елабужский институт (филиал) КФУ

**Фонд оценочных средств по дисциплине (модулю)**
**Б1.О.05.05 Проектирование информационных систем на транспорте**

Направление подготовки: 15.03.06 Мехатроника и робототехника
Профиль подготовки: Физические основы мехатроники и робототехники
Квалификация выпускника: бакалавр
Форма обучения: очная
Язык обучения: русский
Год начала обучения по образовательной программе: 2025

**СОДЕРЖАНИЕ**

**1. Соответствие компетенций планируемым результатам обучения по дисциплине (модулю)**

| **Код и наименование компетенции** | **Индикаторы достижения компетенций для данной дисциплины** | **Оценочные средства текущего контроля и промежуточной аттестации** |
|---|---|---|
| ОПК-11 Способен разрабатывать и применять алгоритмы и современные цифровые программные методы расчетов и проектирования отдельных устройств и подсистем мехатронных и робототехнических систем с использованием стандартных исполнительных и управляющих устройств, средств автоматики, измерительной и вычислительной техники в соответствии с техническим заданием, разрабатывать цифровые алгоритмы и программы управления робототехнических систем | Знать основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем.<br><br>Уметь применять основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем.<br><br>Владеть основными навыками разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем | Текущий контроль:<br>Лабораторные работы по темам<br>Тема 1. Теоретические основы проектирования информационных систем<br>Тема 2. Методологические основы проектирования информационных систем<br>Тема 3. Каноническое и типовое проектирование информационных систем<br>Тема 4. Структурные методы анализа и проектирования информационных систем<br>Тема 6. Проектирование фактографических баз данных<br>Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем<br><br>Тестирование по темам<br>Тема 1. Теоретические основы проектирования информационных систем<br>Тема 2. Методологические основы проектирования информационных систем<br>Тема 3. Каноническое и типовое проектирование информационных систем<br>Тема 4. Структурные методы анализа и проектирования информационных систем<br>Тема 5. Проектирование документальных баз данных<br>Тема 6. Проектирование фактографических баз данных<br>Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем<br>Тема 8. Обеспечение совместного доступа к базам данных и программам<br><br>Реферат по темам<br>Тема 1. Теоретические основы проектирования информационных систем<br>Тема 2. Методологические основы проектирования информационных систем<br>Тема 3. Каноническое и типовое проектирование информационных систем<br>Тема 4. Структурные методы анализа и проектирования информационных систем<br>Тема 5. Проектирование документальных баз данных<br>Тема 6. Проектирование фактографических баз данных<br>Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем<br>Тема 8. Обеспечение совместного доступа к базам данных и программам<br><br>Промежуточная аттестация:<br>Зачет, Экзамен |
| ПК-3 Способен участвовать в научно-исследовательских и опытно-конструкторских работах, проводить теоретические | Знать основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований и экспериментов с использованием стандартных программных средств с целью получения математических | Текущий контроль:<br>Лабораторные работы по темам<br>Тема 1. Теоретические основы проектирования информационных систем<br>Тема 2. Методологические основы проектирования информационных систем<br>Тема 3. Каноническое и типовое |

| | | |
|---|---|---|
| исследования и вычислительные эксперименты с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники | моделей процессов и объектов мехатроники и робототехники.<br><br>Уметь применять основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники.<br><br>Владеть навыками применения основных методов проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники | проектирование информационных систем<br>Тема 4. Структурные методы анализа и проектирования информационных систем<br>Тема 6. Проектирование фактографических баз данных<br>Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем<br><br>Тестирование по темам<br>Тема 1. Теоретические основы проектирования информационных систем<br>Тема 2. Методологические основы проектирования информационных систем<br>Тема 3. Каноническое и типовое проектирование информационных систем<br>Тема 4. Структурные методы анализа и проектирования информационных систем<br>Тема 5. Проектирование документальных баз данных<br>Тема 6. Проектирование фактографических баз данных<br>Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем<br>Тема 8. Обеспечение совместного доступа к базам данных и программам<br><br>Реферат по темам<br>Тема 1. Теоретические основы проектирования информационных систем<br>Тема 2. Методологические основы проектирования информационных систем<br>Тема 3. Каноническое и типовое проектирование информационных систем<br>Тема 4. Структурные методы анализа и проектирования информационных систем<br>Тема 5. Проектирование документальных баз данных<br>Тема 6. Проектирование фактографических баз данных<br>Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем<br>Тема 8. Обеспечение совместного доступа к базам данных и программам<br><br>Промежуточная аттестация:<br>Зачет, Экзамен |

**2. Критерии оценивания сформированности компетенций**

| **Компетенция** | **Зачтено** | | | **Не зачтено** |
|---|---|---|---|---|
| | **Высокий уровень (отлично) (86-100 баллов)** | **Средний уровень (хорошо) (71-85 баллов)** | **Низкий уровень (удовлетворительно) (56-70 баллов)** | **Ниже порогового уровня (неудовлетворительно) (0-55 баллов)** |
| *ОПК-11* | Знает основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и | Знает основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и | Знает основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и | Не знает основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и |

| | | | | |
|---|---|---|---|---|
| | робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем | робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем. Допускает незначительные ошибки при ответе на вопрос или решении поставленной задачи | робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем. Допускает типичные ошибки при ответе на вопрос или решении поставленной задачи | робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем |
| | Умеет применять основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем | Умеет применять основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем. Допускает незначительные ошибки при ответе на вопрос или решении поставленной задачи | Умеет применять основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем. Допускает типичные ошибки при ответе на вопрос или решении поставленной задачи | Не умеет применять основные способы разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем |
| | Владеет основными навыками разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем | Владеет основными навыками разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем. Допускает незначительные ошибки при ответе на вопрос или решении поставленной задачи | Владеет основными навыками разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем. Допускает типичные ошибки при ответе на вопрос или решении поставленной задачи | Не владеет основными навыками разработки и применения информационных систем и цифровых программных методов расчета и проектирования отдельных устройств мехатронных и робототехнических систем, основы разработки цифровых алгоритмов и программ управления робототехнических систем |
| ПК-3 | Знает основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований и экспериментов с использованием стандартных | Знает основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований и экспериментов с использованием стандартных | Знает основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований и экспериментов с использованием стандартных | Не знает основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований и |

| | | | | |
|---|---|---|---|---|
| | программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники | программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники. Допускает незначительные ошибки при ответе на вопрос или решении поставленной задачи | программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники. Допускает типичные ошибки при ответе на вопрос или решении поставленной задачи | экспериментов с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники |
| | Умеет применять основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники | Умеет применять основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники. Допускает незначительные ошибки при ответе на вопрос или решении поставленной задачи | Умеет применять основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники. Допускает типичные ошибки при ответе на вопрос или решении поставленной задачи | Не умеет применять основные методы проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники |
| | Владеет навыками применения основных методов проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники | Владеет навыками применения основных методов проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники. Допускает незначительные ошибки при ответе на вопрос или решении поставленной задачи | Владеет навыками применения основных методов проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники. Допускает типичные ошибки при ответе на вопрос или решении поставленной задачи | Не владеет навыками применения основных методов проведения научно-исследовательских работ, проектирования информационных систем, теоретических исследований с использованием стандартных программных средств с целью получения математических моделей процессов и объектов мехатроники и робототехники |

**3. Распределение оценок за формы текущего контроля и промежуточную аттестацию**

3 семестр:
**Текущий контроль**:
Лабораторные работы по темам
Тема 1. Теоретические основы проектирования информационных систем
Тема 2. Методологические основы проектирования информационных систем
Тема 3. Каноническое и типовое проектирование информационных систем
Тема 4. Структурные методы анализа и проектирования информационных систем
Максимальное количество баллов по БРС - 20.

Тестирование по темам
Тема 1. Теоретические основы проектирования информационных систем
Тема 2. Методологические основы проектирования информационных систем
Тема 3. Каноническое и типовое проектирование информационных систем
Тема 4. Структурные методы анализа и проектирования информационных систем

Максимальное количество баллов по БРС - 10.

Реферат по темам
Тема 1. Теоретические основы проектирования информационных систем
Тема 2. Методологические основы проектирования информационных систем
Тема 3. Каноническое и типовое проектирование информационных систем
Тема 4. Структурные методы анализа и проектирования информационных систем
Максимальное количество баллов по БРС - 20.
Итого 20+10+20= 50 баллов

**Промежуточная аттестация** - зачет - 50 баллов.
Промежуточная аттестация проводится после завершения изучения дисциплины или ее части в форме, определяемой учебным планом образовательной программы с целью оценить работу обучающегося, степень усвоения теоретических знаний, уровень сформированности компетенций.
Преподаватель, принимающий зачет обеспечивает случайное распределение вариантов зачетных заданий между обучающимися с помощью билетов и/или с применением компьютерных технологий; вправе задавать обучающемуся дополнительные вопросы и давать дополнительные задания помимо тех, которые указаны в билете.
Зачет проводится по билетам. В каждом билете оценочные средства одного вида: устный или письменный ответ на вопрос.
Устный или письменный ответ – 50 баллов.

Итого 50 баллов.

Общее количество баллов по дисциплине за текущий контроль и промежуточную аттестацию: 50+50=100 баллов.
Соответствие баллов и оценок:

**Для зачета**:
56-100 – зачтено
0-55 – не зачтено

4 семестр:
**Текущий контроль**:
Лабораторные работы по темам
Тема 6. Проектирование фактографических баз данных
Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем
Максимальное количество баллов по БРС - 20.

Тестирование по темам
Тема 6. Проектирование фактографических баз данных
Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем
Тема 8. Обеспечение совместного доступа к базам данных и программам
Максимальное количество баллов по БРС - 10.

Реферат по темам
Тема 1. Теоретические основы проектирования информационных систем
Тема 2. Методологические основы проектирования информационных систем
Тема 3. Каноническое и типовое проектирование информационных систем
Тема 4. Структурные методы анализа и проектирования информационных систем
Максимальное количество баллов по БРС - 20.
Итого 20+10+20= 50 баллов

**Промежуточная аттестация** - экзамен - 50 баллов.
Промежуточная аттестация проводится после завершения изучения дисциплины или ее части в форме, определяемой учебным планом образовательной программы с целью оценить работу обучающегося, степень усвоения теоретических знаний, уровень сформированности компетенций.
Преподаватель, принимающий экзамен, обеспечивает случайное распределение вариантов заданий между обучающимися с помощью билетов и/или с применением компьютерных технологий; вправе задавать обучающемуся дополнительные вопросы и давать дополнительные задания помимо тех, которые указаны в билете.
Экзамен проводится по билетам. В каждом билете оценочные средства одного вида: устный или письменный ответ на вопрос.
Устный или письменный ответ – 50 баллов.

Итого 50 баллов.

Общее количество баллов по дисциплине за текущий контроль и промежуточную аттестацию: 50+50=100 баллов.
Соответствие баллов и оценок:
Для экзамена:

86-100 – отлично
71-85 – хорошо
56-70 – удовлетворительно
0-55 – неудовлетворительно

**4. Оценочные средства, порядок их применения и критерии оценивания**
**4.1. Оценочные средства текущего контроля**
**4.1.1. Лабораторные работы**
3 семестр
Тема 1. Теоретические основы проектирования информационных систем
Тема 2. Методологические основы проектирования информационных систем
Тема 3. Каноническое и типовое проектирование информационных систем
Тема 4. Структурные методы анализа и проектирования информационных систем
4 семестр
Тема 6. Проектирование фактографических баз данных
Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем

**4.1.1.1. Порядок проведения.**
В аудитории, оснащённой соответствующим оборудованием, обучающиеся проводят учебные эксперименты и тренируются в применении практико-ориентированных технологий. Оцениваются знание материала и умение применять его на практике, умения и навыки по работе с оборудованием в соответствующей предметной области.
Лабораторные работы по дисциплине «Технология изготовления авторской куклы» проводятся преподавателем согласно разработанному и утвержденному на кафедре рабочей программе. Каждая лабораторно-практическая работа выполняется по определенной теме программы в соответствии с заданием.
Перед выполнением каждой работы студенты-бакалавры должны проработать соответствующий материал, используя конспекты теоретических занятий, периодические издания, учебно-методические пособия и учебники
На каждом занятии студенты выполняют работу в соответствии с ее содержанием и методическими указаниями.
По окончании занятий студенты оформляют отчет по каждой работе, соблюдая следующую форму:
- Наименование темы;
- Цель работы;
- Задание и содержание выполненной работы,
- Письменные ответы на контрольные вопросы.
- Выводы по проделанной работе.
- Список использованных источников.

**4.1.1.2. Критерии оценивания**

**17-20 баллов ставится, если обучающийся:**
Оборудование и методы использовал правильно. Проявлена превосходная теоретическая подготовка. Необходимые навыки и умения полностью освоены. Результат лабораторной работы полностью соответствует её целям.
**14-16 баллов ставится, если обучающийся:**
Оборудование и методы использовал в основном правильно. Проявлена хорошая теоретическая подготовка. Необходимые навыки и умения в основном освоены. Результат лабораторной работы в основном соответствует её целям.
**11-15 баллов ставится, если обучающийся:**
Оборудование и методы частично использовал правильно. Проявлена удовлетворительная теоретическая подготовка. Необходимые навыки и умения частично освоены. Результат лабораторной работы частично соответствует её целям.
**0--10 баллов ставится, если обучающийся:**
Оборудование и методы использовал неправильно. Проявлена неудовлетворительная теоретическая подготовка. Необходимые навыки и умения не освоены. Результат лабораторной работы не соответствует её целям.

**4.1.1.3. Содержание оценочного средства**

**Примеры лабораторных работ**

Формулировка задания

Лабораторная работа 1. Создание контекстной диаграммы.

Разработать контекстную TOP диаграмму АО «Компреал»

Создание контекстной диаграммы АО «Компреал»

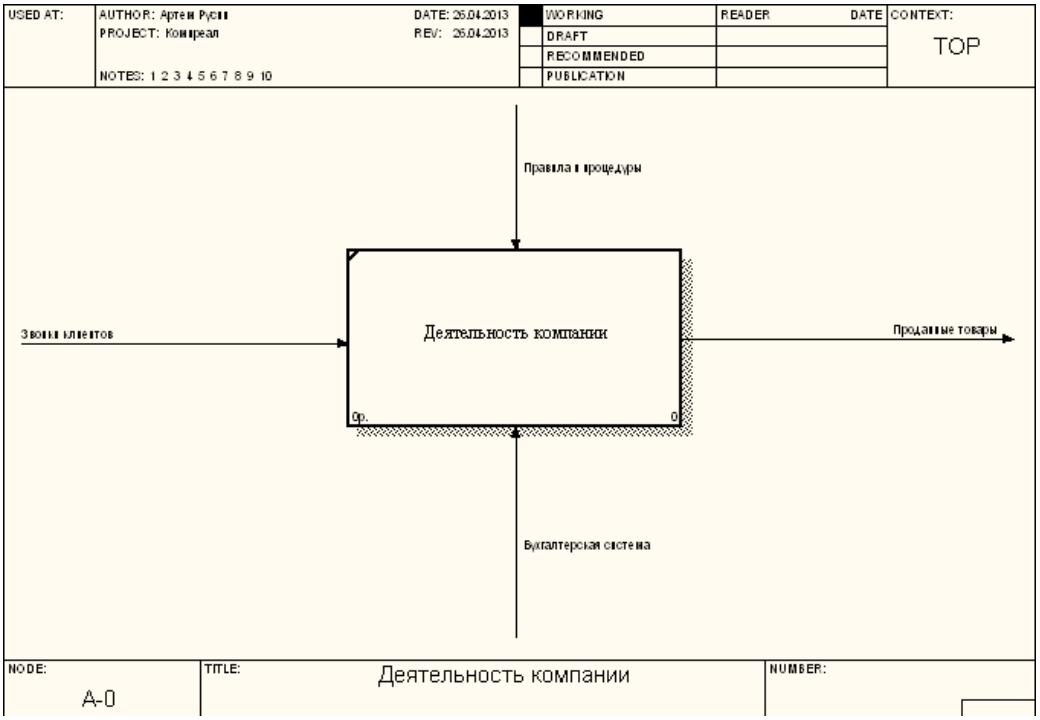



Рисунок 1. Контекстная диаграмма АО «Компреал»

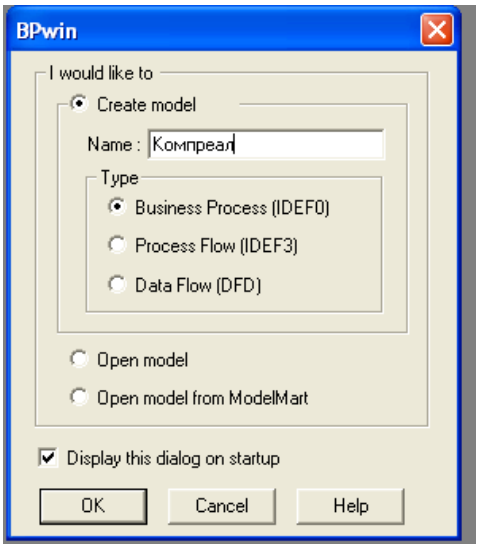



Рисунок 1.1 – Диалоговое окно

1. Контекстная диаграмма будет создана автоматически.
2. В меню Edit выбрать вкладку Model Properties (рисунок 1.2). На появившемся планшете во вкладке General внести имя модели «Компреал», имя проекта «Модель деятельности компании», имя автора и тип модели – Time Frame: AS-IS.

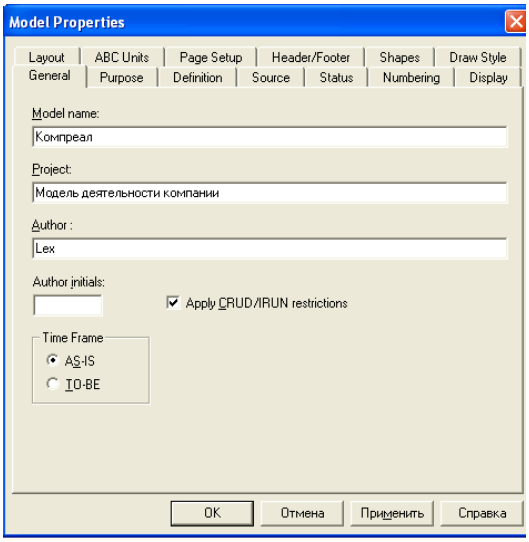



Рисунок 1.2 – Окно диалога Model Properties Editor

3. Во вкладке Purpose внести цель – «Моделирование текущих бизнес-про-цессов (AS-IS) компании» и точку зрения «Viewpoint: директор».
4. Во вкладке Definition внести определение создаваемой модели: «Это модель, описывающая деятельность компании» и цель «Scope: общее управление бизнесом компании (исследование рынка, закупка материалов, сборка, тестирование, продажа готовых компьютеров)».
6. Перейти на контекстную диаграмму и правой кнопкой мыши щелкнуть по работе. В появившемся контекстном меню выбрать Name Editor…. Во вкладке внести имя «Деятельность компании».
7. Заполнить вкладку Definition определением вида работ: «текущие бизнес-процессы компании» (рисунок 1.3).

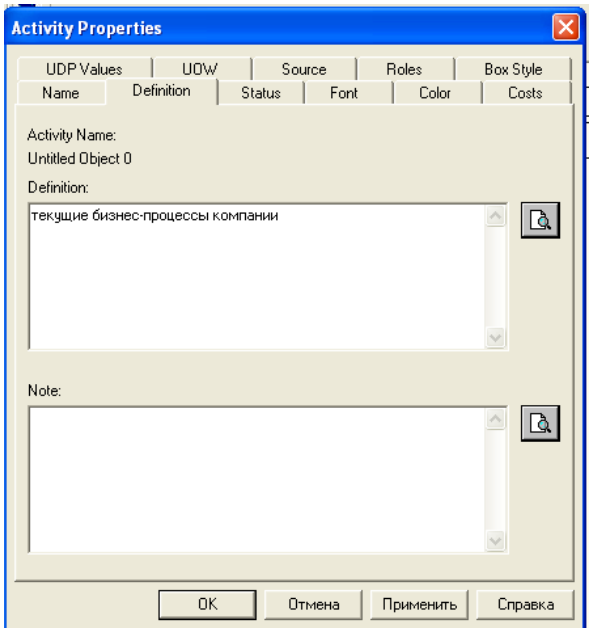



Рисунок 1.3 – Окно диалога IDEF0 Activity Properties

## Лабораторная работа 2. Создание диаграммы декомпозиции.

Создание диаграммы декомпозиции A-0 (TOP-диаграммы) «Деятельность компании» в диаграмму A0.

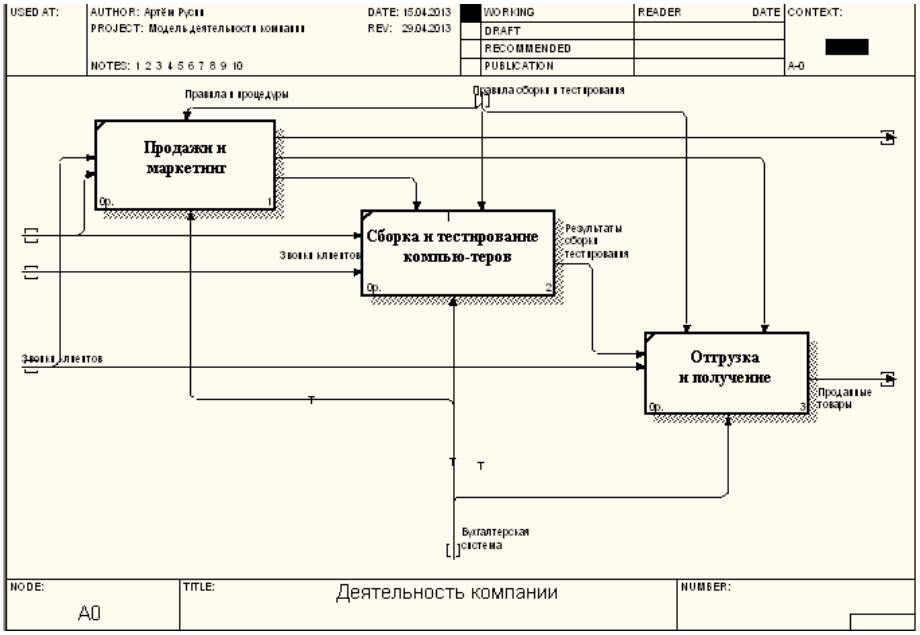



Рисунок 3. Декомпозиции A-0

1. Выбрать кнопку перехода на нижний уровень на панели инструментов и в диалоге **Active Box Count** (рисунок 2.1) установить число функций (работ) на диаграмме нижнего уровня – 3.

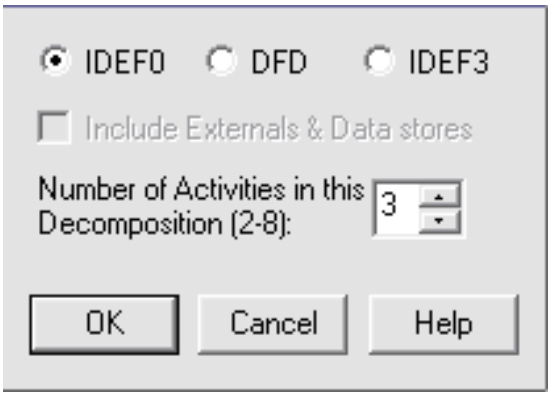



Рисунок 2.1 – Окно диалога **Active Box Count**

2.На автоматически созданной диаграмме декомпозиции через контекстное меню назвать каждую из трех функций (работ) посредством выбора вкладки **Name**, затем внести определение для каждой работы согласно табл. 2.1.

3. Повторите операцию для всех трех работ. Затем внесите определение, статус и источник для каждой работы согласно табл. 4.2.1.

| *Activity Name* | *Definition* |
|---|---|
| Продажи и маркетинг | Телемаркетинг и презентации, выставки |
| Сборка и тестирование компьютеров | Сборка и тестирование настольных и портативных компьютеров |
| Отгрузка и получение | Отгрузка заказов клиентам и получение компонентов от поставщиков |

Таблица 4.2.1. Работы диаграммы декомпозиции AO

*4.* Для изменения свойств работ после их внесения в диаграмму можно воспользоваться словарем работ. Вызов словаря - меню Dictionary /Activity (рис. 4.2.2).

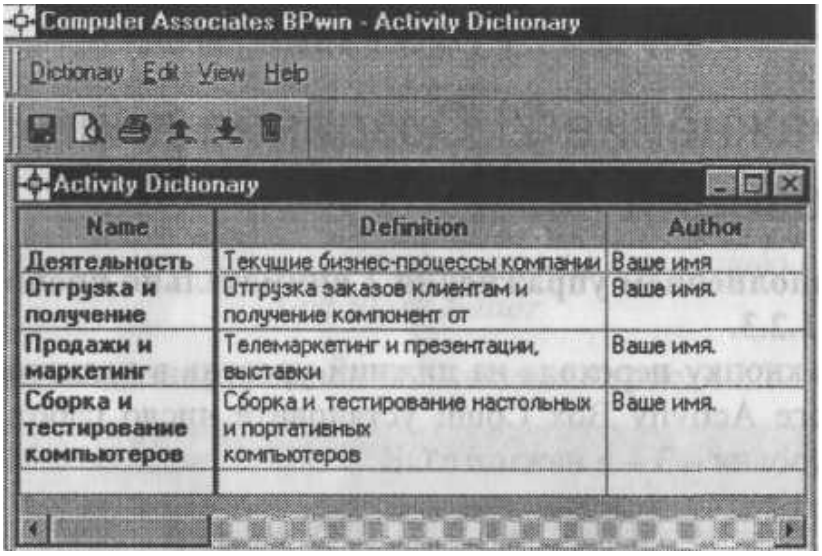



Рис. 4.2.2. Словарь Activity Dictionary

5.Если описать имя и свойства работы в словаре, ее можно будет внести в диаграмму позже с помощью кнопки  в палитре инструментов.

4. Невозможно удалить работу из словаря, если она используется на какой-либо диаграмме. Если работа удаляется из диаграммы, из словаря она не удаляется. Имя и описание такой работы может быть использовано в дальнейшем. Для добавления работы в словарь необходимо перейти в конец списка и щелкнуть правой кнопкой по последней строке. Возникает новая строка, в которой нужно внести имя и свойства работы.

5. (Purge). 3. Перейдите в режим рисования стрелок. Свяжите граничные стрелки  (кнопка L=,rt на палитре инструментов так, как показано на рис. 4.2.3.

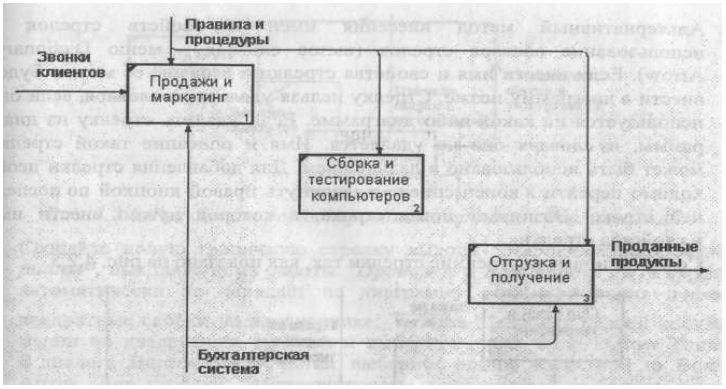



Рис. 4.2.3. Связанные граничные стрелки на диаграмме АО

Правой кнопкой мыши щелкните по ветви стрелки управления работы *"Сборка и тестирование компьютеров"* и переименуйте ее в *"Правила сборки и тестирования"* (рис. 4.2.4).

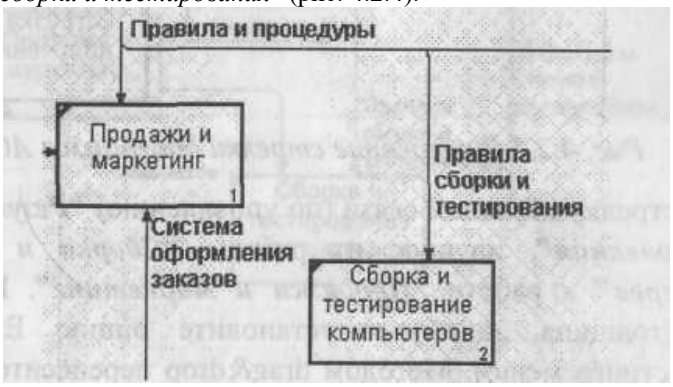



Рис. 4.2.4. Стрелка «Правила сборки и тестирования»

6. Внесите определение для новой ветви: "Инструкции по сборке, процедуры тестирования, критерии производительности и т. д." Правой кнопкой мыши щелкните по ветви стрелки механизма работы *"Продажи и маркетинг"* и переименуйте ее в *"Систему оформления заказов "*.

7. Альтернативный метод внесения имен и свойств стрелок -использование словаря стрелок (вызов словаря - меню Dictionary/Arrow). Если внести имя и свойства стрелки в словарь, ее можно будет внести в диаграмму позже. Стрелку нельзя удалить из словаря, если она используется на какой-либо диаграмме. Если удалить стрелку из диаграммы, из словаря она не удаляется. Имя и описание такой стрелки может быть использовано в дальнейшем. Для добавления стрелки необходимо перейти в конец списка и щелкнуть правой кнопкой по последней строке. Возникает новая строка, в которой нужно внести имя и свойства стрелки.

8. Создайте новые внутренние стрелки так, как показано на рис. 4.2.5.

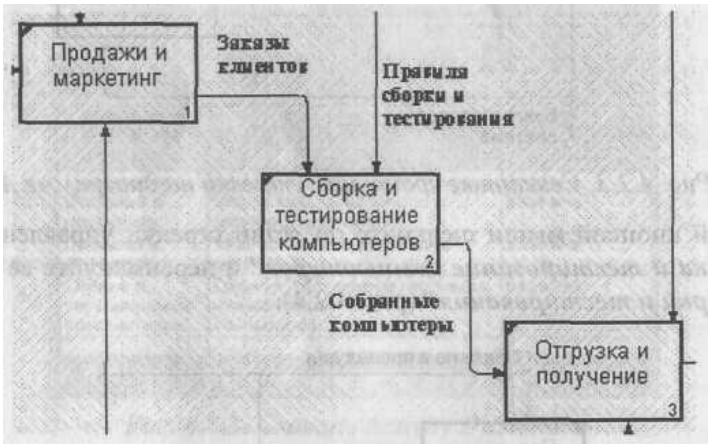



Рис. 4.2.5.Внутренние стрелки диаграммы АО

10. Создайте стрелку обратной связи (по управлению) ***"Результаты сборки и тестирования",*** идущую от работы ***"Сборка и тестирование компьютеров"*** к работе ***"Продажи и маркетинг".*** Измените стиль стрелки (толщина линий) и установите опцию Extra Arrowhead (из контекстного меню). Методом drag&drop перенесите имена стрелок так, чтобы их было удобнее читать. Если необходимо, установите Squiggle (из контекстного меню). Результат изменений показан на рис. 4.2.6.


Рис. 4.2.6. Результат редактирования стрелок на диаграмме АО 8.

11.Создайте новую граничную стрелку выхода ***"Маркетинговые материалы*** , выходящую из работы ***"Продажи и маркетинг".*** Эта стрелка автоматически не попадает на диаграмму верхнего уровня **и** имеет квадратные скобки на наконечнике:--> . Щелкните правой кнопкой мыши по квадратным скобкам и выберите пункт меню Arrow Tunnel В диалоге Border Arrow Editor выберите опцию Resolve it to Border Arrow. Для стрелки ***"Маркетинговые материалы"*** выберите опцию Тпш из контекстного меню. Результат выполнения упражнения 2 показан на рис. 4.2.7.

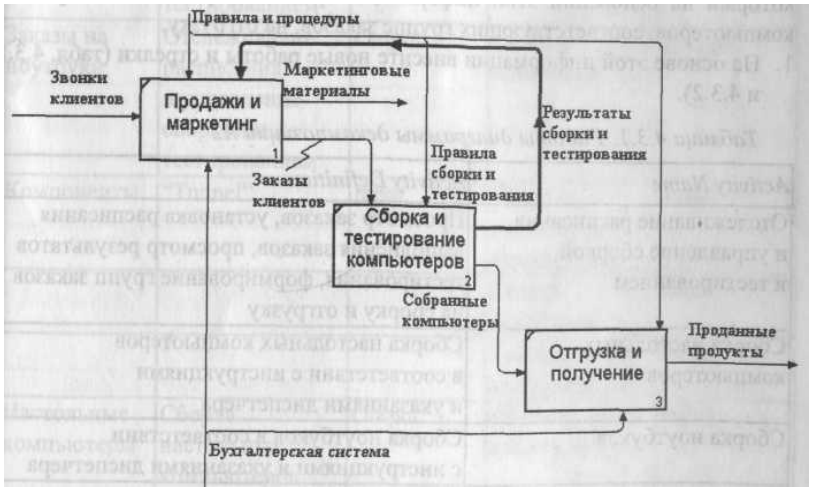



Рис. 4.2.7. Результат выполнения упражнения 2 - диаграмма АО

Лабораторная работа 3. Создание диаграммы декомпозиции А2.

Декомпозируем работу *«Сборка и тестирование компьютеров».*
В результате проведения экспертизы получена следующая информация.
Производственный отдел получает заказы клиентов от отдела продаж по мере их поступления.
Диспетчер координирует работу сборщиков, сортирует заказы, группирует их и дает указание на отгрузку компьютеров, когда они готовы.

Каждые 2 часа диспетчер группирует заказы - отдельно для настольных компьютеров и ноутбуков - и направляет на участок сборки.
Сотрудники участка сборки собирают компьютеры согласно спецификациям заказа и инструкциям по сборке. Когда группа компьютеров, соответствующая группе заказов, собрана, она направляется на тестирование. Тестировщики тестируют каждый компьютер и в случае необходимости заменяют неисправные компоненты.
Тестировщики направляют результаты тестирования диспетчеру, который на основании этой информации принимает решение о передаче компьютеров, соответствующих группе заказов, на отгрузку. 1. На основе этой информации внесите новые работы и стрелки (табл. 4.3.1 и 4.3.2).

Таблица 4.3.1. Работы диаграммы декомпозиции A2

| *Activity Name* | *Activity Definition* |
|---|---|
| Отслеживание расписания и управление сборкой и тестированием | Просмотр заказов, установка расписания выполнения заказов, просмотр результатов тестирования, формирование групп заказов на сборку и отгрузку |
| Сборка настольных компьютеров | Сборка настольных компьютеров в соответствии с инструкциями и указаниями диспетчера |
| Сборка ноутбуков | Сборка ноутбуков в соответствии с инструкциями и указаниями диспетчера |
| Тестирование компьютеров | Тестирование компьютеров и компонентов. Замена неработающих компонентов |

| *Arrow Name* | *Arrow Source* | *Arrow Source Type* | *Arrow Dest.* | *Arrow Dest. Type* |
|---|---|---|---|---|
| Диспетчер | Персонал производственного отдела | | Отслеживание расписания и управление сборкой и тестированием | Mechanism |
| Заказы клиентов | Граница диаграммы | Control | Отслеживание расписания и управление сборкой и тестированием | Control |
| Заказы на настольные компьютеры | Отслеживание расписания и управление сборкой и тестированием | Output | Сборка настольных компьютеров | Control |
| Заказы на ноутбуки | Отслеживание расписания и управление сборкой и тестированием | Output | Сборка ноутбуков | Control |
| Компоненты | "Tunnel" | Input | Сборка настольных компьютеров | Input |
| | | | Сборка ноутбуков | Input |
| | | | Тестирование компьютеров | Input |
| Настольные компьютеры | Сборка настольных компьютеров | Output | Тестирование компьютеров | Input |
| Ноутбуки | Сборка ноутбуков | Output | Тестирование компьютеров | Input |
| | "Tunnel" | Mechanism | Сборка | Mechanism |

Таблица 4.3.2. Стрелки диаграммы декомпозиции A2

| *Arrow Name* | *Arrow Source* | *Arrow Source Type* | *Arrow Dest.* | *Arrow Dest. Type* |
|---|---|---|---|---|
| Персонал производственного отдела | | | настольных компьютеров | |
| | | | Сборка ноутбуков | Mechanism |
| Правила сборки и тестирования | Граница диаграммы | | Сборка настольных компьютеров | Control |
| | | | Сборка ноутбуков | Control |
| | | | Тестирование компьютеров | Control |
| Результаты сборки и тестирования | Сборка настольных компьютеров | Output | Граница диаграммы | Output |
| | Сборка ноутбуков | Output | | |
| | Тестирование компьютеров | Output | | |
| Результаты тестирования | Тестирование компьютеров | Output | Отслеживание расписания и управление сборкой и тестированием | Input |
| Собранные компьютеры | Тестирование компьютеров | Output | Граница диаграммы | Output |
| Тестировщик | Персонал производственного отдела | | Тестирование компьютеров | Mechanism |
| Указание передать компьютеры на отгрузку | Отслеживание расписания и управление сборкой и тестированием | Output | Тестирование компьютеров | Control |

2. Туннелируйте и свяжите на верхнем уровне граничные стрелки, если это необходимо. Результат выполнения упражнения 3 показан на рис. 4.3.1.

Рис. 4.3.1. Результат выполнения

Лабораторная работа 4. Создание диаграммы узлов. Создание FEO. диаграммы. Расщепление и слияние моделей.

Создание диаграммы узлов.

Выберите меню Diagram/Add Node Tree. В первом диалоге гида Node Tree Wizard внесите имя диаграммы, укажите диаграмму корня дерева и количество уровней (рис. 4.4.1).

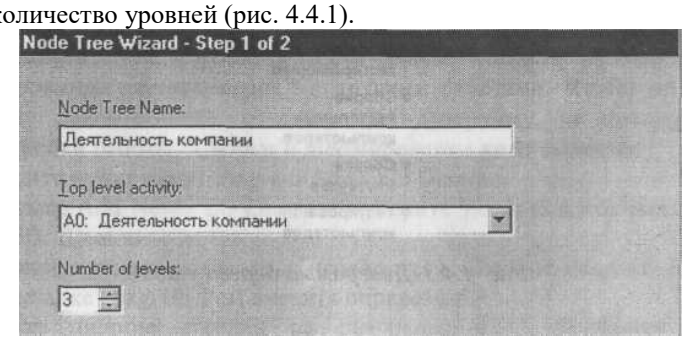


Рис. 4.4.1. Первый диалог гида Node Tree Wizard

2. Во втором диалоге установите опции, как на рис. 4.4.2.

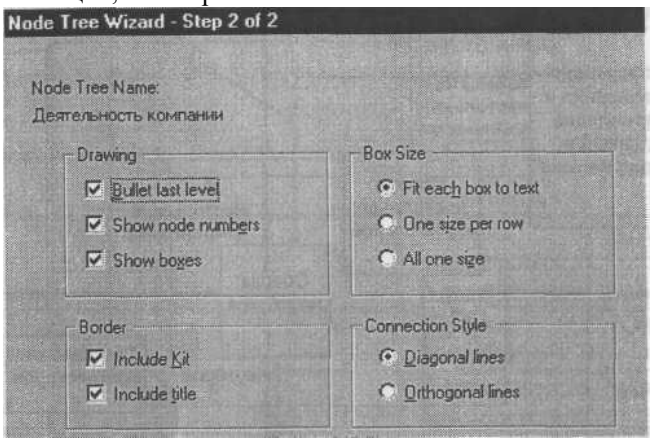


Рис. 4.4.2. Второй диалог гида Node Tree Wizard

Щелкните по Finish. Создается диаграмма дерева узлов. Результат можно посмотреть на рис. 4.4.3.

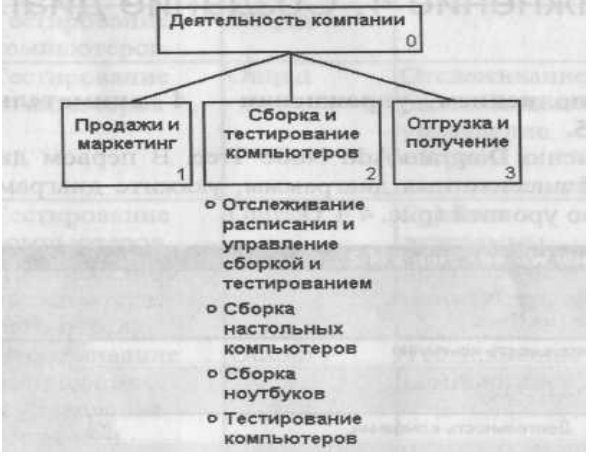


Рис. 4.4.3. Диаграмма дерева узлов

Диаграмму дерева узлов можно модифицировать. Нижний уровень может быть отображен не в виде списка, а в виде прямоугольников, так же как и верхние уровни.
Для модификации диаграммы правой кнопкой мыши щелкните по свободному месту, не занятому объектами, выберите меню Node tree Diagram Properties и во вкладке Style диалога Node Tree Properties отключите опцию Bullet Last Level. Щелкните по ОК. Результат показан на рис. 4.4.4.


Рис. 4.4.4. Результат выполнения упражнения 4

Создание FEO. Диаграммы

Предположим, что при обсуждении бизнес-процессов возникла необходимость детально рассмотреть взаимодействие работы *"Сборка и тестирование компьютеров"* с другими работами. Чтобы не портить диаграмму декомпозиции, создайте FEO-диаграмму, на которой будут только стрелки работы *"Сборка и тестирование компьютеров "*.
Выберите пункт меню Diagram/Add FEO Diagram.
В диалоге Add New FEO Diagram выберите тип и внесите имя диаграммы FEO. Щелкните по ОК.
Для определения диаграммы перейдите в Diagram/Diagram Properties и во вкладке Diagram Text внесите определение.
Удалите лишние стрелки на диаграмме FEO. Результат показан на рис. 4.5.1.

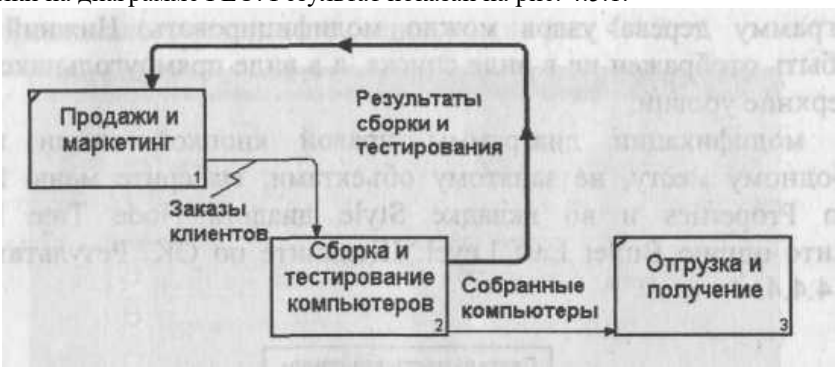



Рис. 4.5.1. Диаграмма FEO

Для перехода между стандартной диаграммой, деревом узлов **и** FEO используйте кнопку

на палитре инструментов.

***Расщепление и слияние моделей***

**Расщепление модели**

Перейдите на диаграмму AO. Правой кнопкой мыши щелкните по работе ***"Сборка и тестирование компьютеров"*** и выберите Split model.

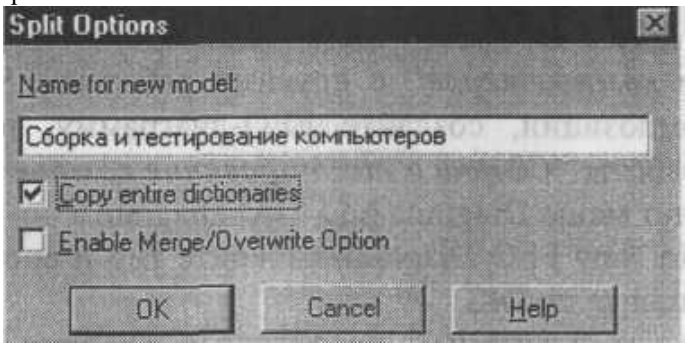



Рис. 4.6.1. Диалог Split Option

**В** диалоге Split Option внесите имя новой модели "Сборка и тестирование компьютеров", установите опции, как на рисунке, **и** щелкните по ОК (рис. 4.6.1).
Посмотрите на результат: в Model Explorer появилась новая модель, а на диаграмме АО модели "Деятельность компании" появилась стрелка вызова ***"Сборка и тестирование компьютеров".***
Создайте в модели "Сборка и тестирование компьютеров" новую стрелку ***"Неисправные компоненты".*** На диаграмме АО это будет граничная стрелка выхода, на диаграмме АО - граничная стрелка выхода от работ ***"Сборка настольных компьютеров", "Тестирование компьютеров"*** и ***"Сборка ноутбуков".***
2.Внесите свойства новой модели:
Time Frame: AS-IS;
Purpose: Документировать работу ***"Отгрузка и получение";***
Viewpoint: Начальник отдела;
Definition: Модель создается для иллюстрации возможностей BPwin по расщеплению и слиянию моделей
Scope: Работы по получению комплектующих и отправке готовой продукции.
3.Декомпозируйте контекстную работу на 3 работы (табл. 4.11.1).

Таблица 4.11.1. Декомпозиция работы "Отгрузка и получение"

| *Activity Name* | *Activity Definition* |
|---|---|
| Получить комплектующие | Физически получить комплектующие и сделать соответствующие записи в информационной системе |
| Доставить комплектующие | Доставить комплектующие сборщикам и тестировщикам |
| Отгрузить товар и возврат | Отгрузить товар клиентам и неисправные компоненты (возврат) поставщикам |

Свяжите граничные стрелки, как показано на рис. 4.11.1.

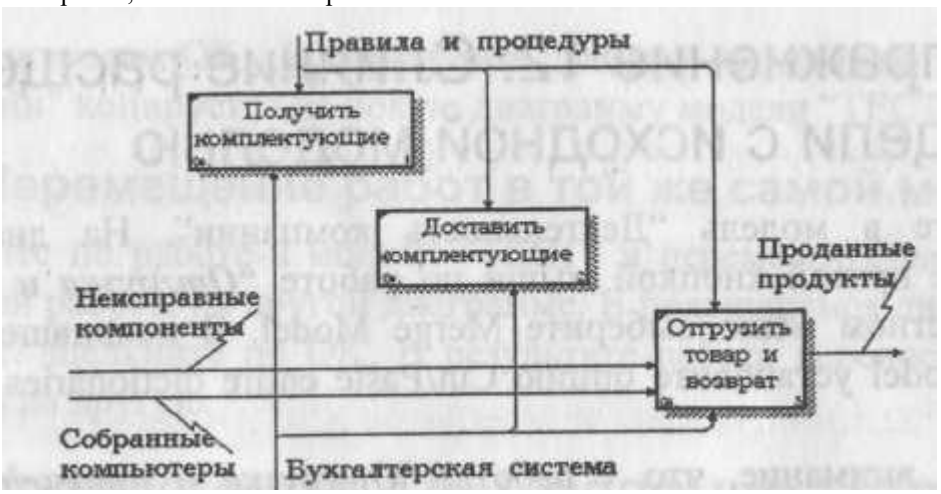



Рис. 4.11.1. Внутренние стрелки на декомпозиции работы "Отгрузка и получение"

5. Внесите следующие внутренние и граничные стрелки (табл. **4.11**.2).

Таблица 4.11.2. Внутренние и граничные стрелки на декомпозиции работы "Отгрузка и получение"

| *Arrow Name* | *Arrow Definition* |
|---|---|
| Возврат поставщику | Неисправные компоненты |
| Компоненты | *Выберите название из списка (словаря)* |
| Компоненты от поставщика | |
| Проверенные компоненты | Проверенные и подготовленные для передачи сборщикам и тестировщикам компоненты |

6. Туннелируйте граничные стрелки (Resolve Border Arrow). Результат выполнения упражнения показан на рис. 4.11.2.

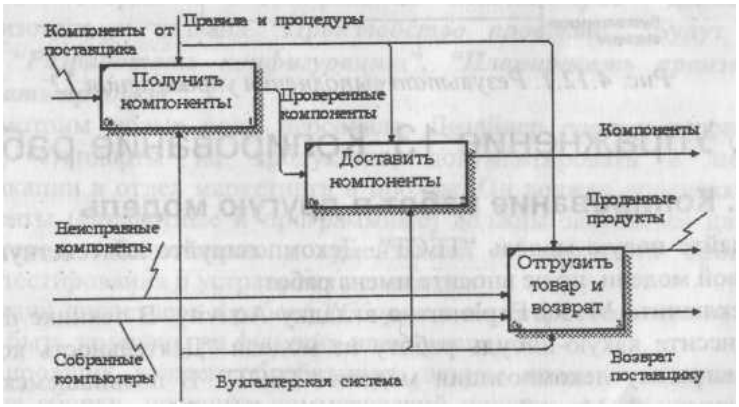



Рис. 4.11.2. Результат выполнения упражнения 11

**Слияние модели**

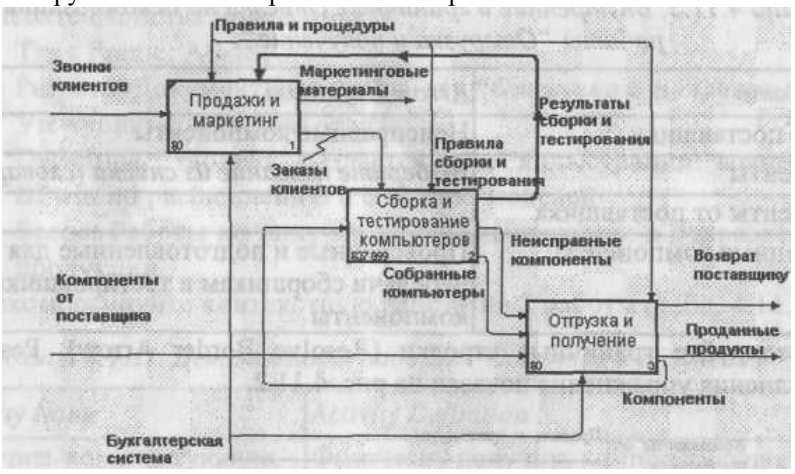
1.Перейдите в модель "Деятельность компании". На диаграмме АО щелкните правой кнопкой мыши по работе ***"Отгрузка и получение".***

**В** контекстном меню выберите Merge Model. В появившемся диалоге Merge Model установите опцию Cut/Paste entire dictionaries и щелкните **по** ОК.

Обратите внимание, что у работы ***"Отгрузка и получение"*** исчезла стрелка вызова и появилась новая декомпозиция.

Появились новые стрелки с квадратными скобками. Туннелируйте эти стрелки (Resolve Border Arrow).

2.На диаграмме АО туннелируйте и свяжите стрелки согласно рис. 4.12.1.




Лабораторная работа 5. Создание диаграммы IDEF3. Создание сценария. Создание отчетов.

Создание диаграммы IDEF3

**1.** Перейдите на диаграмму А2 и декомпозируйте работу "Сборка настольных компьютеров". В диалоге Activity Box Count (рис. 4.7.1) установите число работ 4 и нотацию IDEF3.

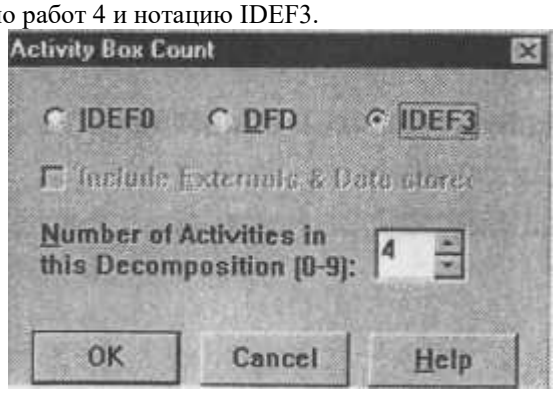



Рис. 4.7.1. Выбор нотации IDEF3 в диалоге Activity Box Count

Возникает диаграмма IDEF3, содержащая работы (UOW). Правой кнопкой мыши щелкните по работе, выберите в контекстном меню Name и внесите имя работы ***"Подготовка компонентов".*** Затем во вкладке Definition внесите определение "Подготавливаются все компоненты компьютера согласно спецификации заказа". 2. Во вкладке UOW внесите свойства работы (табл. 4.7.1).

Таблица 4.7.1. Свойства UOW

| | |
|---|---|
| *Objects* | Компоненты: винчестеры, корпуса, материнские платы, видеокарты, звуковые карты, дисководы CD-ROM и флоппи, модемы, программное обеспечение |
| *Facts* | Доступные операционные системы: Windows 98, Windows NT, Windows 2000 |
| *Constrains* | Установка модема требует установки дополнительного программного обеспечения |

3.Внесите в диаграмму еще 3 работы (кнопка I 

I).
Внесите имена работ:
***Установка материнской платы и винчестера;***
***Установка модема;***
***Установка дисковода CD-ROM;***
***Установка флоппи- дисковода;***
***Инсталляция операционной системы;***
***Инсталляция дополнительного программного обеспечения.***
4.С помощью кнопки  — _ палитры инструментов создайте объект ссылки. Внесите имя объекта внешней ссылки ***«Компоненты».***
Свяжите стрелкой объект ссылки и работу ***"Подготовка компонент".***
5.Свяжите стрелкой работы *"Подготовка компонентов"* (выход) и ***"Установка материнской платы и винчестера".*** Измените стиль стрелки на Object Flow.
В IDEF3 имя стрелки может отсутствовать, хотя BPwin показывает отсутствие имени как ошибку. Результат показан на рис. 4.7.2.


Рис. 4.7.2. Результат создания VOW и объекта ссылки

6. С помощью кнопки  «» на палитре инструментов внесите два перекрестка типа "асинхронное или" и свяжите работы с перекрестками, как показано на рис. 4.7.3.

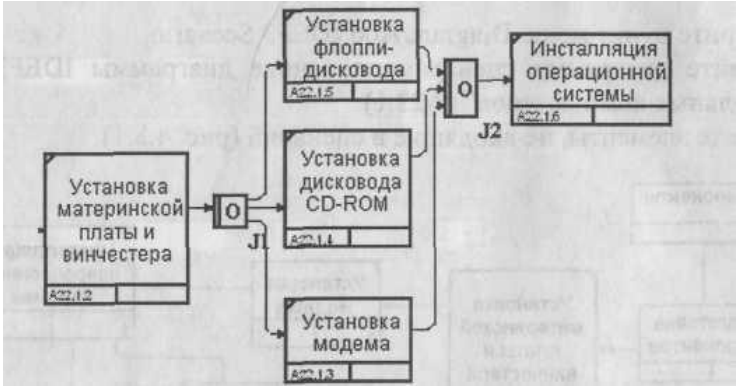



Рис. 4.7.3. Диаграмма IDEF3 после создания перекрестков

7. Правой кнопкой щелкните по перекрестку для разветвления (fan-out), выберите Name и внесите имя ***"Компоненты, требуемые в спецификации заказа".***
Создайте два перекрестка типа исключающего "ИЛИ" и свяжите работы, как показано на рис. 4.7.4.

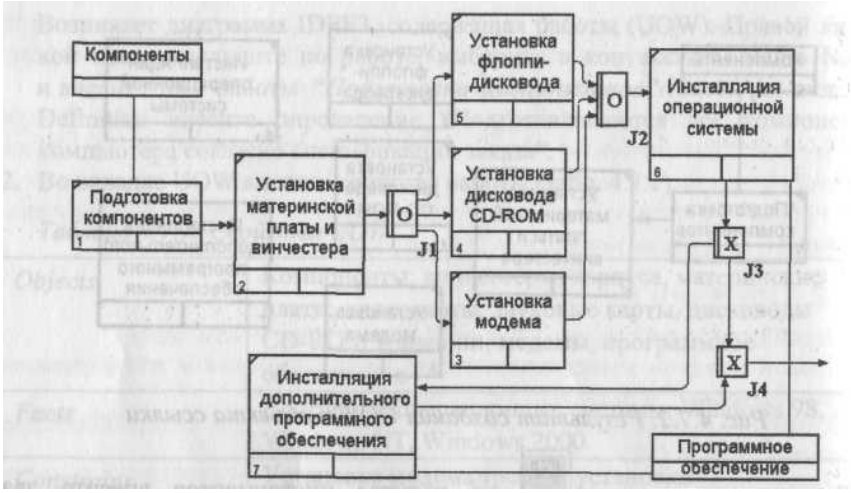



Рис. 4.7.4. Результат выполнения упражнения 7

Создание сценария

1.Выберите пункт меню Diagram/Add IDEF3 Scenario.

Создайте диаграмму сценария на основе диаграммы IDEF3 "Сборка настольных компьютеров" (A22.1).

2.Удалите элементы, не входящие в сценарий (рис. 4.8.1).


*Рис. 4.8.1. Результат выполнения*

Создание отчетов

***Создание отчетов в BPwin.***

**Встроенные шаблоны отчетов.**

Существует три способа создания отчетов в BPwin 4.0:

с помощью встроенных шаблонов;

с помощью Report Template Builder;

с помощью RPTwin.

Для создания отчетов по функциональной модели можно также использовать генераторы отчетов третьих фирм, например Crystal Reports.

Отчеты на основе встроенных шаблонов можно создать, выбрав из меню Tools/Reports необходимый тип шаблона. Всего имеется семь типов шаблонов отчетов:

Model Report. Этот отчет уже упоминался в 1.2.1. Он включает информацию о контексте модели - имя модели, точку зрения, область, цель, имя автора, дату создания и др.

Diagram Report. Отчет по конкретной диаграмме. Включает список объектов: работ, стрелок, хранилищ данных, внешних ссылок и т. д.

Diagram Object Report. Наиболее полный отчет по модели. Может включать полный список объектов модели: работ, стрелок с указанием их типа и др. - и свойства, определяемые пользователем.

Activity Cost Report. Отчет о результатах стоимостного анализа. Будет рассмотрен ниже.

Arrow Report. Отчет по стрелкам. Может содержать информацию из словаря стрелок, информацию о работе-источнике, работе-назначении стрелки и информацию о разветвлении и слиянии стрелок.

DataUsage Report. Отчет о результатах связывания модели процессов и модели данных. (Будет рассмотрен ниже.)

Model Consistency Report. Отчет, содержащий список синтаксических ошибок модели.

Синтаксические ошибки IDEF0 с точки зрения BPwin разделяются на три типа:

первых, это ошибки, которые BPwin выявить не в состоянии.

Например, синтаксис IDEF0 требует, чтобы имя работы было выражено отглагольным существительным *("Изготовление изделия", "Обслуживание клиента", "Выписка счета"* и т. д.), а имя стрелки также должно быть выражено существительным. BPwin не позволяет анализировать синтаксис естественного языка (английс кого и русского) и смысл имен объектов и поэтому игнорирует ошибки этого типа. Выявление таких ошибок - ручная работа, которая ложится на плечи аналитиков и должна контролироваться руководителем проекта.

Ошибки второго типа BPwin просто не допускает. Например, каждая грань работы предназначена для определенного типа стрелок. BPwin просто не позволит создать на диаграмме IDEF0 внутреннюю стрелку, выходящую из левой грани работы и входящую в правую грань.

Третий тип ошибок BPwin позволяет допустить, но детектирует их. Полный их список можно получить в отчете Model Consistency Report. Список ошибок может содержать, например, неименованные работы и стрелки (unnamed arrow, unnamed activity), несвязанные стрелки (unconnected border arrow), неразрешенные стрелки (unresolved (square tunneled) arrow connections) и т. д. Пример отчета Model Consistency Report .

При выборе пункта меню, который соответствует какому-либо отчету, появляется диалог настройки отчета. Для каждого из семи типов отчетов он выглядит по-своему. Рассмотрим типичный диалог Arrow Report .

Раскрывающийся список Standart Reports позволяет выбрать один из стандартных отчетов. Стандартный отчет - это запоминаемая комбинация переключателей, флажков и других элементов управления диалога. Для создания собственного стандартного отчета необходимо задать опции отчета, ввести имя отчета в поле списка выбора и щелкнуть по кнопке New. BPwin сохраняет информацию о стандартном отчете в файле BPWINRPT.INI. Все определения этого файла доступны из любой модели. Единственное ограничение - свойства, определяемые пользователем (User-Defined Properties). Они сохраняются в виде указателя и поэтому доступны только из "родной" модели. Стандартный отчет можно изменить (кнопка Update) или удалить (кнопка Delete).

В правом верхнем углу диалога находится группа управляющих элементов для выбора формата отчета. Доступны следующие форматы:

Labeled - отчеты включают метку поля, затем, в следующей строке, печатается содержимое поля;

Fixed Column - каждое поле печатается в собственной колонке;

Tab-Comma Delimited - каждое поле печатается в собственной колонке. Колонки разделяются знаком табуляции или запятыми;

DDE Table - данные передаются по протоколу DDE в приложение, например в MS Word или Excel;

RPTwin - отчет создается в формате RPTwin - специализированного генератора отчетов, который входит в поставку BPwin.

Опция Ordering (на отчете по стрелкам отсутствует) сортирует данные по какому-либо значению.

Опция Multi-Valued Format регулирует вывод полей в отчете при группировке данных:

Repeating Group - детальные данные объединяются в одно поле, между значениями вставляется +.

Filled — дублирование данных для каждого заголовка группы;

Header (опция по умолчанию) - печатается заголовок группы, затем - детальная информация.

**Создание отчетов с помощью Report Template Builder**

Собственный шаблон отчета можно создать с помощью диалога Report Template Builder. Пункт меню Tools/Reports Builder вызывает диалог Report Templates (рис. 2.1.3). Кнопка New служит для создания нового шаблона, кнопка Edit - для редактирования существующего. Список выбора Output Type позволяет задать формат результата выполнения отчета. Отчет может быть экспортирован в текстовый формат, RTF и HTML. Кнопка Run позволяет выполнить отчет.

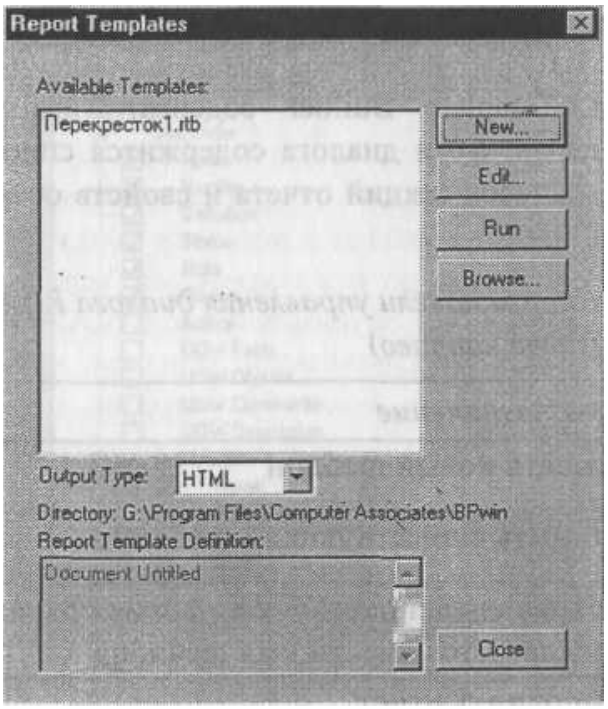



*Рис. 2.1.3. Диалог Report Templates*

Щелчок по кнопке New или Edit вызывает диалог диалога Report Template Builder (рис. 2.1.4).

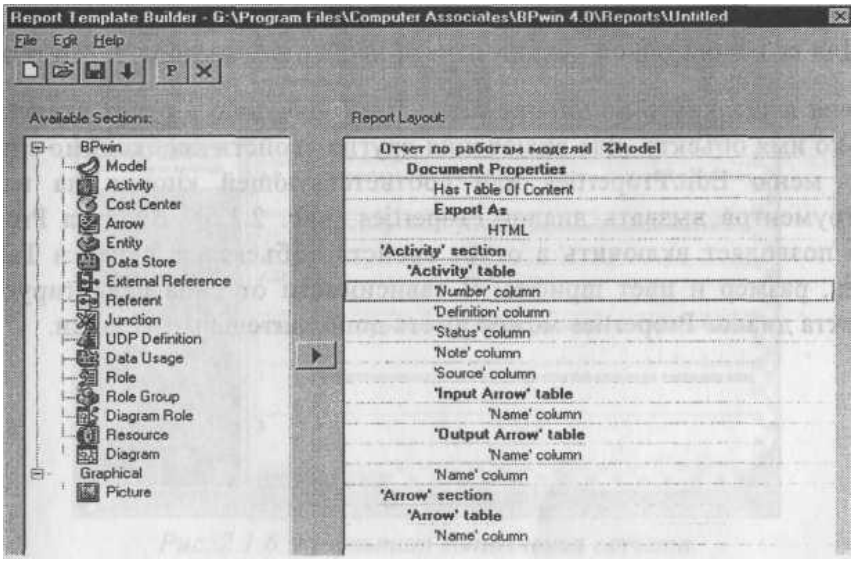



*Рис. 2.1.4. Диалог Report Template Builder*

Смысл кнопок панели управления диалога Report Template Builder приведен в табл. 2.1.1.

Диалог Report Template Builder содержит два списка и панель инструментов. В левой части диалога содержится список типов объектов модели, в правой - список секций отчета и свойств объектов, включенных в отчет.

*Таблица 2.1.1. Кнопки панели управления диалога Report Template Builder (слева направо)*

| *Кнопка* | *Предназначение* |
|---|---|
|  | Создать новый шаблон |
|  | Открыть существующий шаблон |
|  | Сохранить шаблон. Если шаблон сохраняется впервые, предлагается внести имя шаблона |
|  | Выполнить отчет |
| | Вызов диалога Properties |
|  | Удаление пункта отчета |
| | |
| | |
| | |

Для создания новой секции отчета необходимо выбрать тип объекта модели и щелкнуть по кнопке

По умолчанию в отчет включается только имя объекта. Для включения других свойств необходимо с помощью меню Edit/Properties или соответствующей кнопки на панели инструментов вызвать диалог Properties (рис. 2.1.5). Вкладка Property Tree позволяет включить в отчет свойство объекта, а вкладка Table -стиль, размер и цвет шрифта. В зависимости от типа редактируемого объекта диалог Properties может иметь дополнительные вкладки.

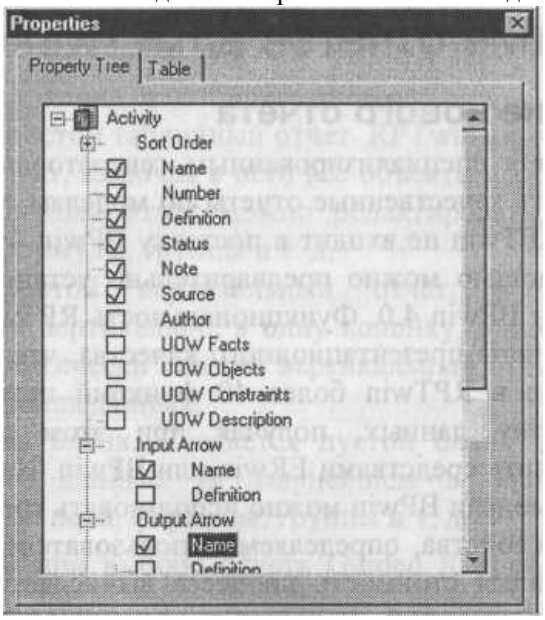



*Рис. 2.1.5. Диалог Properties*

В результате выполнения отчет экспортируется либо в текстовый файл формата RTF или в файл формата HTML (рис. 2.1.6).

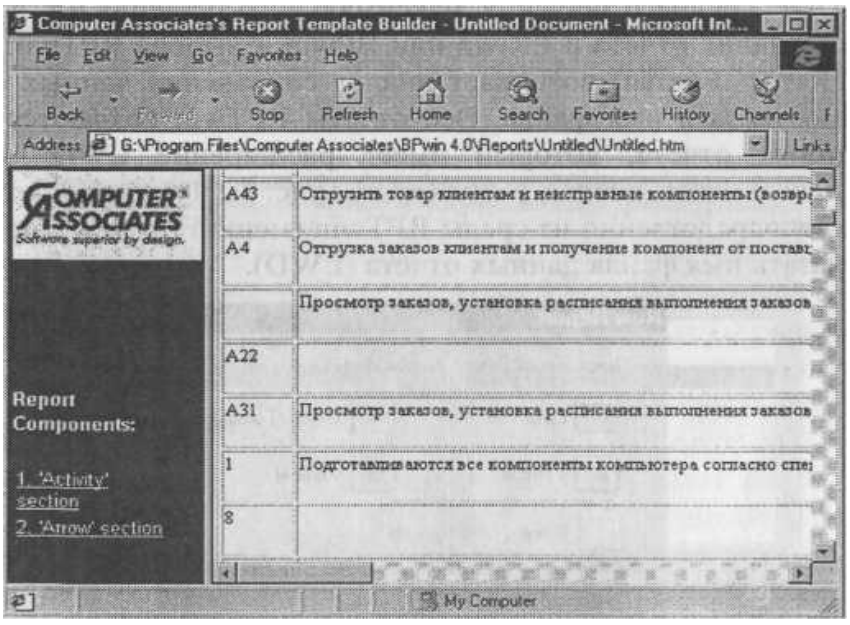



*Рис. 2.1.6. Результат выполнения отчета*

Лабораторная работа 6. Стоимостной анализ.

В диалоге Model Properties (вызывается из меню Mode/Model Properties) во вкладке ABC Units (рис. 4.9.1) установите единицы измерения денег и времени — рубли и часы.

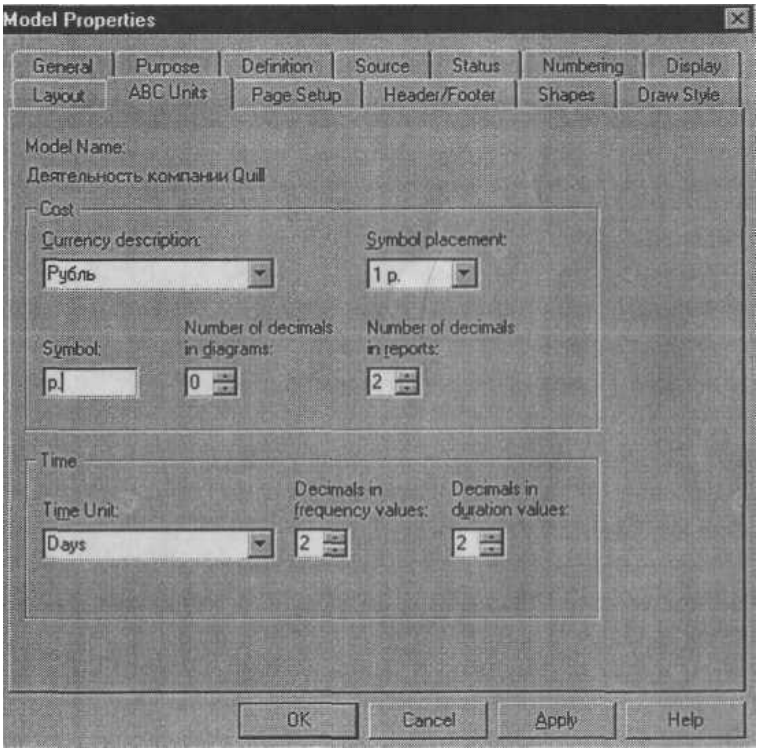



*Рис. 4.9.1. Вкладка ABC Units диалога Model Properties*

Перейдите в Dictionary/Cost Center и в диалоге Cost Center Dictionary внесите название и определение центров затрат (табл. 4.9.1).

*Таблица 4.9.1. Центры затрат ABC*

| *Центр затрат* | *Определение* |
|---|---|
| Управление | Затраты на управление, связанные с составлением графика работ, формированием партий компьютеров, контролем над сборкой и тестированием |
| Рабочая сила | Затраты на оплату рабочих, занятых сборкой и тестированием компьютеров |
| Компоненты | Затраты на закупку компонентов |

Для отображения стоимости каждой работы в нижнем левом углу прямоугольника перейдите в меню Model/Model Properties и во вкладке Display диалога Model Properties включите опцию ABC Data (рис. 4.9.2).

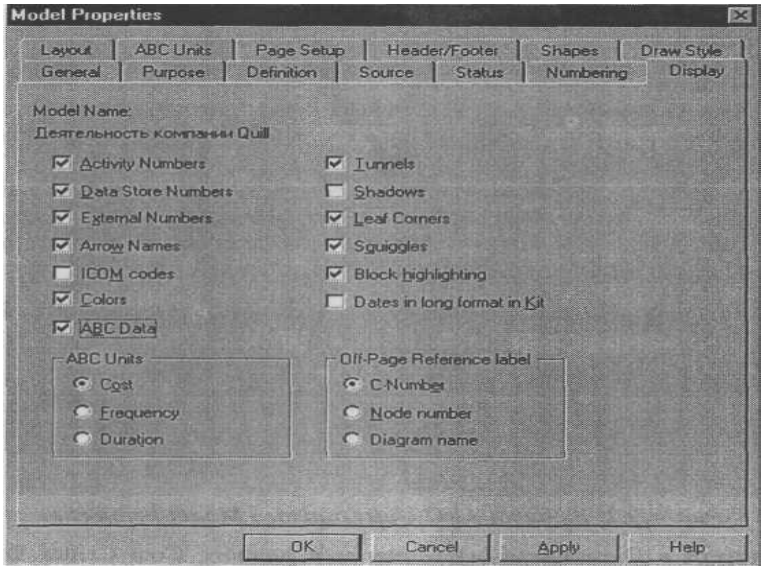



*Рис. 4.9.2. Вкладка Display диалога Model Properties*

Для отображения частоты или продолжительности работы переключите радиокнопки в группе ABC Units.

Для назначения стоимости работе следует щелкнуть по ней правой кнопкой мыши и выбрать в контекстном меню Cost (рис. 4.9.3).

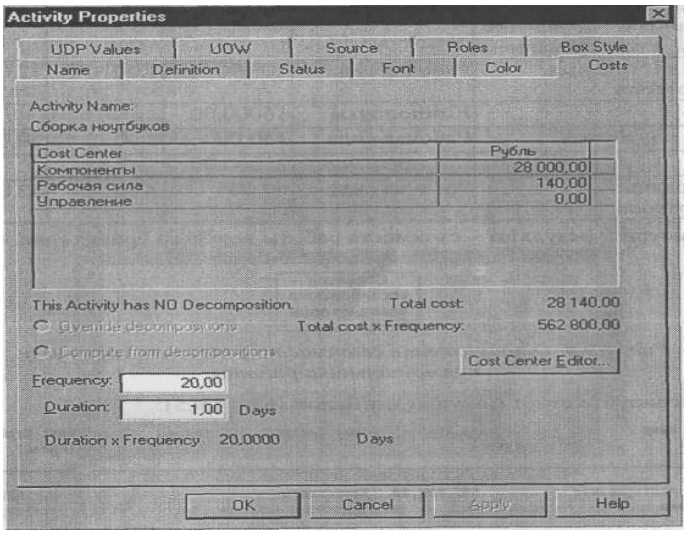



*Рис. 4.9.3. Вкладка Cost диалога Activity Properties 3.*

Для работ на диаграмме А2 внесите параметры ABC (табл. 4.9.2).

*Таблица 4.9.2. Стоимости работ на диаграмме А2*

| *Activity Name* | *Cost Center* | *Cost Center Cost, руб.* | *Duration, день* | *Frequency* |
|---|---|---|---|---|
| Отслеживание расписания и управление сборкой и тестированием | Управление | 500,00 | 1,00 | 1,00 |
| Сборка настольных компьютеров | Рабочая сила | 100,00 | 1,00 | 12,00 |
| | Компоненты | 16000,00 | | |
| Сборка ноутбуков | Рабочая сила | 140,00 | 1,00 | 20,00 |
| | Компоненты | 28000,00 | | |
| Тестирование компьютеров | Рабочая сила | 60,00 | 1,00 | 32,00 |

Посмотрите результат - стоимость работы верхнего уровня (рис. 4.9.4).

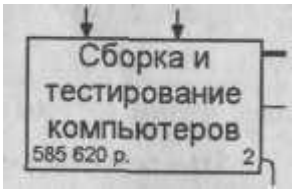



*Рис. 4.9.4. Отображение стоимости в нижнем левом углу прямоугольника работы*

4. Сгенерируйте отчет Activity Cost Report (рис. 4.9.5).

**Activity** Name ftctiuity Cost Cost CenterCost Center Cost

**<РУ.бль)(Рубль)**

**Сборка и тестирование 585 620,00**

**Компоненты579** 200,

80 компьютеров

Рабочая **сила5 920,08**

**Управление**500,00

**Отслеживание расписания 500,00**

**Управление500,08**

**Сборка настольных1 700,00**

**Компоненты1 600,**

**88 компьютеров**

**Рабочая сила100,08**

Лабораторная работа 7. Использование категорий UDP.

Перейдите в меню Dictionary/UDP Keywords и в диалоге UDP Keyword

List внесите ключевые слова UDP (рис. 4.10.1):

Расход ресурсов;

Документация;

Информационная система.

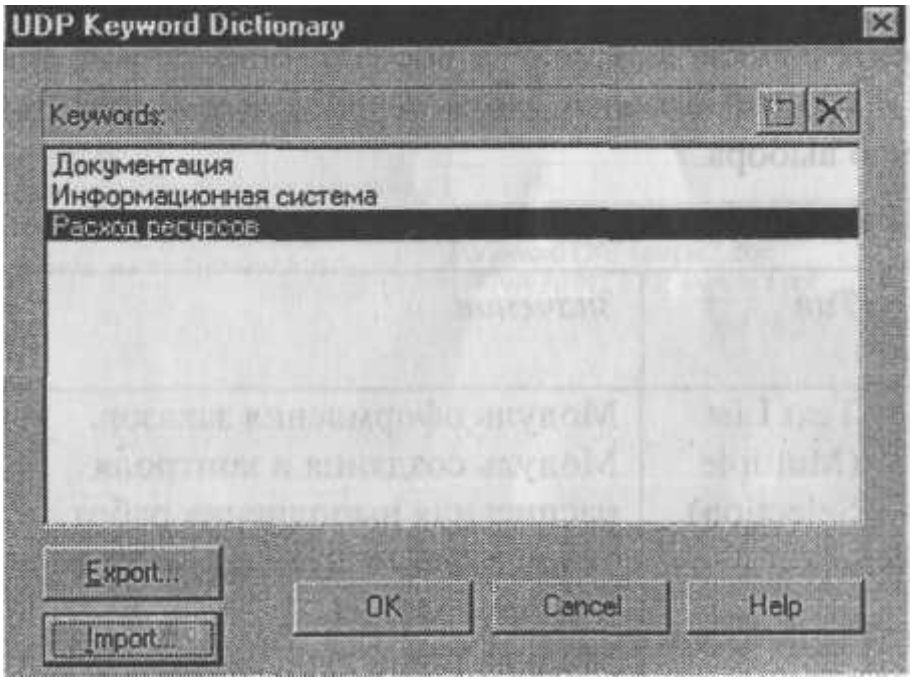



*Рис. 4.10.1. Словарь ключевых слов UDP*

Создайте UDP. Для этого перейдите в Dictionary/UDP и в словаре внесите имя UDP, например "Приложение".

Для UDP типа List необходимо в поле Value задать список значений. Для UDP - "Приложение". Внесите значение "Модуль оформления зака зов" (рис. 4.10.2).

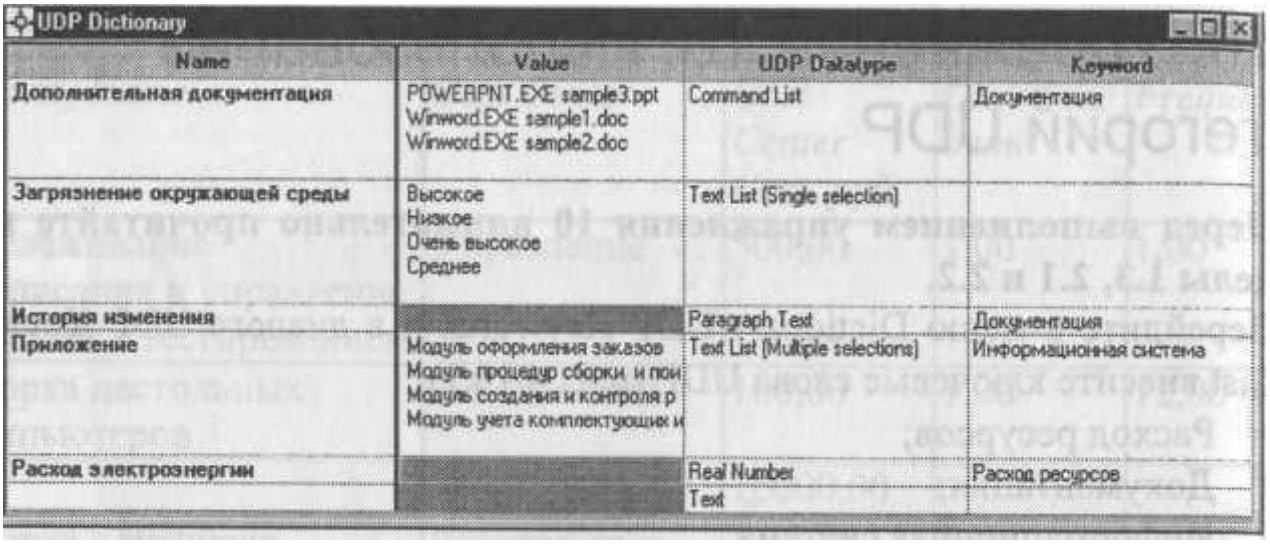



*Рис. 4.10.2. Словарь UDP*

Затем внесите другие значения в соответствии с табл. **4.10.1.** Для подключения к UDP ключевого слова перейдите к полю Keyword и щелкните по полю выбора.

*Таблица 4.10.1. Наименование и свойства UDP*

| *Наименование UDP* | *Тип* | *Значение* | *Ключевое слово* |
|---|---|---|---|
| Приложения | Text List (Multiple Selection) | Модуль оформления заказов.<br>Модуль создания и контроля расписания выполнения работ.<br>Модуль учета комплектующих и оборудования.<br>Модуль процедур сборки и поиска неисправностей | Информационная система |
| Дополнительная документация | Command List | Winword.EXE sample1.doc<br>Winword.EXE sample2.doc<br>POWERPNT.EXE sample3.ppt | Документация |
| История изменения | Paragraph Text | | Документация |
| Загрязнение окружающей среды | Text List (Single Selection) | Очень высокое<br>Высокое<br>Среднее<br>Низкое | |
| Расход электроэнергии | Real Number | | Расход ресурсов |

Для назначения UDP работе следует щелкнуть по ней правой кнопкой мыши и выбрать в контекстном меню UDP. Появляется вкладка UDP Values диалога Activity Properties (рис. 4.10.3).

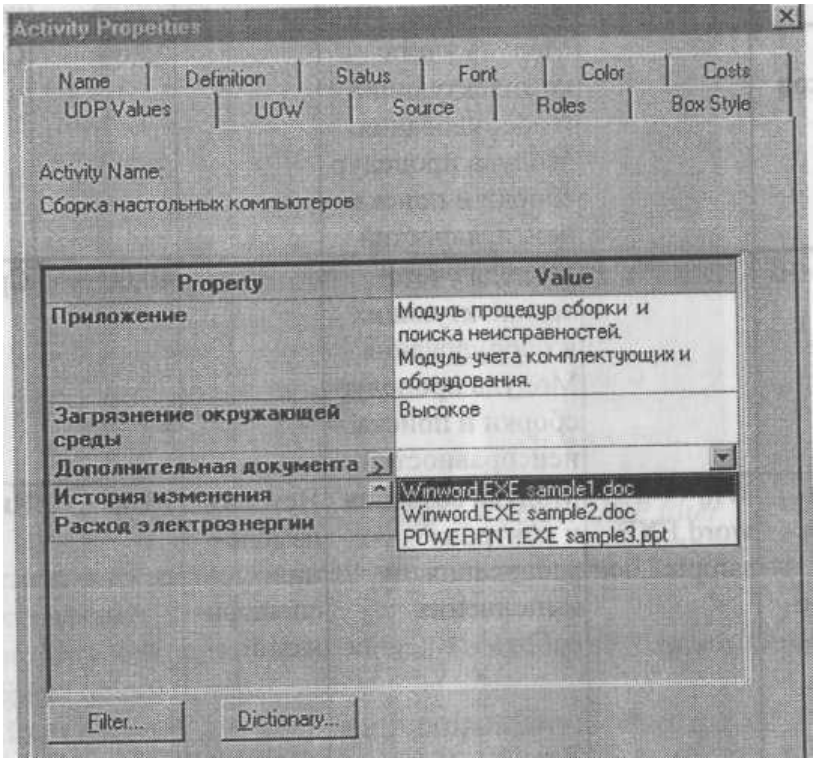



*Рис. 4.10.3. Вкладка UDP Values диалога Activity Properties*

Внесите значения UDP для работ (таблица 4.10.2).

*Таблица 4.10.2. Значения UDP*

| *Activity Name* | *Дополнительная документация* | *Приложения* | *История изменения* | *Расход электроэнергии* | *Загрязнение окружающей среды* |
|---|---|---|---|---|---|
| Сборка настольных компьютеров | | Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей | | 20,00 | Среднее |
| Сборка ноутбуков | | Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей | | 25,00 | Среднее |
| Тестирование компьютеров | | Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей | | 40,00 | Среднее |
| Отслеживание расписания и управление сборкой и тестированием | Winword.EXE sample2.doc | Модуль создания и контроля расписания выполнения работ | История изменения спецификаций | 10,00 | Низкое |

После внесения UDP типа Command или Command List щелчок по кнопке


приведет к запуску приложения.

В диалоге Activity Properties щелкните по кнопке Filter. В появившемся диалоге Diagram object UDP filter (рис. 4.10.4) отключите ключевые слова "Информационная система". Щелкните по OK. В результате в диалоге Activity Properties не будут отображаться UDP с ключевыми словами "Информационная система".

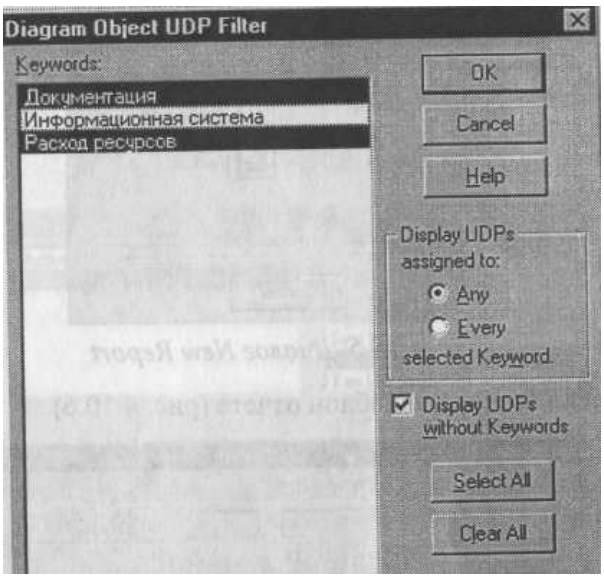



*Рис. 4.10.4. Диалог Diagram object UDP filter*

стрелкам.

7.Посмотрите отчет по UDP. Меню Tools/Report/Diagram Object Report.

Выберите опции отчета:

Start from Activity: A2. Сборка и тестирование компьютеров

Number of Levels: 2

User Defined Properties: Расход электроэнергии

Report Format: RPTwin.

8.Щелкните по кнопке Report. В появившемся диалоге "Сохранение файла" щелкните по кнопке "Сохранить".

Запускается генератор отчетов RPTwin и появляется диалог New Report. Выберите тип отчета Columnar (рис. 4.10.5).

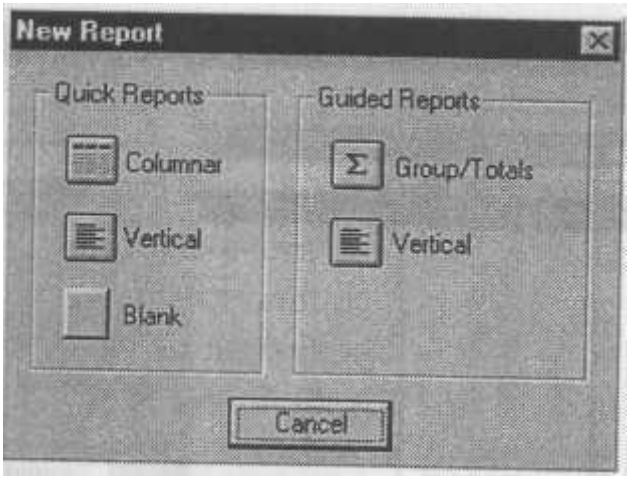


*Рис. 4.10.5. Диалог New Report* Автоматически создается шаблон отчета (рис. 4.10.6).

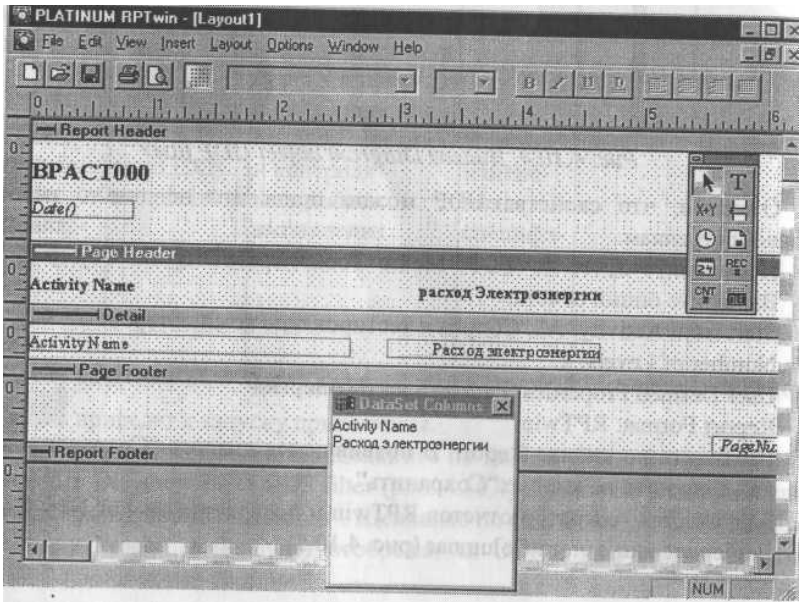



*Рис. 4.10.6. Шаблон отчета в RPTwin*

Нажатие на кнопку позволяет просмотреть отчет. Отразим в отчете суммарный расход электроэнергии.

9. Выберите в меню Insert/Formula Field, затем переместите маркер в секцию отчета Page Footer, затем щелкните один раз. Появляется диалог Formula Editor (рис. 4.10.7).

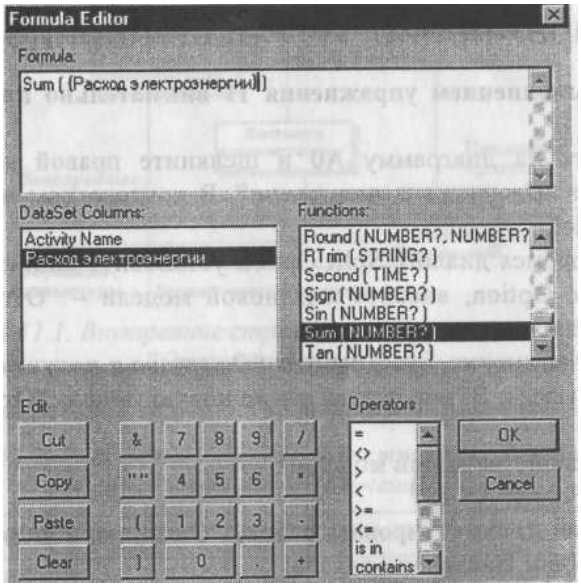



*Рис. 4.10.7. Диалог Formula Editor*

В поле Formula внесите текст формулы: Sum ({"Расход электроэнергии"})

Затем щелкните по ОК. Отчет показывается в окне просмотра (рис. 4.10.8). В нижней части страницы расположено суммирующее поле - результат вычисления формулы (на рис. 4.10.8 не видно).

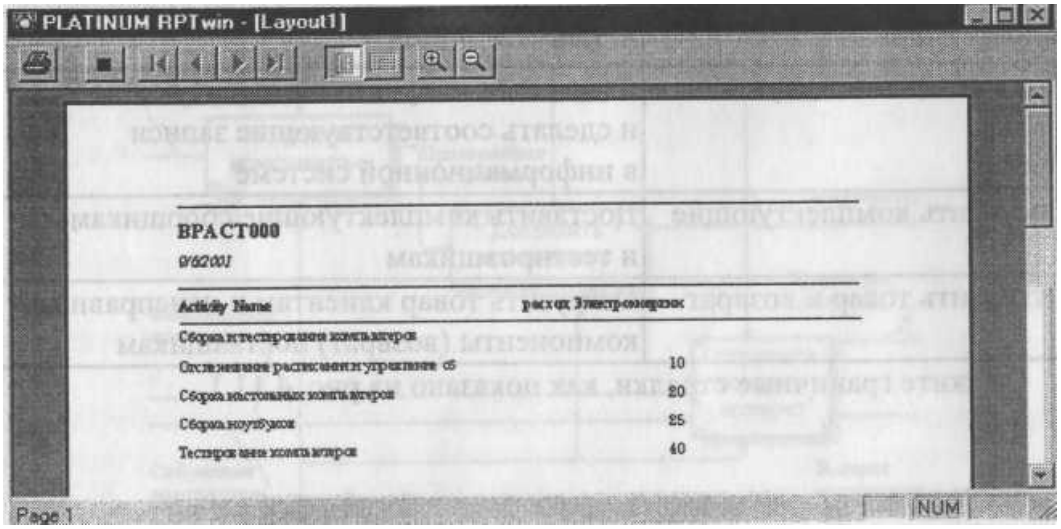



*Рис. 4.10.8. Окно просмотра отчета в RPTwin*

Лабораторная работа 8. Расщепление модели. Слияние расщепленной модели с исходной моделью.

1.Перейдите на диаграмму АО и щелкните правой кнопкой мыщи по работе *"Отгрузка и* ***получение".*** В контекстном меню выберите Split Model.

В появившемся диалоге Split Option установите опцию Enable Merge /Overwrite Option, внесите имя новой модели - "Отгрузка и получение" и щелкните по ОК.

Обратите внимание, что у работы ***"Отгрузка и получение"*** появилась стрелка вызова. BPwin создал также новую модель "Отгрузка и получение".

2.Внесите свойства новой модели:

Time Frame: AS-IS;

Purpose: Документировать работу ***"Отгрузка и получение";***

Viewpoint: Начальник отдела;

Definition: Модель создается для иллюстрации возможностей BPwin по расщеплению и слиянию моделей

Scope: Работы по получению комплектующих и отправке готовой продукции.

3.Декомпозируйте контекстную работу на 3 работы (табл. 4.11.1).

*Таблица 4.11.1. Декомпозиция работы "Отгрузка и получение"*

| *Activity Name* | *Activity Definition* |
|---|---|
| Получить комплектующие | Физически получить комплектующие и сделать соответствующие записи в информационной системе |
| Доставить комплектующие | Доставить комплектующие сборщикам и тестировщикам |
| Отгрузить товар и возврат | Отгрузить товар клиентам и неисправные компоненты (возврат) поставщикам |

Свяжите граничные стрелки, как показано на рис. 4.11.1.


*Рис. 4.11.1. Внутренние стрелки на декомпозиции работы "Отгрузка и получение"*

5. Внесите следующие внутренние и граничные стрелки (табл. **4.11**.2).

*Таблица 4.11.2. Внутренние и граничные стрелки на декомпозиции работы "Отгрузка и получение"*

| *Arrow Name* | *Arrow Definition* |
|---|---|
| Возврат поставщику | Неисправные компоненты |
| Компоненты | *Выберите название из списка (словаря)* |
| Компоненты от поставщика | |
| Проверенные компоненты | Проверенные и подготовленные для передачи сборщикам и тестировщикам компоненты |

6. Туннелируйте граничные стрелки (Resolve Border Arrow). Результат выполнения упражнения показан на рис. 4.11.2.

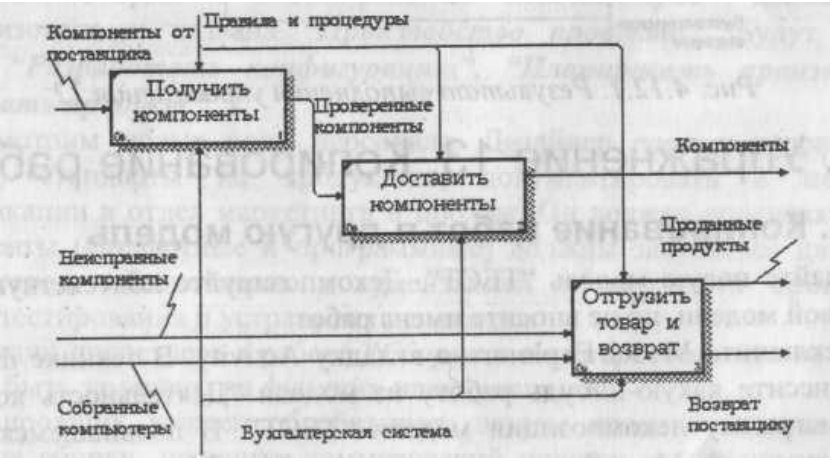



*Рис. 4.11.2. Результат выполнения упражнения 11*

1.Перейдите в модель "Деятельность компании". На диаграмме AO щелкните правой кнопкой мыши по работе ***"Отгрузка и получение".***

**В** контекстном меню выберите Merge Model. В появившемся диалоге Merge Model установите опцию Cut/Paste entire dictionaries и щелкните **по** OK.

Обратите внимание, что у работы ***"Отгрузка и получение"*** исчезла стрелка вызова и появилась новая декомпозиция.

Появились новые стрелки с квадратными скобками. Туннелируйте эти стрелки (Resolve Border Arrow).

2.На диаграмме AO туннелируйте и свяжите стрелки согласно рис. 4.12.1.

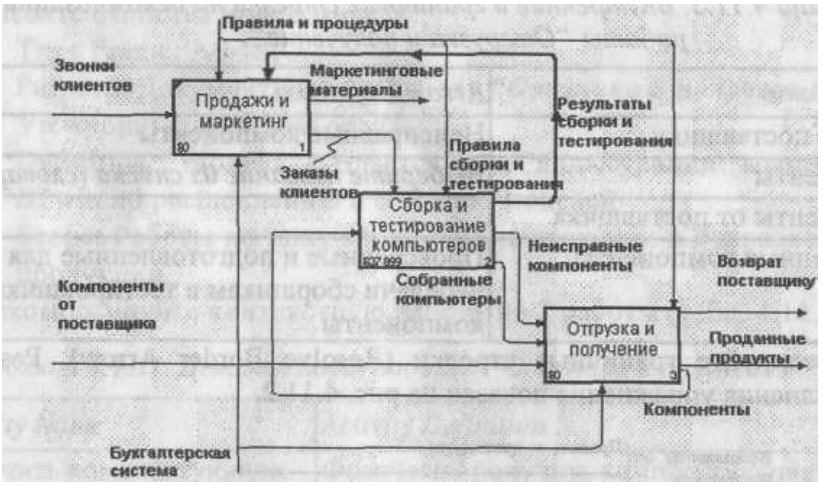



*Рис. 4.12.1. Результат выполнения упражнения 1*

## Лабораторная работа 9. Создание модели TO-BE.

Модель TO-BE создается на основе анализа модели AS-IS. Анализ может проводиться как по формальным признакам (отсутствие выходов или управлений у работ, отсутствие обратных связей и т. д.), так и по неформальным - на основе знаний предметной области.

Допустим, в результате анализа принимается решение реорганизовать функции производства и тестирования компьютеров и оставить функциональности ***"Продажи и маркетинг"*** и ***"Отгрузка и получение"*** пока без изменений.

Принято решение сформировать отдел дизайна, который должен формировать конфигурацию компьютеров, разрабатывать корпоративные стандарты, подбирать приемлемых поставщиков, разрабатывать инструкции по сборке, процедуры тестирования и устранения неполадок для всего производственного отдела.

Работа ***"Сборка и тестирование компьютеров"*** должна быть реорганизована и названа *"Производство продукта".* Будут созданы работы ***"Разработать конфигурацию", "Планировать производство"*** и ***"Собрать продукт".***

Рассмотрим новые роли персонала. Дизайнер должен разрабатывать систему, стандарты на продукцию, документировать и передавать спецификации в отдел маркетинга и продаж. Он должен определять, какие компоненты (аппаратные и программные) должны закупаться для сборки компьютеров, обеспечивать документацией и управлять процедурами сборки, тестирования и устранения неполадок.

Функции диспетчера в работе *"Сборка и тестирование компьютеров"* должны быть заменены на функции планировщика.

Планировщик должен обрабатывать заказы клиентов и генерировать заказы на сборку, получить коммерческий прогноз из отдела маркетинга и формировать требования на закупку компонентов и собирать информацию от поставщиков.

Диспетчер должен составлять расписание производства на основании заказов на сборку, полученных в результате работы *"Планировать производство",* получать копии заказов клиентов и отвечать за упаковку и комплектацию заказанных" компьютеров, передаваемых в работу ***"Отгрузка и получение".***

## Лабораторная работа 10. Создание диаграммы DFD.

При оформлении заказа важно проверить, существует ли такой клиент в базе данных и, если не существует, внести его в базу данных и затем оформить заказ. Оформление заказа начинается со звонка клиента. В процессе оформления заказа база данных клиентов может просматриваться и редактироваться. Заказ должен включать как информацию о клиенте, так и информацию о заказанных продуктах. Оформление заказа подразумевает чтение и запись информации о прочих заказах.

В процессе декомпозиции согласно правилам DFD необходимо преобразовать граничные стрелки во внутренние, начинающиеся и заканчивающиеся на внешних ссылках.

Декомпозируйте работу ***"Оформление** заказов"* на диаграмме A2.

В диалоге Activity Box Count выберите количество работ 2 и нотацию DFD (рис. 4.15.1).

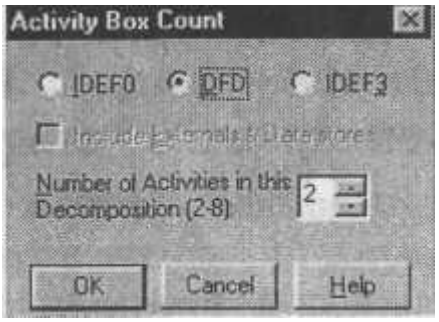



*Рис. 4.15.1. Выбор нотации DFD в диалоге Activity Box Count*

3.Щелкните по OK и внесите в новую диаграмму DFD A22 имена работ:

Проверка и внесение клиента;

Внесение заказа.

4.Используя кнопкуI на палитре инструментов, внесите хранилища

данных:

Список клиентов;

Список продуктов;

Список заказов.

Удалите граничные стрелки с диаграммы DFD A22.

Используя кнопку

на палитре инструментов, внесите внешнюю ссылку:

Звонки клиентов.

Создайте внутренние ссылки согласно рис. 4.15.2. При именовании стрелок используйте словарь.


*Рис. 4.15.2. Диаграмма A22*

8.Обратите внимание, что стрелки ***"Информация о клиентах"*** и ***"Заказы клиентов"*** двунаправленные. Для того чтобы сделать стрелку двунаправленной, щелкните правой кнопкой по стрелке, выберите в контекстном меню пункт Style и во вкладке Style выберите опцию Bidirectional.

9.На родительской диаграмме A2 туннелируйте (Change to Tunnel) стрелки, подходящие и исходящие из работы *"Оформление заказов"* (рис. 4.15.3).


*Рис. 4.15.3. Работа "Оформление заказов" на диаграмме A2*

Лабораторная работа 11. Отображение модели данных в инструментальных средствах проектирования.

**Отображение модели данных в ERwin**

**2.1.1. Физическая и логическая модель данных**

CA ERwin Data Modeler (далее ERwin) - CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных.

Работа с программой начинается с создания новой модели, для которой нужно указать тип и целевую СУБД (рис.1).

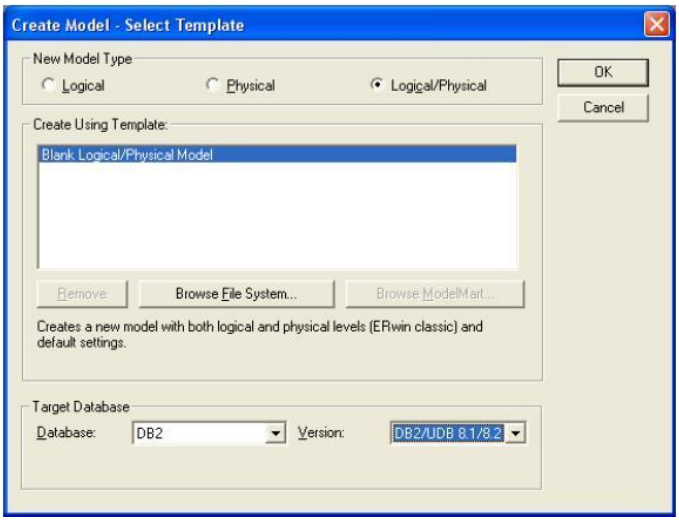


Рисунок 1.Создание новой модели

ERwin позволяет создавать логическую, физическую модели и модель, совмещающую логический и физический уровни.

**Логический уровень** - это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире (например "Постоянный клиент", "Отдел" или "Заказ").

Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

**Физический уровень** зависит от конкретной СУБД. В физической модели содержится информация о всех объектах БД. Физическая модель зависит от конкретной реализации СУБД. Одной и той же логической модели могут соответствовать несколько разных физических моделей.

На логическом уровне ERwin поддерживает две нотации (IE и IDEF1X), на физическом - три (IE, IDEF1X и DM). Далее будет рассматриваться работа с ERwin в нотации IDEF1X.

Переключение между логической и физической моделями данных осуществляется через список выбора на стандартной панели (рис.2).

Рисунок 2.Переключение между уровнями

**Примечание**. В созданной модели с настройками по умолчанию некорректно отображаются русские символы. Чтобы устранить этот недостаток, необходимо подкорректировать используемые в модели шрифты. Для этого необходимо зайти в меню Format -> Default Fonts & Colors, последовательно пройтись по всем вкладкам, в качестве шрифта выбрав любой шрифт, название которого заканчивается на CYR (например, Arial CYR), и выставив переключатель Apply To в значение All Objects.

Лабораторная работа 12. Создание логической модели данных.

Логический уровень модели данных

Для создания на логическом уровне сущностей и связей между ними предназначена панель *Toolbox*:


Рисунок 3.Панель Toolbox

| Вид кнопки | Назначение кнопки |
|---|---|
|  | Создание новой сущности. Для этого нужно щелкнуть по кнопке и затемпо свободному месту на модели |
|  | Создание категории. Для установки категориальной связи нужно щелкнуть по кнопке, далее - по сущности-родителю, и затем - по сущности-потомку. |
|  | Создание идентифицирующей связи. Для связывания двух сущностей нужно щелкнуть по кнопке, далее - по сущности-родителю, затем - по сущности-потомку. |
|  | Создание связи "многие ко многим" |
|  | Создание неидентифицирующей связи |

После создания сущности ей нужно задать атрибуты. Для этого нужно дважды щелкнуть по ней или в контекстном меню выбрать пункт *Attributes* (рис.4).

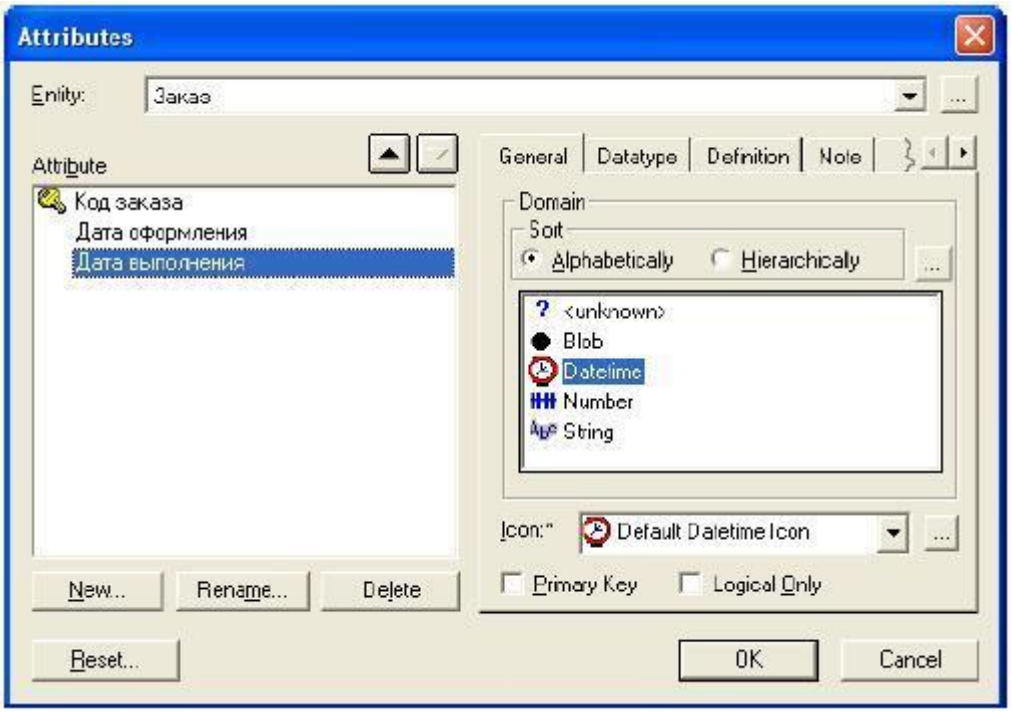



Рисунок 4.Окно атрибутов выбранной сущности

В появившемся окне можно просмотреть и отредактировать информацию о созданных атрибутах, создать новые. Здесь же задается первичный ключ. Для создания нового атрибута следует нажать кнопку *New*. В появившемся окне можно выбрать тип атрибута (BLOB, дата/время, число, строка), задать имя атрибута (*Attribute Name*) и имя столбца (Column Name), который будет соответствовать атрибуту на физическом уровне (рис.5).

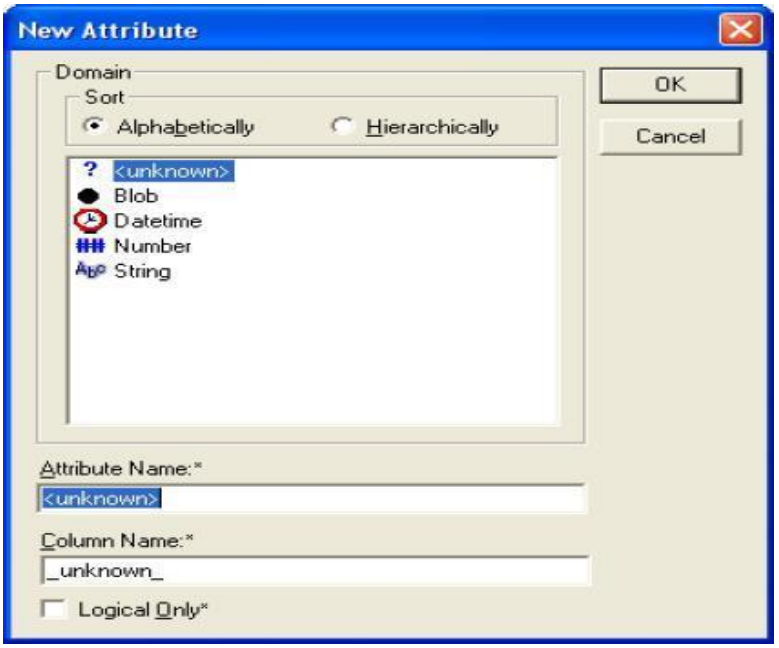



Рисунок 5.Окно создания атрибута

После создания сущностей создаются связи между ними. При создании идентифицирующей связи атрибуты, составляющие первичный ключ сущности-родителя, мигрируют в состав первичного ключа сущности-потомка, при создании неидентифицирующей связи - просто в состав атрибутов сущности-потомка. Задать свойства связи или поменять ее тип можно дважды щелкнув по ней или выбрав в контекстном меню пункт *Relationship Properties* (рис. 6). Здесь во вкладке *General* можно задать имя связи (в направлении родитель-потомок и потомок-родитель), мощность связи (ноль, один или больше; один и больше (P); ноль или один (Z); точно (конкретное число) ), поменять тип связи. Во вкладке *RI Action* можно задать ограничения целостности.

Пример логической модели базы данных приведен на рис. 7.

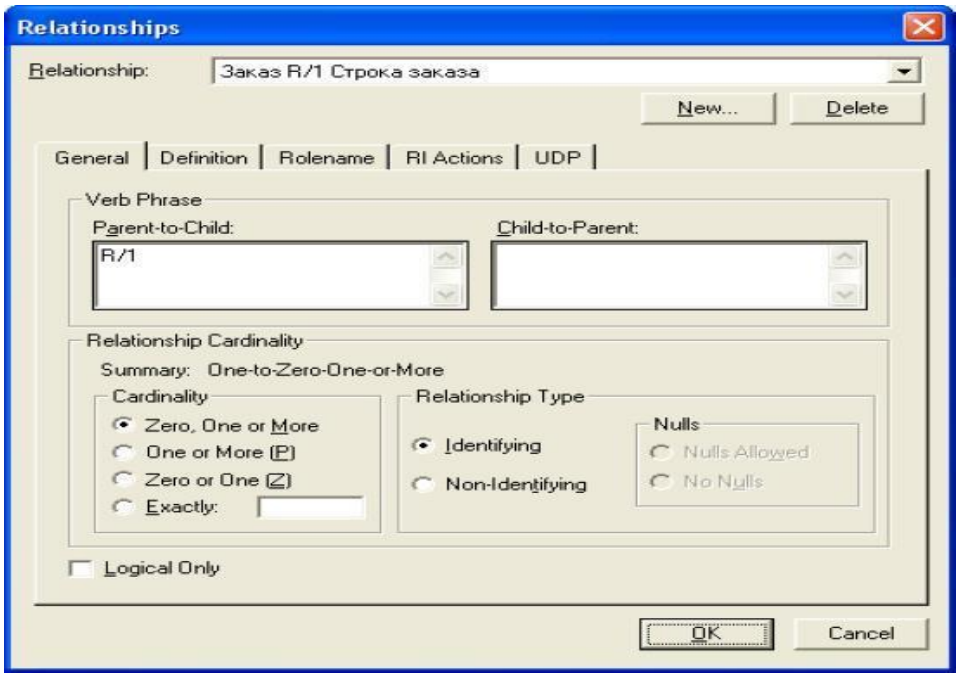



Рисунок 6.Окно свойств связи


Рисунок 7.Пример логической схемы БД

Лабораторная работа 13. Создание физической модели данных.

При переключении с логического уровня на физический автоматически будет создана физическая схема базы данных (рис.8)


Рисунок 8.Автоматически созданная физическая схема БД

Ее можно дополнить, отредактировать или изменить. Принципы работы с физической схемой аналогичны принципам работы с логической схемой.

По готовой физической схеме можно сгенерировать скрипты для выбранной СУБД. Для этого предназначен пункт меню *Tools -> Forward Engineering/Schema Generation* (рис.9).

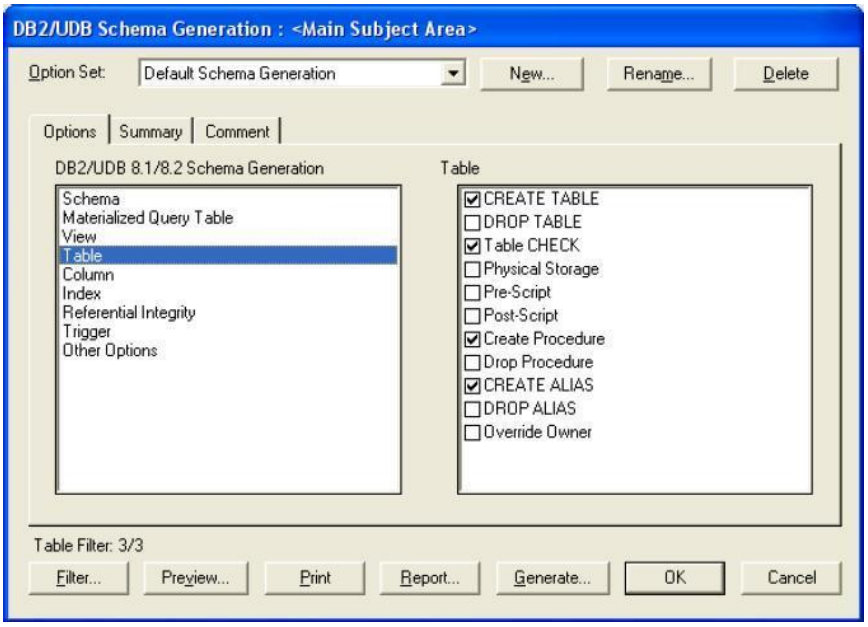



Рисунок 9.Окно генерации SQL-скриптов для целевой СУБД__

Здесь можно указать, какие именно скрипты следует генерировать, предварительно просмотреть их и непосредственно сгенерировать (при этом ERwin произведет подключение к целевой СУБД и в автоматическом режиме выполнит все SQL-скрипты).

Лабораторная работа 14. Разработка диаграммы вариантов использования и редактирование свойств ее элементов.

Особенности разработки диаграмм вариантов использования в среде IBM Rational Rose 2003

Работа над моделью в среде *IBM* Rational Rose начинается с общего анализа проблемы и построения *диаграммы вариантов использования*, которая отражает функциональное назначение проектируемой программной системы. Для вновь создаваемого проекта можно воспользоваться мастером типовых проектов, если он установлен в данной конфигурации. Мастер типовых проектов доступен из *меню* **File>New** (*Файл*>Новый) или при первоначальной загрузке программы *IBM* Rational Rose 2003. В случае разработки проекта, для которого не известна или не выбрана технология его реализации, следует отказаться от мастера, в результате чего появится *рабочий интерфейс программы IBM* Rational Rose 2003 с чистым окном активной *диаграммы классов* и именем проекта untitled по умолчанию.

В качестве проекта далее будет рассматриваться модель системы управления банкоматом. Достоинством этого проекта является то, что он не требует специального описания *предметной области*, поскольку предполагает интуитивное знакомство читателей с особенностями функционирования банкомата. При этом разрабатываемая модель системы управления банкоматом используется в качестве сквозного примера, в рамках которого иллюстрируются особенности разработки различных диаграмм языка *UML* в среде *IBM* Rational Rose 2003. Для изменения имени проекта, предложенного программой по умолчанию, следует сохранить модель во внешнем файле на диске, например, под именем ATMmodel.mdl.

В этом случае изменится имя в строке заголовка и имя проекта в иерархическом представлении модели в браузере проекта.

Как и другие программы, *IBM* Rational Rose позволяет настраивать глобальные параметры среды, такие как выбор шрифтов и цвета для представления различных элементов модели. Настройка шрифтов, цвета линий и графических элементов производится через операцию главного *меню*: **Tools>Options** (Инструменты>Параметры). Характерной особенностью среды является возможность работы с символами кириллицы. Однако следует заметить, что при спецификации элементов модели с последующей генерацией текста программного кода следует записывать имена и свойства классов, *ассоциаций*, атрибутов, операций и компонентов символами того языка, который поддерживается соответствующим языком программирования.

Для разработки *диаграммы вариантов использования* модели в среде *IBM* Rational Rose 2003 необходимо активизировать соответствующую диаграмму в окне диаграммы. Это можно сделать следующими способами:

- раскрыть представление *вариантов использования* **Use Case View** в браузере проекта и дважды щелкнуть на пиктограмме **Main** (Главная);
- с помощью операции главного меню **Browse>Use Case Diagram** (Браузер>Диаграмма вариантов использования).

При этом появляется новое окно с чистым *рабочим листом диаграммы вариантов использования* и специальная *панель инструментов*, содержащая кнопки с изображением графических элементов, необходимых для разработки *диаграммы вариантов использования*. Назначение отдельных кнопок данной панели можно узнать также из всплывающих подсказок, которые появляются, если подвести и задержать на некоторое время *указатель* мыши над той или иной кнопкой (табл. 3.1).

Таблица 3.1. Назначение кнопок специальной панели инструментов для диаграммы вариантов использования

| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
|---|---|---|
|  | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
|  | *Text Box* | Добавляет на диаграмму текстовую область |
|  | Note | Добавляет на диаграмму примечание |
|  | *Anchor* Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
|  | Package | Добавляет на *диаграмму пакет* |
|  | Use Case | Добавляет на диаграмму вариант использования |
|  | *Actor* | Добавляет на диаграмму актера |
|  | *Unidirectional* Association | Добавляет на диаграмму направленную *ассоциацию* |
|  | *Dependency* or *Instantiates* | Добавляет на диаграмму отношение зависимости |
|  | Generalization | Добавляет на диаграмму отношение обобщения |

На специальной панели инструментов по умолчанию присутствует только часть кнопок с пиктограммами элементов, которые могут быть использованы для построения диаграммы. Добавить кнопки с пиктограммами других графических элементов, например, таких как *бизнес-вариант использования* (business *use case*), *бизнес-актер* (business *actor*), сотрудник (business *worker*), или удалить ненужные кнопки можно с помощью настройки специальной панели инструментов.

Открыть *диалоговое окно* настройки специальных панелей инструментов для диаграмм в среде *IBM* Rational Rose 2003 можно с помощью *операции* главного *меню*: **Tools>Options** (Инструменты>Параметры), раскрыв вкладку **Toolbars** (Панели инструментов) и нажав соответствующую кнопку (например, **Use Case diagram** ) в группе опций **Customize Toolbars** (Настройка панелей инструментов). Это окно настройки также можно открыть с помощью *операции* контекстного *меню* **Customize** (Настройка) при позиционировании курсора на специальной панели инструментов (рис. 3.1).

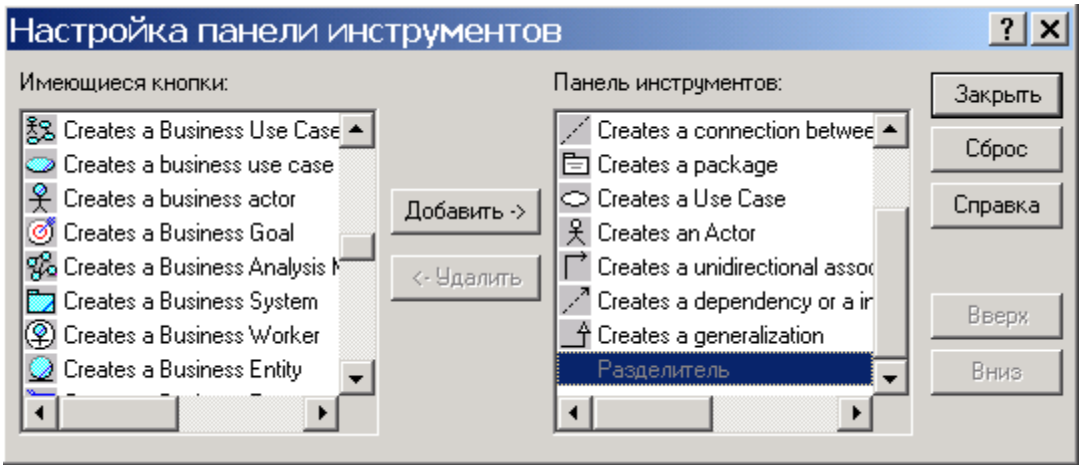



**Рис. 3.1.** Диалоговое окно настройки специальной панели инструментов для диаграммы вариантов использования

Для добавления необходимых кнопок на панель следует выделить их в левом окне со списком пиктограмм графических элементов, после чего нажать кнопку **Добавить** в центре диалогового окна. Для удаления ненужных кнопок с панели инструментов следует выделить их в правом окне со списком пиктограмм графических элементов, после чего нажать кнопку **Удалить** в центре диалогового окна. Для восстановления набора пиктограмм по умолчанию можно нажать кнопку **Сброс**. После настройки специальной панели инструментов соответствующее окно следует закрыть нажатием на кнопку **Закрыть**.

Добавление актера на диаграмму вариантов использования и редактирование его свойств

Для добавления актера на диаграмму *варианта использования* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы актера на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение актера с маркерами изменения его геометрических размеров и предложенным программой именем по умолчанию NewClass. Для разрабатываемой модели банкомата предложенное программой имя актера следует изменить на Клиент Банкомата (рис. 3.2).

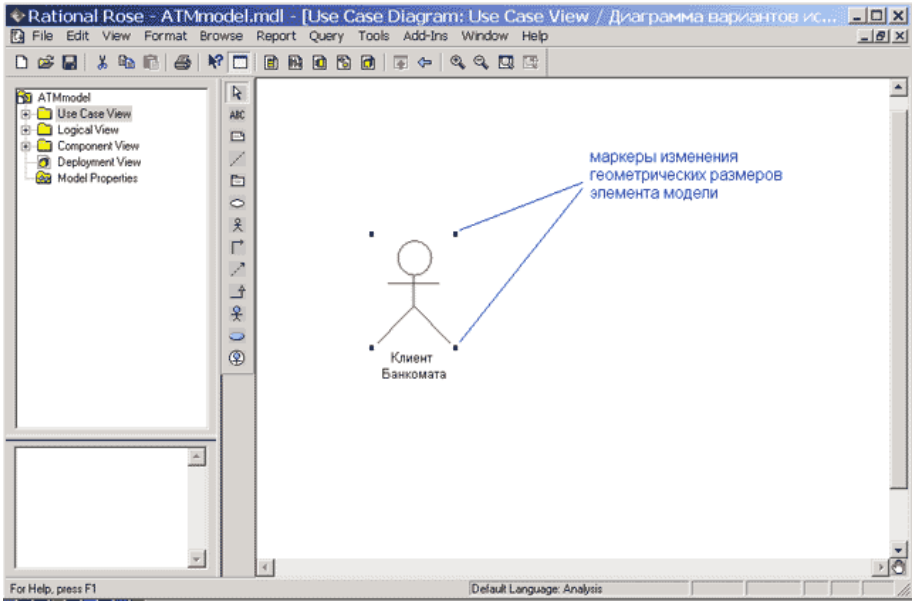



**Рис. 3.2.** Диаграмма вариантов использования после добавления на нее актера

Чтобы изменить расположение изображения графического элемента модели, следует щелчком левой кнопки мыши выделить его в рабочей области диаграммы, и, не отпуская левой кнопки, переместить в нужное *место* диаграммы. При этом выделенный элемент визуально отличается от остальных наличием маркеров изменения его

геометрических размеров в форме небольших черных квадратов. Более точное перемещение элемента можно осуществить с помощью стрелок: ← , ↑ , → , ↓ на клавиатуре.

Чтобы изменить графические размеры изображения элемента модели, прежде всего, следует щелчком левой кнопки мыши выделить его в рабочей области диаграммы. Далее необходимо подвести *указатель* мыши к нужному маркеру геометрических размеров элемента и нажать левую кнопку мыши. В результате этих действий появится пунктирный *прямоугольник*, изображающий границы выбранного геометрического элемента. После чего, не отпуская левой кнопки мыши, следует диагонально изменить размеры этого прямоугольника нужным образом (рис. 3.3).

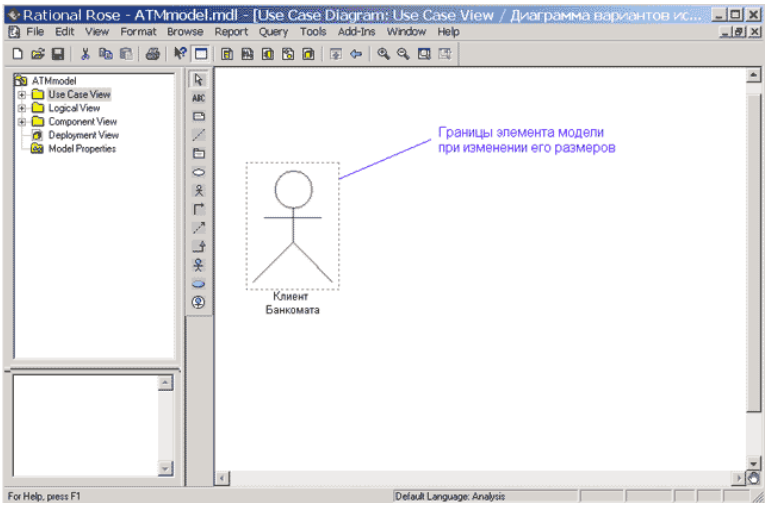



**Рис. 3.3.** Диаграмма вариантов использования при изменении графических размеров актера

Имя размещенного на диаграмму элемента разработчик может изменить либо сразу после добавления элемента на диаграмму, либо в ходе последующей работы над проектом. Для любого графического элемента модели по щелчку правой кнопкой мыши на выбранном элементе вызывается контекстное *меню* данного элемента, среди операций которого имеется *пункт* **Open Specification** (Открыть спецификацию). В этом случае появляется дополнительное *диалоговое окно* со специальными вкладками, в поля ввода которых можно занести всю информацию по данному элементу. Для добавленного актера Клиент Банкомата окно спецификации свойств выглядит следующим образом (рис. 3.4).

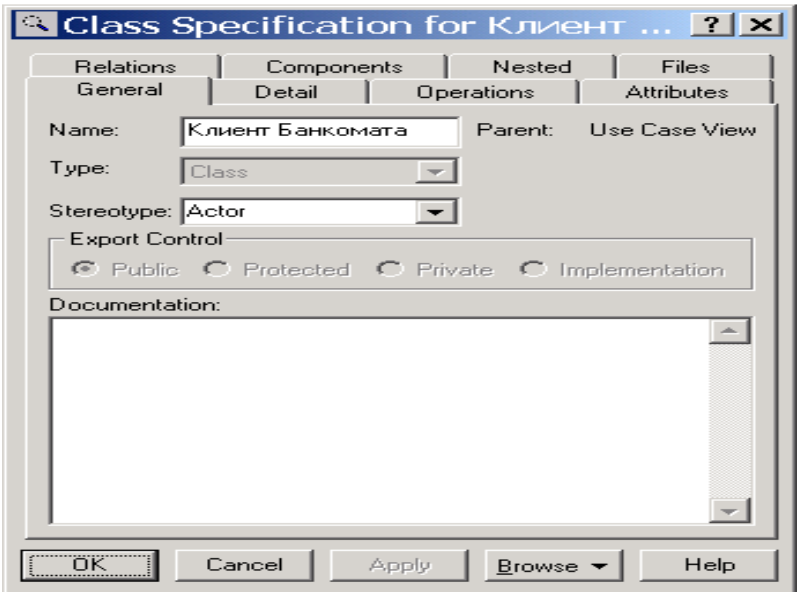



**Рис. 3.4.** Диалоговое окно спецификации свойств актера Клиент Банкомата

Следует отметить, что открыть *диалоговое окно* спецификации свойств любого элемента модели можно также двойным щелчком левой кнопкой мыши на графическом изображении этого элемента на диаграмме. Хотя в среде *IBM* Rational Rose *актер* является классом, для него некорректно специфицировать атрибуты и *операции*, поскольку *актер* является внешней по отношению к разрабатываемой системе сущностью.

Для актера Клиент Банкомата можно уточнить его назначение в модели. С этой целью следует изменить его *стереотип* и добавить текст документации. Для изменения *стереотипа* во вложенном списке **Stereotype** нужно

выбрать строку **Business Actor** ( *бизнес-актер* ). Для добавления текста документации в секцию **Documentation** следует ввести текст: "Любое физическое лицо, пользующееся услугами банкомата" и нажать кнопку **Apply** (Применить) или **OK**. После изменения данных свойств актера Клиент Банкомата окно спецификации свойств будет выглядеть следующим образом (рис. 3.5).

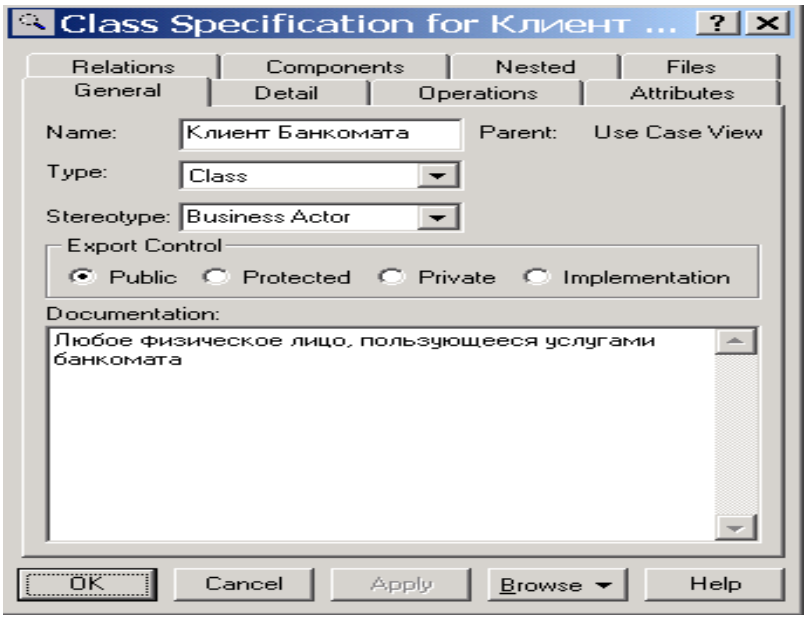



**Рис. 3.5.** Диалоговое окно спецификации свойств после изменения стереотипа и добавления текста документации для актера Клиент Банкомата

Добавление и редактирование варианта использования

Для добавления *варианта использования* на диаграмму нужно с помощью левой кнопки мыши нажать кнопку с изображением *варианта использования* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте диаграммы. На диаграмме появится изображение *варианта использования* с маркерами изменения его геометрических размеров и предложенным программой именем по умолчанию NewUseCase. Для разрабатываемой модели банкомата предложенное программой имя *варианта использования* следует изменить на Снятие наличных по кредитной карточке (рис. 3.6).

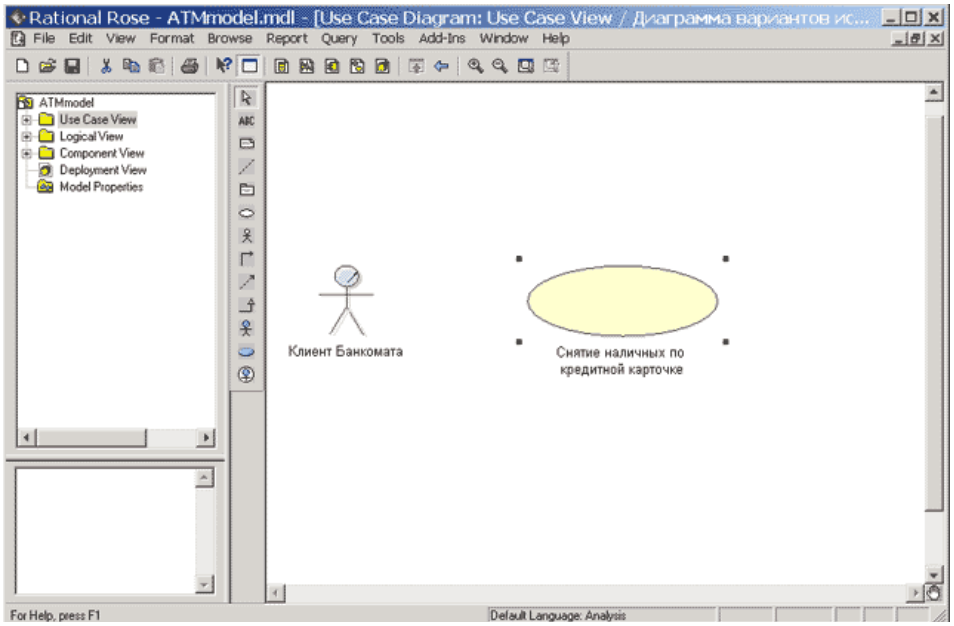



**Рис. 3.6.** Диаграмма вариантов использования после добавления на нее варианта использования

Для уточнения свойств данного *варианта использования* следует открыть *диалоговое окно* спецификации его свойств, например, с помощью двойного щелчка левой кнопкой мыши на изображении этого элемента на диаграмме. Для изменения *стереотипа* во вложенном списке **Stereotype** нужно выбрать строку **Business Use Case**. Для добавления текста документации в секцию **Documentation** следует ввести текст: "Основной *вариант использования* для разрабатываемой модели банкомата" и нажать кнопку **Apply** (Применить) или **OK**. После изменения данных свойств *варианта использования* окно спецификации его свойств будет выглядеть следующим образом (рис. 3.7).

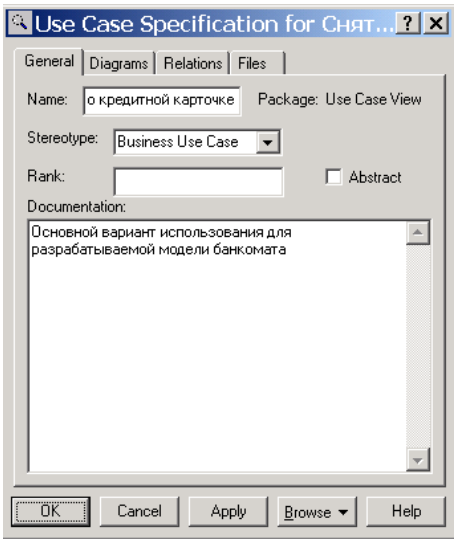



**Рис. 3.7.** Диалоговое окно спецификации свойств варианта использования Снятие наличных по кредитной карточке Добавление ассоциации

Для добавления *ассоциации* между актером и вариантом использования на диаграмму нужно с помощью левой кнопки мыши нажать на специальной панели инструментов кнопку с изображением пиктограммы направленной *ассоциации*, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении актера на диаграмме и отпустить ее на изображении *варианта использования*. В результате этих действий на диаграмме появится изображение *ассоциации*, соединяющей актера с вариантом использования (рис. 3.8).

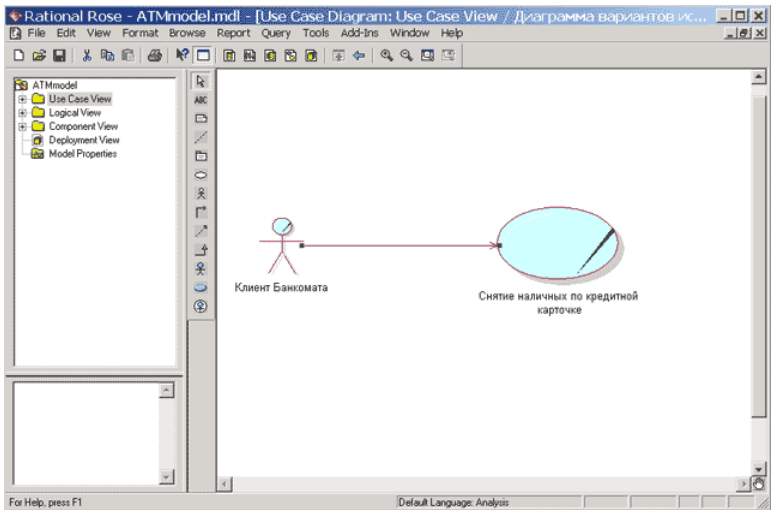



**Рис. 3.8.** Диаграмма вариантов использования после добавления на нее направленной ассоциации

При необходимости можно сделать направленную *ассоциацию* ненаправленной, для чего следует воспользоваться диалоговым окном свойств *ассоциации*. Открыть это окно можно, например, двойным щелчком на изображении линии *ассоциации* на диаграмме, после чего убрать отметку строки выбора **Navigable** (*Навигация*) на вкладке **Role A Detail** (Детальные свойства концевой точки *ассоциации* A).

Добавление отношения зависимости и редактирование его свойств

Для добавления отношения зависимости между двумя вариантами использования на диаграмму необходимо предварительно рассмотренным выше способом добавить второй *вариант использования* с именем Проверка ПИН-кода. После этого с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы зависимости на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении *варианта использования* Снятие наличных по кредитной карточке и отпустить ее на изображении *варианта использования* Проверка ПИН-кода. В результате этих действий на диаграмме появится изображение отношения зависимости, которое соединяет два выбранных *варианта использования*.

Поскольку *вариант использования* Проверка ПИН-кода выполняется всегда, для добавленного отношения зависимости дополнительно следует указать текстовый *стереотип* <<include>>. Выполнить это можно уже известным способом с помощью диалогового окна спецификации свойств этого отношения и выбора нужного *стереотипа* из предлагаемого списка.

После задания для данного отношения *зависимости стереотипа* <<include>> текст этого *стереотипа* в угловых скобках появится рядом с изображением пунктирной линии зависимости, связывающей соответствующие варианты использования (рис. 3.9). С целью лучшей визуализации диаграммы текстовую область *стереотипа* можно переместить в нужное *место* диаграммы. Выполнить это можно с помощью общего способа перемещения графических элементов модели, который был рассмотрен ранее в этой лекции применительно к актеру Клиент Банкомата.

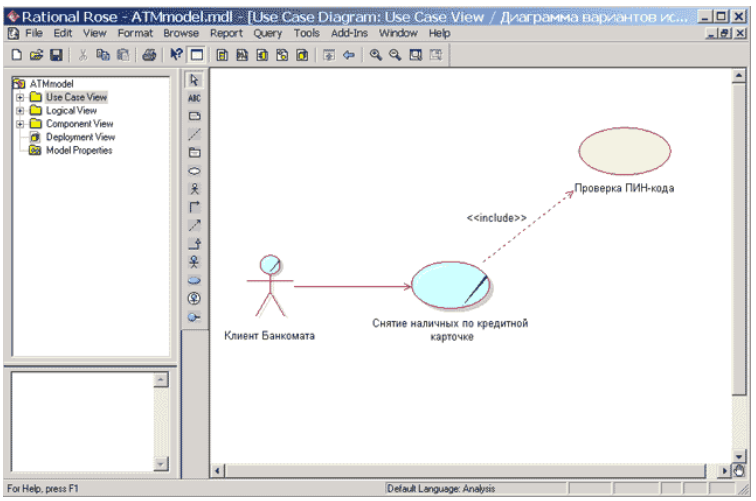



**Рис. 3.9.** Диаграмма вариантов использования после добавления на нее отношения зависимости

Аналогичным образом могут быть добавлены на диаграмму вариантов использования отношения зависимости со стереотипом <<extend>>, которые применяются для моделирования исключений при выполнении отдельных *вариантов использования*.

Окончательное построение диаграммы вариантов использования

К отдельному *варианту использования* можно добавить текстовый *файл* с описанием сценария его выполнения. Для этого необходимо выделить этот *вариант использования* в браузере проекта и выполнить операцию контекстное *меню*: **New>File** (Новый>*Файл*). В результате этого будет вызвано стандартное окно открытия файла, в котором необходимо задать имя предварительно созданного с помощью офисной программы MS *Word* добавляемого файла. После нажатия кнопки **Открыть** пиктограмма добавленного файла появится в браузере проекта ниже соответствующего *варианта использования*. В последующем можно вернуться к редактированию этого файла сценария, выполнив *двойной щелчок* на этой пиктограмме. При этом *файл* сценария будет открыт в соответствующем приложении - в текстовом процессоре MS *Word*.

Для окончательного построения диаграммы *варианта использования* для рассматриваемой модели банкомата следует выполнить следующие действия:

1. Добавить актера с именем Банк, для которого выбрать стереотип **Service** ( *Сервис* ), означающий, что банкомат использует некоторые услуги Банка в качестве *сервиса*.
2. Добавить *вариант использования* Получение справки о состоянии счета, для которого выбрать *стереотип* **Business Use Case** ( *Бизнес-вариант использования* ).
3. Добавить *вариант использования* Блокирование кредитной карточки.
4. Добавить направленную *ассоциацию* от *бизнес-актера* Клиент Банкомата к *варианту использования* Получение справки о состоянии счета.
5. Добавить направленную *ассоциацию* от *варианта использования* Снятие наличных по кредитной карточке к *сервису* Банк.
6. Добавить направленную *ассоциацию* от *варианта использования* Получение справки о состоянии счета к *сервису* Банк.
7. Добавить отношение зависимости со стереотипом <<include>>, направленное от *варианта использования* Получение справки о состоянии счета к *варианту использования* Проверка ПИН-кода.
8. Добавить отношение зависимости со стереотипом <<extend>>, направленное от *варианта использования* Блокирование кредитной карточки к *варианту использования* Проверка ПИН-кода.

Выполнить эти действия предлагается читателям самостоятельно. При этом *отношение* зависимости со стереотипом <<extend>> на данной диаграмме означает следующее. *Вариант использования* Блокирование кредитной карточки будет выполняться только в том случае, если в результате проверки ПИН-кода будет установлено, что соответствующая кредитная карточка утрачена ее владельцем или признана недействительной. Построенная таким образом *диаграмма* вариантов использования будет иметь следующий вид (рис. 3.10).

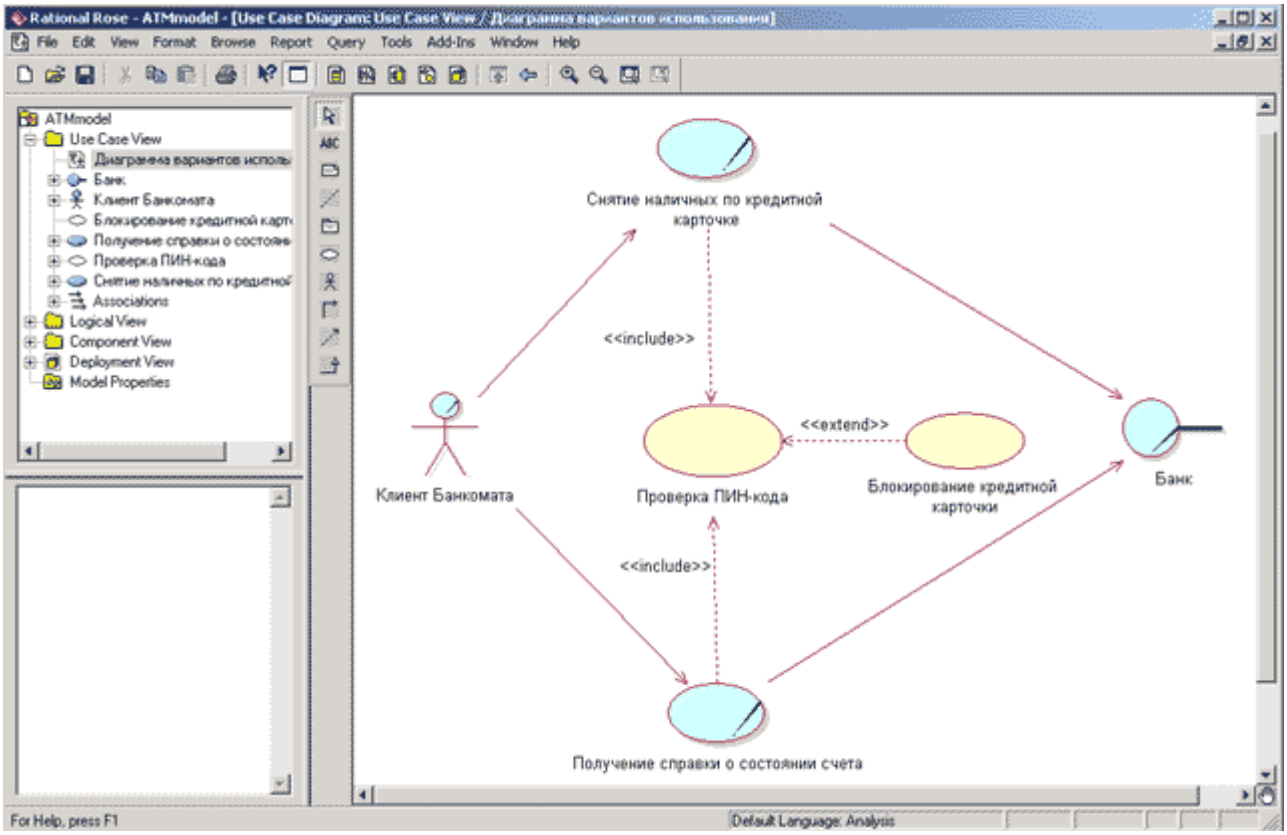



**Рис. 3.10.** Окончательный вид диаграммы вариантов использования для разрабатываемой модели банкоматов

Напомним, что *диаграмма* вариантов использования является высокоуровневым концептуальным представлением модели, поэтому она не должна содержать слишком много *вариантов использования* и актеров. В последующем построенная *диаграмма* может быть изменена посредством добавления новых элементов, таких как варианты использования и актеры, или их удаления.

Для удаления любого графического элемента с диаграммы его следует выделить на диаграмме и нажать клавишу **Delete** на клавиатуре. При этом выделенный элемент будет удален с активной диаграммы, но не из модели. Для удаления элемента не только из диаграммы, но и из модели проекта необходимо выделить удаляемый элемент на диаграмме и воспользоваться операцией главного *меню* **Edit>Delete from Model** (Редактирование>Удалить из модели). Для этой же цели служит комбинация клавиш быстрого доступа: **Ctrl+D**.

При работе с отношениями на диаграмме вариантов использования следует помнить о назначении соответствующих отношений в нотации языка *UML*. Речь идет о том, что если для двух элементов выбранный вид отношения не является допустимым, то в большинстве случаев *программа IBM* Rational Rose 2003 сообщит об этом разработчику, и соответствующая *линия связи* не будет добавлена на диаграмму.

После окончания сеанса работы над проектом выполненную работу необходимо сохранить в файле проекта с расширением ".MDL". Это можно сделать через *меню* **File>Save** (*Файл*>Сохранить) или **File>Save As** (*Файл*>Сохранить как). При этом вся *информация* о проекте, включая диаграммы и спецификации элементов, будет сохранена в одном файле.

## Лабораторная работа 15. Разработка диаграммы классов и редактирование их свойств

Особенности разработки диаграмм классов в среде IBM Rational Rose 2003

*Диаграмма классов* является основным логическим представлением модели и содержит детальную информацию о внутреннем устройстве объектно-ориентированной программной системы или, используя современную терминологию, об архитектуре программной системы. Активизировать рабочее окно диаграммы *классов* можно несколькими способами:

- окно диаграммы *классов* появляется по умолчанию в рабочем окне диаграммы после создания нового проекта;
- щелкнуть на кнопке с изображением диаграммы *классов* на *стандартной панели инструментов*;

- раскрыть логическое представление (Logical View) в браузере проекта и дважды щелкнуть на пиктограмме **Main** (Главная);
- выполнить операцию главного меню: **Browse>Class Diagram** (Обзор>Диаграмма *классов* ).

При этом появляется новое окно с чистым *рабочим листом* диаграммы *классов* и специальная *панель инструментов*, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы *классов* (табл. 4.1). Назначение отдельных кнопок панели можно узнать также из всплывающих подсказок.

Таблица 4.1. Назначение кнопок специальной панели инструментов для диаграммы классов

| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
|---|---|---|
|  | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
|  | *Text Box* | Добавляет на диаграмму текстовую область |
|  | Note | Добавляет на диаграмму примечание |
|  | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
|  | Class | Добавляет на диаграмму *класс* |
|  | Interface | Добавляет на диаграмму *интерфейс* |
|  | *Unidirectional* Association | Добавляет на диаграмму направленную *ассоциацию* |
|  | *Association Class* | Добавляет на диаграмму *ассоциацию класс* |
|  | Package | Добавляет на *диаграмму пакет* |
|  | *Dependency* or *Instantiates* | Добавляет на диаграмму отношение зависимости |
|  | Generalization | Добавляет на диаграмму отношение обобщения |
|  | Realize | Добавляет на диаграмму отношение реализации |

На специальной панели инструментов по умолчанию присутствует только часть пиктограмм элементов, которые могут быть использованы для построения диаграммы *классов*. Добавить кнопки с пиктограммами других графических элементов таких как, например, отношения агрегации и композиции, *шаблон*, *класс* бизнес-сущность, *управляющий класс*, или удалить ненужные кнопки можно с помощью настройки специальной панели инструментов. Соответствующее *диалоговое окно* настройки специальной панели инструментов для диаграммы *классов* можно вызвать аналогично другим панелям с помощью *операции* контекстного *меню* **Customize** (Настройка) при позиционировании курсора на специальной панели инструментов.

Добавление класса на диаграмму классов и редактирование его свойств

Для добавления *класса* на диаграмму *классов* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *класса* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение *класса* с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию NewClass.

Продолжая разработку модели банкомата в качестве сквозного примера проекта, построим для этой модели следующую каноническую диаграмму - диаграмму *классов*. С этой целью следует изменить предложенное по умолчанию имя диаграммы Main на Диаграмма *классов* ATM, а имя добавленного на диаграмму *класса* - на Транзакция Банкомата (рис. 4.1).

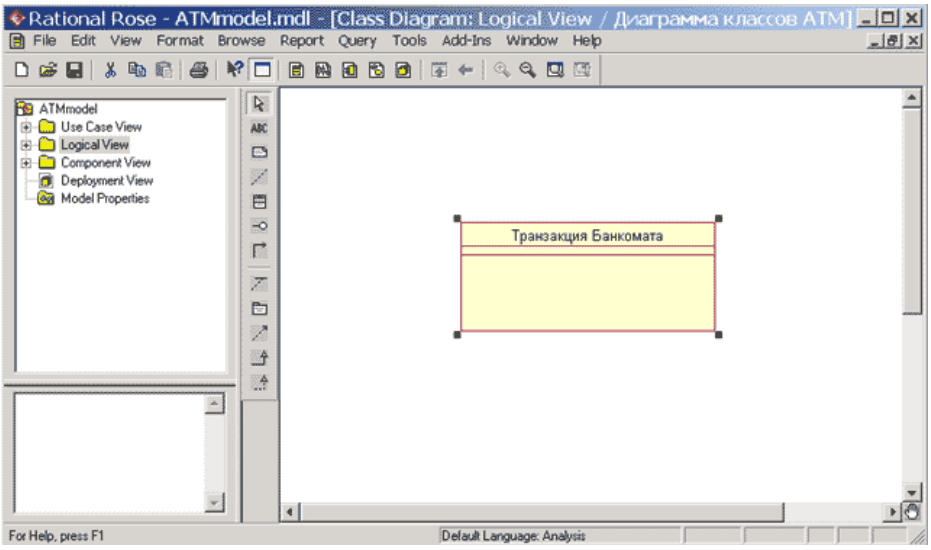



**Рис. 4.1.** Диаграмма классов модели банкомата после добавления на нее класса Транзакция Банкомата

Поскольку разрабатываемая модель банкомата на начальных этапах работы над проектом используется для анализа общей архитектуры проекта и согласования ее с различными участниками рабочей группы, имена *классов*, их атрибутов и операций для большей наглядности и понимания задаются на русском языке с пробелами и записываются символами кириллицы. В последующем по мере выполнения проекта и реализации модели на некотором языке программирования, имена соответствующих *классов*, атрибутов и операций должны быть преобразованы в символы латиницы. При этом имена этих элементов модели должны быть записаны без пробелов. В контексте управляемой моделью архитектуры первую модель еще называют независимой от платформы реализации, а вторую - зависимой от платформы реализации.

Для *класса* Транзакция Банкомата можно уточнить его назначение в модели с помощью указания стереотипа и пояснительного текста в форме документации. С этой целью двойным щелчком левой кнопкой мыши на изображении этого *класса* на диаграмме или в браузере проекта следует открыть *диалоговое окно* спецификации свойств этого *класса* (рис. 4.2) и на вкладке **General** (Общие) выбрать из вложенного списка **Stereotype** стереотип **entity** (сущность).

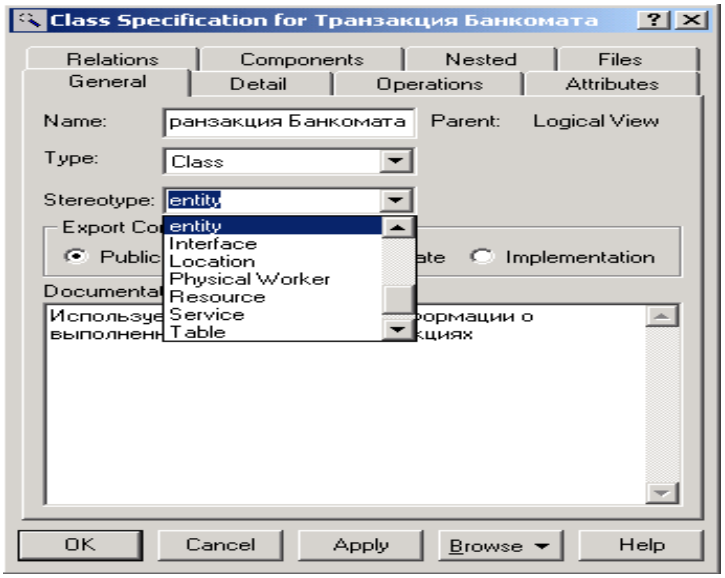



**Рис. 4.2.** Диалоговое окно спецификации свойств класса Транзакция Банкомата при выборе из вложенного списка стереотипа entity

Выбор данного стереотипа означает, что соответствующий *класс* предназначен для хранения информации, которая должна сохраняться в системе после уничтожения объектов данного *класса*. Далее в секцию документации данного *класса* можно ввести поясняющий текст: "Используется для сохранения информации о выполненных банкоматом транзакциях" и нажать кнопку **Apply** или **OK**, чтобы сохранить результаты редактирования свойств

выбранного *класса*. После назначения стереотипа классу *Транзакция* банкомата текст данного стереотипа в угловых скобках появится выше имени данного *класса* (рис. 4.3).

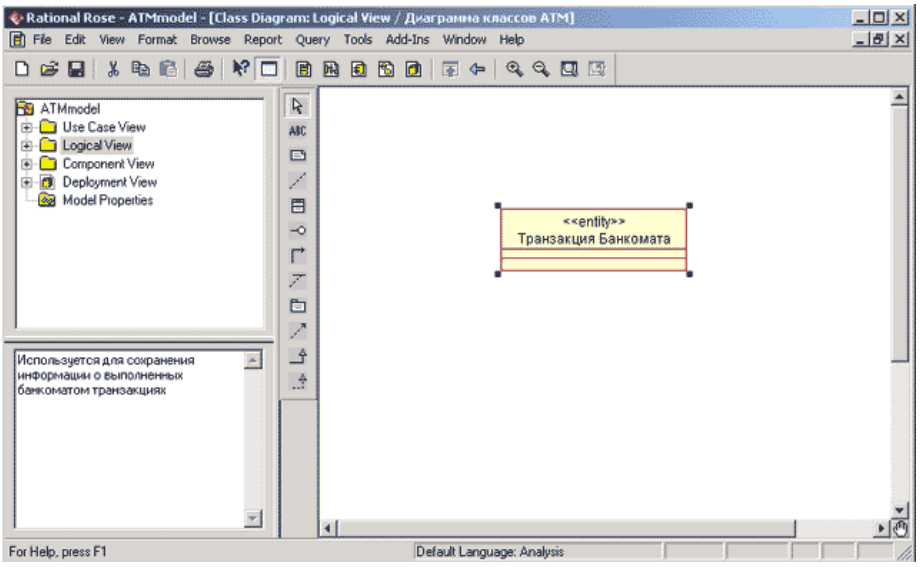



**Рис. 4.3.** Диаграмма классов модели банкомата после выбора стереотипа для класса Транзакция Банкомата

Для отдельного *класса* можно уточнить также и другие его свойства, доступные для редактирования на вкладке **Detail** (Подробно) окна спецификации свойств этого *класса*. Например, на этой вкладке с помощью вложенного списка **Multiplicity** (*Кратность*) можно задать количество объектов или экземпляров данного *класса*, для чего следует выбрать строку с буквой n. Данное *значение* означает, что у *класса Транзакция* банкомата может быть любое конечное число экземпляров (рис. 4.4). *Поле* ввода с именем **Space** (*Пространство*) служит для указания объема абсолютной или относительной памяти, которая требуется, по оценке разработчика, для реализации каждого объекта данного *класса*. Применительно к рассматриваемой модели это *поле* можно оставить пустым.

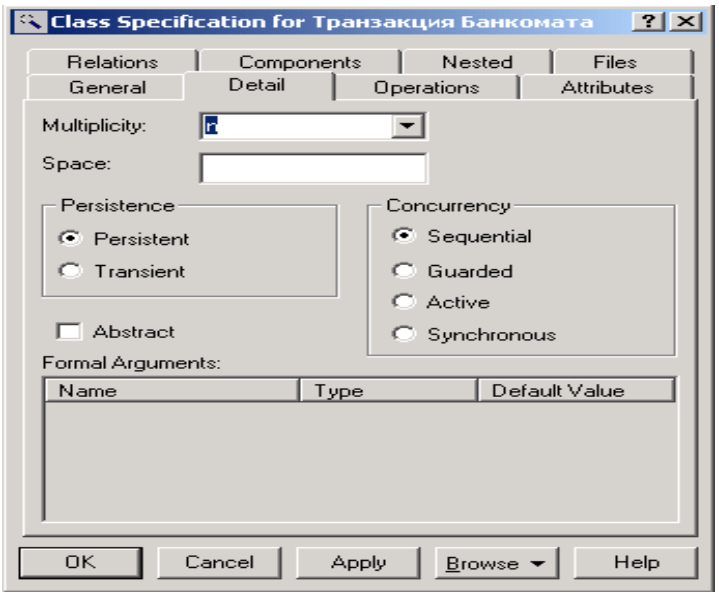



**Рис. 4.4.** Диалоговое окно спецификации свойств класса Транзакция Банкомата, открытое на вкладке Detail (Подробно)

Далее можно задать *устойчивость классов* в группе выбора **Persistence**. При этом выбор свойства **Persistent** (Устойчивый) означает, что *информация* об объектах данного *класса* должна быть сохранена в системе. Выбор свойства **Transient** (Временный) означает, что нет необходимости сохранять информацию об объектах данного *класса* в системе после завершения работы программного приложения. Применительно к рассматриваемой модели следует выбрать свойство **Persistent**.

В группе выбора **Concurrency** (Параллельность) можно специфицировать условия на возможность реализации объектов данного *класса* в параллельных потоках управления. Для выбора могут быть использованы следующие свойства:

- **Sequential** (Последовательный) - свойство по умолчанию, которое означает, что объекты *класса* будут вести себя нормально только при наличии одного потока управления, т. е. соответствующие операции объектов должны выполняться последовательно. В то же время при наличии нескольких потоков управления стабильное поведение объектов *класса* не гарантируется.
- **Guarded** (Безопасный) - означает, что при наличии нескольких потоков управления объекты *класса* будут вести себя ожидаемым от них образом. Для этого объекты в различных потоках должны взаимодействовать друг с другом для того, чтобы гарантировать отсутствие конфликта между ними.
- **Active** (Активный) - означает, что *класс* должен иметь свой собственный поток управления.
- **Synchronous** (Синхронный) - означает, что объекты *класса* будут вести себя ожидаемым от них образом при наличии нескольких потоков управления. При этом нет необходимости во взаимодействии объектов в различных потоках управления, поскольку объекты данного *класса* могут самостоятельно разрешать возможные конфликты.

Для того, чтобы специфицировать *класс* как абстрактный, т.е. не имеющий экземпляров, следует на этой же вкладке выставить отметку в свойстве **Abstract** (Абстрактный). Применительно к рассматриваемой модели для *класса Транзакция* банкомата следует выбрать свойства **Persistent** и **Sequential**, а отметку для свойства **Abstract** оставить пустой.

Следует заметить, что для предотвращения потери информации о разрабатываемой модели и результатов редактирования свойств ее графических элементов необходимо периодически сохранять модель во внешнем файле. Для этого следует выполнить операцию главного *меню*: **File>Save** (*Файл*>Сохранить) или нажать комбинацию клавиш: **Ctrl+S**. Для этой же цели служит соответствующая кнопка на *стандартной панели инструментов* (см. табл. 1.1).

Стереотипы классов и их графическое представление

На разрабатываемой диаграмме *классов* выбран текстовый способ изображения стереотипов *классов*, при котором стереотип записывается в угловых кавычках выше имени соответствующего *класса*. *Программа IBM* Rational Rose 2003 позволяет альтернативно представлять стереотипы в форме специальных графических изображений (как в браузере проекта) или в форме небольших декоративных значков в верхней секции прямоугольника *класса* на диаграмме, а также вообще отказаться от изображения стереотипов.

Изменить изображение стереотипа для отдельного *класса* можно, например, с помощью одной из вложенных операций контекстного *меню*: **Options>Stereotype Display** (Параметры>Изображение стереотипа). В качестве примера можно представить изображение *класса* Транзакция Банкомата в форме специальной графической пиктограммы стереотипа. С этой целью следует выполнить операцию контекстного *меню*: **Options>Stereotype Display>Icon** (Параметры>Изображение стереотипа>Пиктограмма). Соответствующее графическое изображение стереотипа <<*entity*>> для *класса* Транзакция Банкомата в форме пиктограммы будет иметь следующий вид (рис. 4.5, а).

Для сравнения можно выбрать изображение *класса* Транзакция Банкомата в форме декоративного графического стереотипа. С этой целью выполним операцию контекстного *меню*: **Options>Stereotype Display>Decoration** (Параметры>Изображение стереотипа>Декорация). Соответствующее графическое изображение стереотипа <<*entity*>> для *класса* Транзакция Банкомата в форме декорации будет иметь следующий вид (рис. 4.5, б).

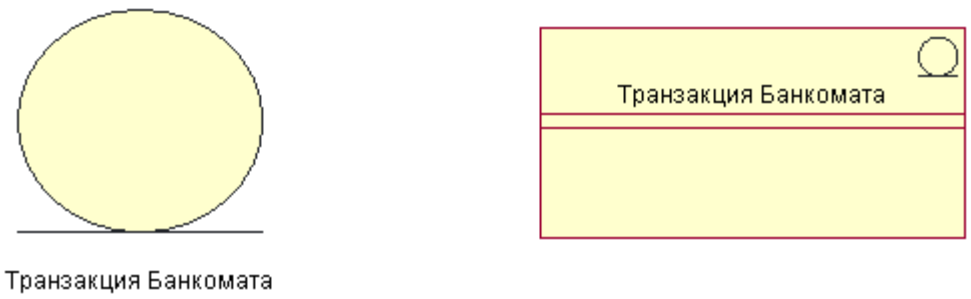



**Рис. 4.5.** Графические способы изображения стереотипа <<entity>> для класса Транзакция Банкомата

Изменить изображение стереотипов одновременно для нескольких *классов* диаграммы можно с помощью одной из вложенных операций главного *меню*: **Format>Stereotype Display** (Формат>Изображение стереотипов). В этом случае необходимо выделить все классы модели в окне диаграммы *классов* или в браузере проекта. Для выделения группы *классов* на диаграмме или в браузере проекта следует, удерживая нажатой клавишу **Ctrl** или **Shift** на клавиатуре, последовательно щелкать на их изображении левой кнопкой мыши.

Выделить все графические элементы на диаграмме *классов*, также как и на любой другой диаграмме модели, можно с помощью выполнения *операции* главного *меню*: **Edit>Select All** (Редактирование>Выделить все) или с помощью комбинации клавиш **Ctrl+A**. Следует отметить, что выбор того или иного способа изображения стереотипов *классов* на диаграмме *классов* определяется разработчиком исходя из его личных предпочтений, и не оказывает влияния на содержательный аспект логического представления модели.

Продолжая разработку модели банкомата, добавим на диаграмму второй *класс* с именем Контроллер Банкомата, для которого в окне спецификации свойств выберем стереотип **control** ( *управляющий класс* ), а в качестве документации введем текст: "Реализует логику функционирования банкомата". При этом атрибуты и *операции* у данного *класса* будут отсутствовать. Соответствующий фрагмент диаграммы *классов* после добавления *управляющего класса* Контроллер Банкомата будет иметь следующий вид (рис. 4.6).

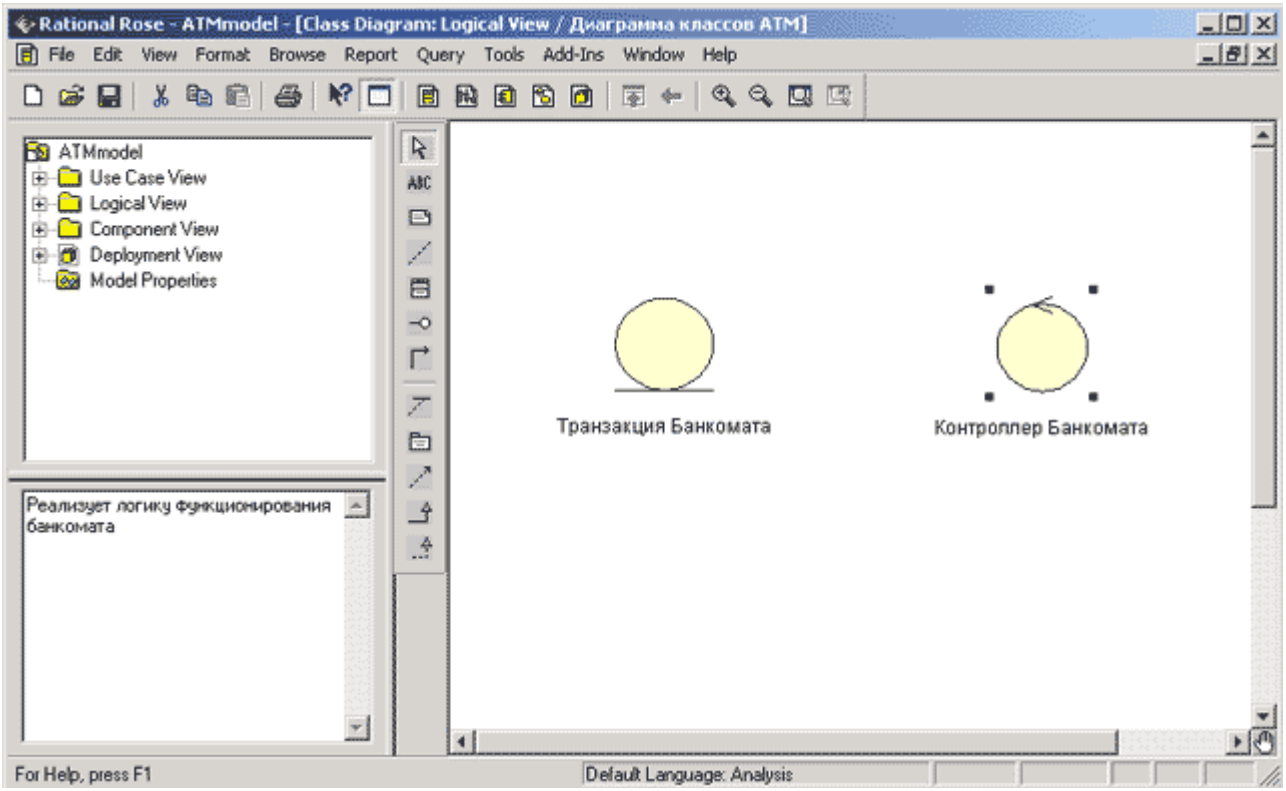



**Рис. 4.6.** Фрагмент диаграммы классов модели банкомата после добавления на нее класса Контролер Банкомата

Продолжая разработку модели банкомата, добавим на диаграмму третий *класс* с именем Устройство чтения карточки, для которого в окне спецификации свойств выберем стереотип **boundary** ( *граничный класс* ). Применение этого стереотипа означает, что данный *класс* находится на границе моделируемой системы, в качестве которой рассматривается модель банкомата. После этого в секцию документации данного *класса* можно ввести поясняющий текст: "Устанавливается на банкомате".

Далее следует добавить *класс* с именем IКонтроллер Банка, для которого выбрать стереотип **Interface** ( *Интерфейс* ), означающий, что банкомат пользуется услугами Банка при обработке своих транзакций. Заметим, что первой буквой в имени этого *класса* является английское "I", которое служит в языке *UML* для указания *интерфейса*. Соответствующий фрагмент диаграммы *классов* после добавления на нее *классов* Устройство чтения карточки и IКонтроллер Банка будет иметь следующий вид (рис. 4.7).

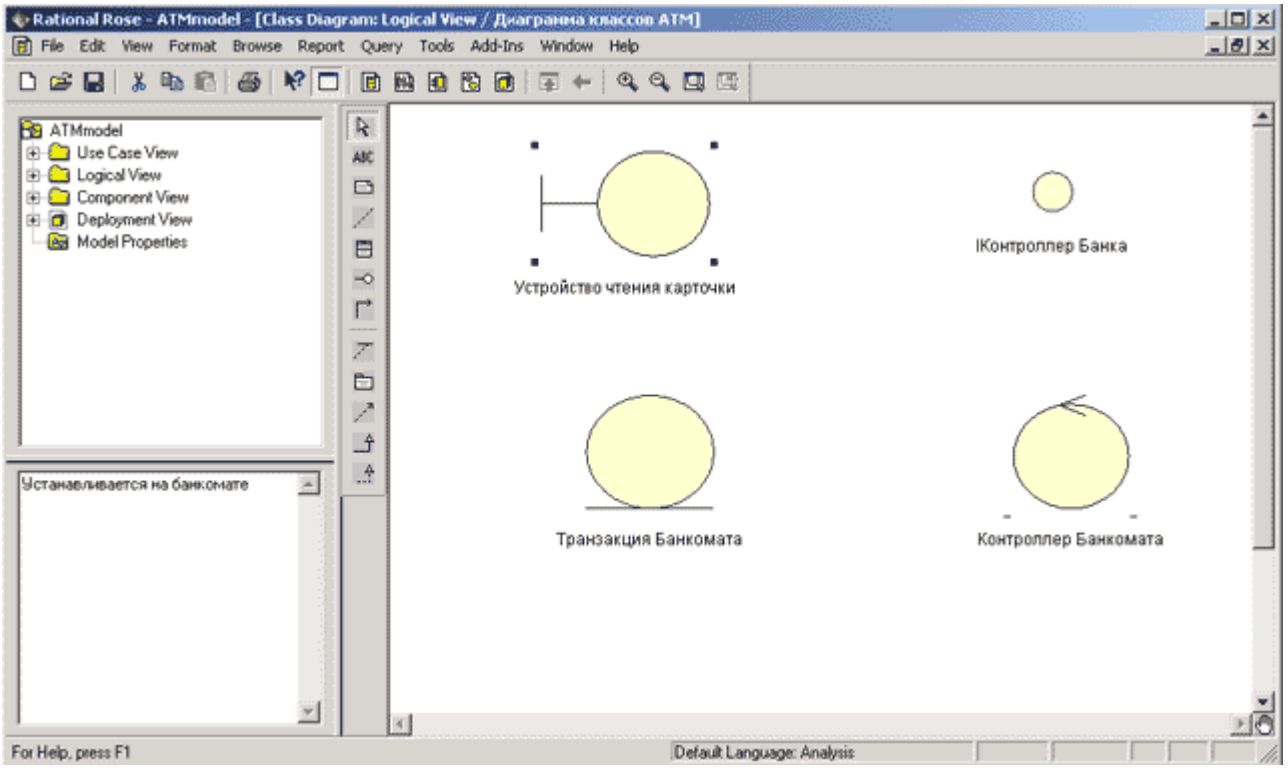



**Рис. 4.7.** Фрагмент диаграммы классов модели банкомата после добавления на нее классов Устройство чтения карточки и Контролер Банкомата

Практические действия по добавлению атрибутов и операций *классов*, редактированию их свойств, по добавлению отношений в процессе продолжения разработки диаграммы *классов* для модели банкомата, а также окончательный вид диаграммы *классов* для разрабатываемой модели банкомата будут рассмотрены в следующих лекциях.

Лабораторная работа 16. Разработка диаграммы кооперации и редактирование свойств ее элементов.

*Диаграмма кооперации* является разновидностью *диаграммы взаимодействия*, и в контексте языка *UML* описывает динамический аспект взаимодействия объектов при реализации отдельных вариантов использования. Общие рекомендации *по* построению диаграммы *кооперации* были рассмотрены в лекции 7 курса "Основы *объектно-ориентированного моделирования* в нотации *UML*". Активизировать рабочее окно диаграммы *кооперации* в программе *IBM* Rational Rose 2003 можно несколькими способами:

- Щелкнуть на кнопке с изображением *диаграммы взаимодействия* на *стандартной панели инструментов* и выбрать для построения новую диаграмму *кооперации*.
- Выполнить операцию главного меню: **Browse** → **Interaction Diagram** (Браузер → Диаграмма взаимодействия) и выбрать для построения новую диаграмму *кооперации*.
- Выполнить операцию контекстного меню: **New** → **Collaboration Diagram** (Новая → Диаграмма *кооперации*) для логического представления или представления вариантов использования в браузере проекта.

При этом появляется новое окно с чистым *рабочим листом* диаграммы *кооперации* и специальная *панель инструментов*, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы *кооперации* (табл. 7.1). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 7.1. Назначение кнопок специальной панели инструментов диаграммы *кооперации*

| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
|---|---|---|
|  | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
|  | *Text Box* | Добавляет на диаграмму текстовую область |

| | | |
|---|---|---|
|  | Note | Добавляет на диаграмму примечание |
|  | Anchor Note to Item | Добавляет на диаграмму *связь* примечания с соответствующим графическим элементом диаграммы |
|  | Object | Добавляет на диаграмму *объект* |
|  | *Class Instance* | Добавляет на диаграмму экземпляр класса |
|  | Object Link | Добавляет на диаграмму *связь* |
|  | Link To Self | Добавляет на диаграмму *рефлексивную связь* |
|  | Link Message | Добавляет на *связь* диаграммы прямое *сообщение* |
|  | Reverse Link Message | Добавляет на *связь* диаграммы обратное *сообщение* |
|  | Data Token | Добавляет на *связь* диаграммы элемент прямого потока данных |
|  | Reverse Data Token | Добавляет на *связь* диаграммы элемент обратного потока данных |

На специальной панели инструментов *по* умолчанию присутствуют практически все кнопки с пиктограммами элементов, которые могут быть использованы для построения диаграммы. В данной лекции в качестве примера рассматривается процесс построения диаграммы *кооперации*, которая представляет собой реализацию варианта использования Снятие наличных по кредитной карточке применительно к разрабатываемому проекту системы управления банкоматом. В модели данная *диаграмма кооперации* соответствует этому варианту использования и может быть размещена в представлении вариантов использования ( **Use Case View** ). После активизации новой диаграммы *кооперации* одним из описанных выше способов следует в качестве имени данной диаграммы задать: Снятие наличных *по* кредитной карточке.

В общем случае работа с диаграммой *кооперации* состоит в добавлении объектов, *связей* и *сообщений*, а также редактировании их свойств. При этом изменения, вносимые в диаграмму *кооперации*, автоматически вносятся в *диаграмму последовательности*, что можно увидеть в любой момент, активизировав последнюю нажатием клавиши <F5>.

Добавление объекта на диаграмму кооперации и редактирование его свойств

Добавить *объект* на диаграмму *кооперации* можно стандартным образом с помощью соответствующей кнопки на специальной панели инструментов. Однако, в случае наличия построенной ранее *диаграммы классов*, более удобным представляется следующий способ. В браузере проекта выделить необходимый *класс* и, удерживая нажатой левую кнопку мыши, перетащить изображение пиктограммы класса из браузера на свободное *место рабочего листа* диаграммы *кооперации*. В результате этих действий на диаграмме *кооперации* появится изображение *объекта* с именем класса и маркерами изменения его геометрических размеров (рис. 7.1).

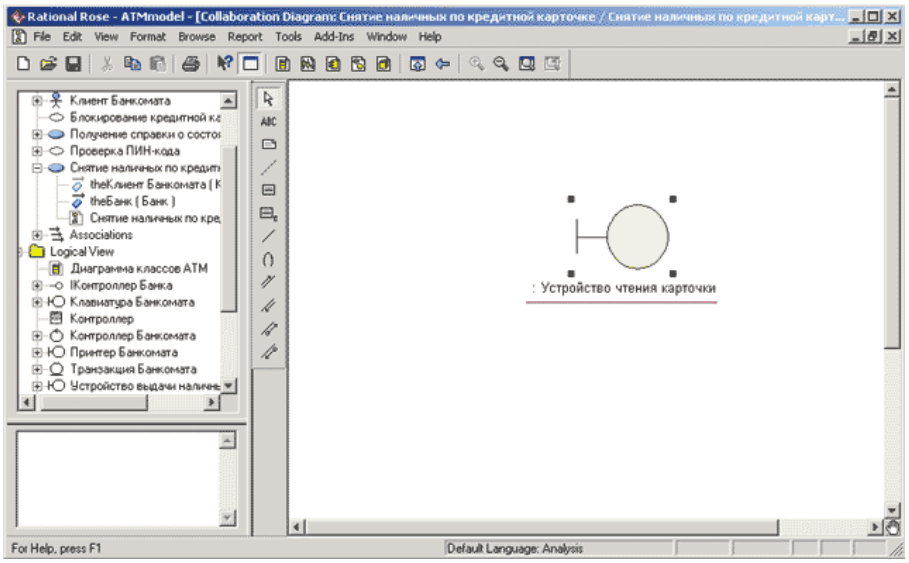



**Рис. 7.1.** Диаграмма кооперации после добавления на нее анонимного объекта класса Устройство чтения карточки

*По* умолчанию каждый добавляемый *объект* считается анонимным. При необходимости можно задать собственное имя *объекта*, для чего двойным щелчком на изображении *объекта* на диаграмме *кооперации* следует вызвать *диалоговое окно* свойств этого *объекта* (рис. 7.2).

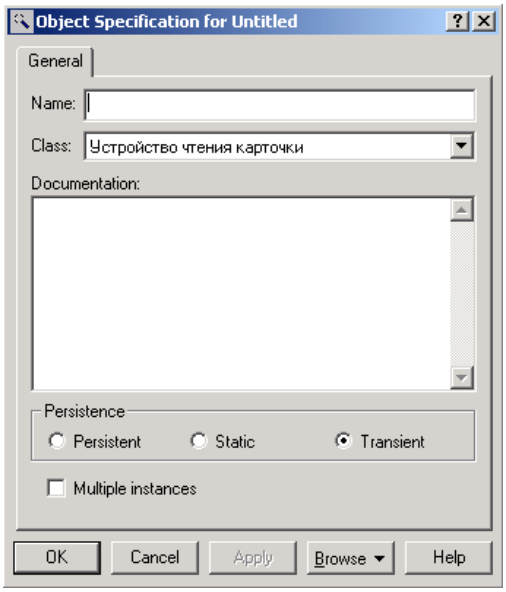



**Рис. 7.2.** Диалоговое окно спецификации свойств объекта класса Устройство чтения карточки

Как видно из рассмотрения этого окна свойств, для *объекта* выбранного класса можно задавать: собственное имя *объекта*, особенности его реализации и множественность экземпляров.

*Группа* свойств **Persistence** (*Устойчивость*) предназначена для спецификации устойчивости объектов соответствующего класса. При этом *свойство* **Persistent** (Устойчивый) означает, что *информация* об *объектах* данного класса должна быть сохранена в системе некоторым подходящим способом. Свойство **Static** (Статический) означает, что соответствующий *объект* сохраняется в памяти компьютера в течение всего времени работы программного приложения. Свойство **Transient** (Временный) соответствующий *объект* хранится в памяти компьютера в течение короткого времени, необходимого только для выполнения его операций. Применительно к рассматриваемой для *объекта* класса Устройство чтения карточки модели следует выбрать свойство **Persistent**.

При необходимости можно представить *объект* в форме *мультиобъекта*. Для этого следует выбрать отметку у *свойства* **Multiple instances** (Несколько экземпляров). Однако для *объекта* класса Устройство чтения карточки это свойство следует оставить пустым, поскольку данный *объект* присутствует в модели в единственном экземпляре.

Добавление связи и редактирование ее свойств

Для добавления *связи* между предварительно размещенными на диаграмме *объектами* нужно с помощью левой кнопки мыши нажать кнопку с изображением *связи* на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении одного *объекта* на диаграмме и отпустить ее на изображении другого *объекта*. В результате этих действий на диаграмме появится изображение *связи*, например, соединяющей *объект* класса Клиент Банкомата (актера) с объектом класса Устройство чтения карточки (рис. 7.3). Поскольку кнопка с изображением актера отсутствует на специальной панели инструментов диаграммы *кооперации*, соответствующий *объект* следует предварительно поместить на диаграмму способом перетаскивания пиктограммы актера из браузера проекта.

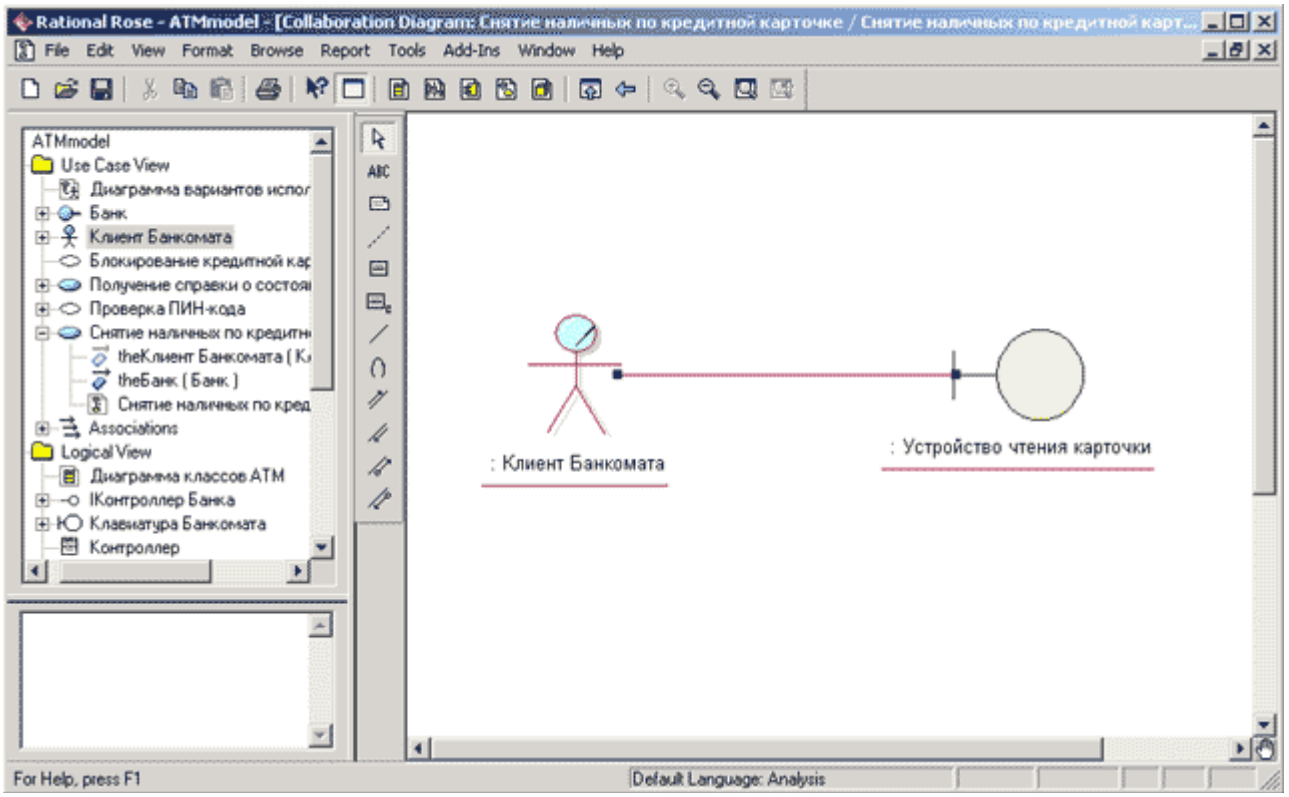



**Рис. 7.3.** Диаграмма кооперации после добавления связи между объектом класса Клиент Банкомата (актером) и объектом класса Устройство чтения карточки

*По* умолчанию каждая добавляемая *связь* считается анонимной. При необходимости можно задать имя *связи* с помощью диалогового окна спецификации свойств данной *связи* (рис. 7.4).

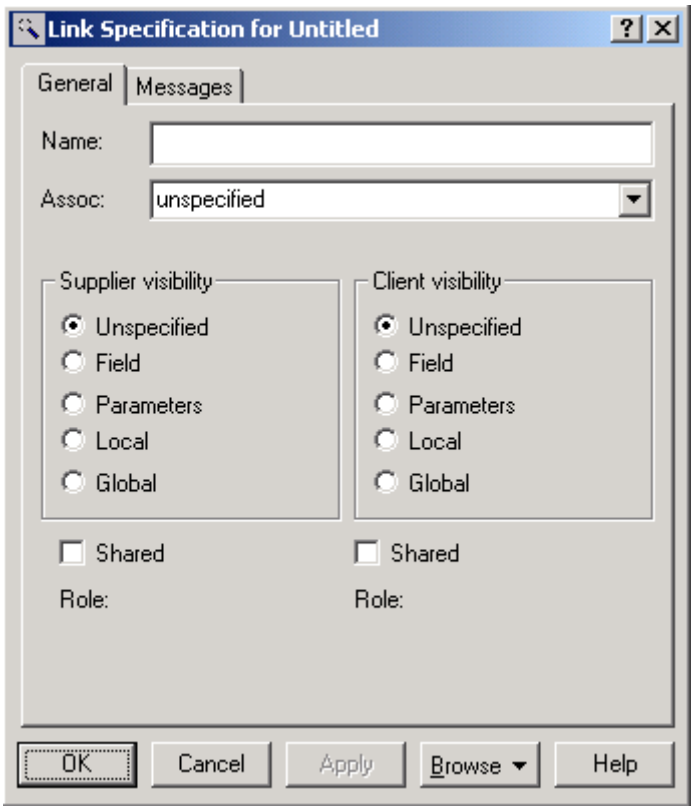


**Рис. 7.4.** Диалоговое окно редактирования свойств связи

Кроме имени *связи* можно также задать: *имя ассоциации*, видимость соответствующей пары объектов и наличие общих ролей. Однако более важной представляется следующая вкладка **Messages** ( *сообщения* ), служащая для спецификации *сообщений*, передаваемых между соответствующей парой объектов.

Добавление сообщения и редактирование его свойств

Добавить *сообщения* на диаграмму *кооперации* можно несколькими способами. Стандартный способ заключается в использовании кнопки с пиктограммой *сообщения* на специальной панели инструментов. В этом случае необходимо левой кнопкой мыши нажать кнопку с изображением прямого или обратного *сообщения* на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении линии *связи* на диаграмме и отпустить ее. В результате этих действий на диаграмме рядом с линией *связи* появится изображение стрелки *сообщения*.

Однако более удобным представляется способ добавления *сообщений* с помощью диалогового окна свойств *связей*. Для этого двойным щелчком на линии *связи* вызывается окно ее свойств и раскрывается вкладка **Messages** ( *сообщения* ). После этого следует выполнить операцию контекстного *меню* **Insert To** (Вставить в направлении), в результате чего появляется вложенный *список* с предложением выбрать одну из операций целевого класса для спецификации имени *сообщения* (рис. 7.5).

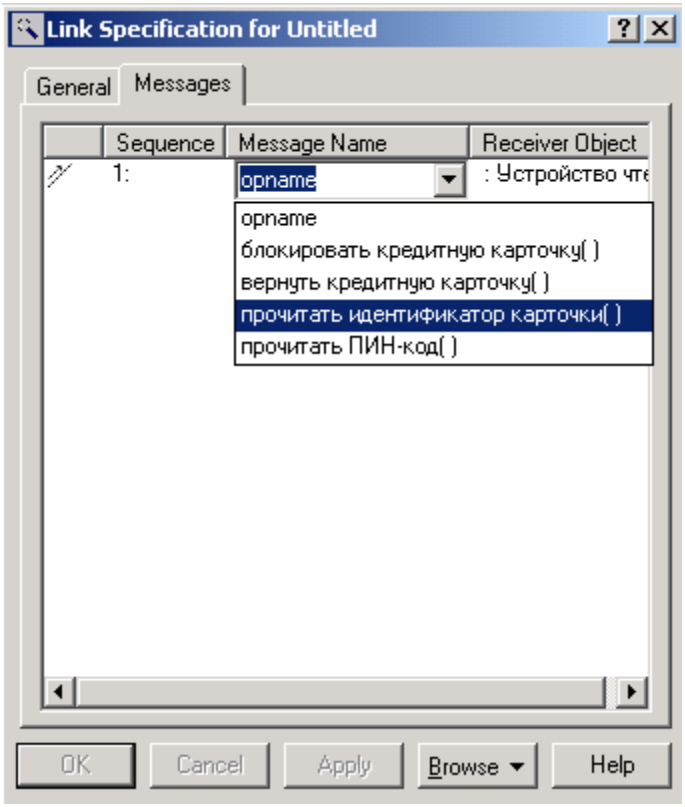



**Рис. 7.5.** Диалоговое окно добавления сообщения для выбранной связи

Для рассматриваемой модели банкомата для первого *сообщения* следует выбрать операцию прочитать идентификатор карточки(). После выбора *операции* для данного *сообщения* оно добавляется в *список сообщений* данной *связи*, а рядом с линией *связи* на диаграмме *кооперации* появится стрелка с номером и именем этого *сообщения* (рис. 7.6).

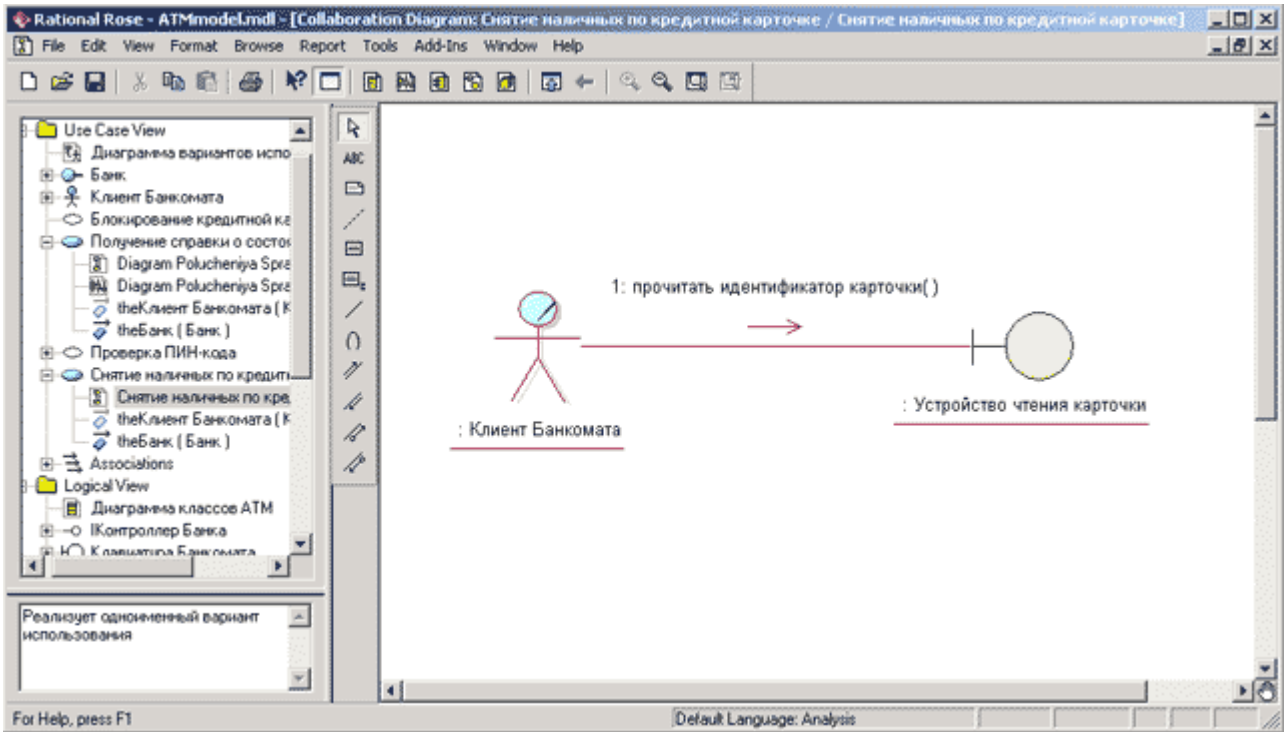



**Рис. 7.6.** Диаграмма кооперации после добавления связи между объектом класса Клиент Банкомата (актером) и объектом класса Устройство чтения карточки

Кроме имени *сообщения* можно также задать *стереотип* синхронизации и частоту передачи. Для этой цели следует воспользоваться диалоговым окном спецификации свойств *сообщений* (рис. 7.7), которое можно открыть двойным щелчком на имени *сообщения* в списке рассматриваемой вкладки **Messages** окна спецификации свойств *связи*.

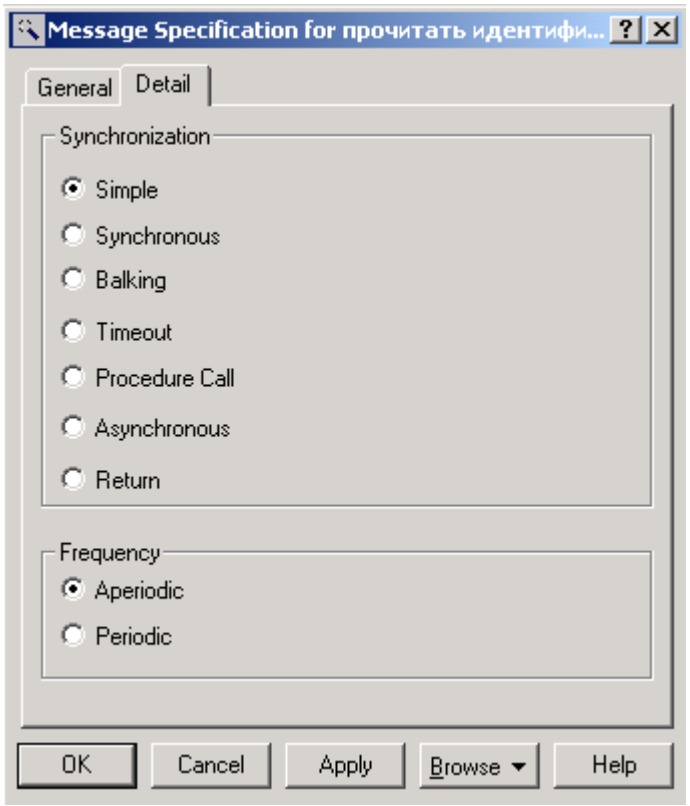



**Рис. 7.7.** Диалоговое окно спецификации свойств сообщения

*Группа* свойств **Synchronization** (Синхронизация) предназначена для определения способа синхронизации передаваемого *сообщения*. При изменении этого *свойства* изменяется графическое изображение стрелки соответствующего *сообщения*. Характеристика отдельных свойств синхронизации *сообщений* и графическое изображение соответствующих стрелок *сообщений* приводится в следующей таблице (табл. 7.2).

Таблица 7.2. Характеристика свойств синхронизации сообщений

| Название свойства | Графическое изображение стрелки | Назначение свойства |
|---|---|---|
| Simple (Простое) |  | Данное *сообщение* выполняется в одном *потоке управления*. Это *свойство* задается добавляемому на диаграмму *сообщению* по умолчанию |
| Synchronous (Синхронное) |  | После передачи данного *сообщения* клиент ожидает ответа от объекта-приемника о результате выполнения соответствующей операции |
| Balking (С отказом) |  | После передачи данного *сообщения* объект-приемник отказывает клиенту в выполнении соответствующей операции, если он занят выполнением других операций |
| *Timeout* (С ожиданием) |  | После передачи данного *сообщения* объект-приемник может поместить данное *сообщение* в очередь с ограниченным временем ожидания, если он занят выполнением других операций |
| *Procedure Call* (Вызов процедуры) |  | Клиент посылает данное *сообщение* объекту-приемнику и, чтобы продолжить свою работу ожидает, пока вся дальнейшая вложенная последовательность *сообщений* не будет обработана приемником |
| *Asynchronous* (Асинхронное) |  | Клиент посылает данное *сообщение* и продолжает свою работу, не ожидая подтверждения от объекта-приемника о получении этого *сообщения*. При этом соответствующая операция может быть как выполнена, так и не выполнена |
| Return (Возврат) |  | Данное *сообщение* посылается клиенту после окончания выполнения вызова процедуры |

*Группа* свойств **Frequency** (Частота) предназначена для указания периодического характера передачи *сообщения*. При изменении этого *свойства* графическое изображение стрелки соответствующего *сообщения* не изменяется. *Свойство* **Aperiodic** (Апериодическое) означает, что *сообщение* посылается клиентом нерегулярно. При этом *сообщение* может быть отправлено один или несколько раз через различные промежутки времени. Это *свойство* задается для *сообщения по* умолчанию. *Свойство* **Periodic** (Периодическое) означает, что *сообщение* регулярно посылается клиентом через определенные промежутки времени.

Применительно для модели банкомата можно оставить рассмотренные *свойства сообщений* без изменения, в том виде, в каком они определены *по* умолчанию программой *IBM* Rational Rose 2003.

Лабораторная работа 17. Разработка диаграммы последовательности.

Особенности разработки диаграммы последовательности в среде IBM Rational Rose

*Диаграмма последовательности* является другой формой визуализации взаимодействия в модели и, как и *диаграмма кооперации*, оперирует *объектами* и *сообщениями*. Общие рекомендации по построению *диаграммы последовательности* были рассмотрены в лекции 8 курса "Основы *объектно-ориентированного моделирования* в нотации *UML*". Особенность работы в среде *IBM* Rational Rose 2003 заключается в том, что этот вид *канонической диаграммы* может быть создан автоматически после построения диаграммы *кооперации* и нажатия клавиши <F5>. С помощью этой же клавиши осуществляется переключение между диаграммами последовательности и *кооперации* в модели.

Однако в отдельных случаях бывает удобно начать построение *диаграмм взаимодействия* с *диаграммы последовательности*. В этом случае активизировать рабочее окно *диаграммы последовательности* можно несколькими способами:

- Щелкнуть на кнопке с изображением диаграммы взаимодействия на *стандартной панели инструментов* и выбрать для построения диаграмму последовательности.

- Выполнить операцию главного меню: **Browse** → **Interaction Diagram** (Браузер → Диаграмма взаимодействия) и выбрать для построения новую диаграмму последовательности.
- Выполнить операцию контекстного меню: **New** → **Sequence Diagram** (Новая → *Диаграмма последовательности*) для логического представления или представления вариантов использования в браузере проекта.

При этом появляется новое окно с чистым *рабочим листом диаграммы классов* и специальная *панель инструментов*, содержащая кнопки с изображением графических примитивов, необходимых для разработки *диаграммы последовательности* (табл. 8.1). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 8.1. Назначение кнопок специальной панели инструментов диаграммы последовательности

| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
|---|---|---|
|  | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
|  | *Text Box* | Добавляет на диаграмму текстовую область |
|  | Note | Добавляет на диаграмму примечание |
|  | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
|  | Object | Добавляет на диаграмму *объект* |
|  | Object Message | Добавляет на диаграмму простое *сообщение* |
|  | Message To Self | Добавляет на диаграмму *рефлексивное сообщение* |
|  | Return Message | Добавляет на диаграмму *сообщение* типа возврата из вызова процедуры |
|  | *Destruction Marker* | Добавляет на диаграмму символ уничтожения *объекта* |
|  | *Procedure Call* | Добавляет на диаграмму *сообщение* типа вызова процедуры (по умолчанию отсутствует) |
|  | *Asynchronous Message* | Добавляет на диаграмму асинхронное *сообщение* (по умолчанию отсутствует) |

На специальной панели инструментов по умолчанию присутствует практически все пиктограммы элементов, которые могут быть использованы для построения *диаграммы последовательности*. Из дополнительных пиктограмм графических элементов на специальную *панель инструментов* можно добавить лишь *сообщение* типа вызова процедуры и асинхронное *сообщение* (последняя строка табл. 8.1). Относительно изображения асинхронного *сообщения* в форме полустрелки следует заметить, что хотя в версии языка *UML* 1.5 этот элемент отсутствует, в среде *IBM* Rational Rose 2003 возможно изобразить этот тип *сообщений* в форме специального графического *стереотипа*.

Лабораторная работа 18. Разработка диаграммы состояний и редактирование свойств ее элементов.

Особенности разработки диаграммы состояний в среде IBM Rational Rose 2003

Переходя к рассмотрению диаграммы *состояний*, следует отметить, что в среде *IBM* Rational Rose 2003 этот тип диаграмм может относиться к отдельному классу, *операции* класса, варианту использования, пакету или представлению. Общие рекомендации *по* построению диаграммы *состояний* были рассмотрены в лекциях 9 и 10

курса "Основы *объектно-ориентированного моделирования* в нотации *UML*". Для того чтобы построить диаграмму *состояний*, ее вначале необходимо создать и активизировать.

Начать построение диаграммы *состояний* для выбранного элемента модели или моделируемой системы в целом можно одним из следующих способов:

- Щелкнуть на кнопке с изображением диаграммы *состояний* на *стандартной панели инструментов*, после чего следует выбрать представление и тип разрабатываемой диаграммы - новая диаграмма *состояний*.
- Выделить логическое представление ( **Logical View** ) или представление вариантов использования ( **Use Case View** ) в браузере проекта и выполнить операцию контекстного меню: **New** → **Statechart Diagram** (Новая → Диаграмма *состояний* ).
- Раскрыть логическое представление ( **Logical View** ) в браузере проекта и выделить рассматриваемый класс, *операцию класса*, пакет, или раскрыть представление вариантов использования ( **Use Case View** ) и выбрать вариант использования, после чего выполнить операцию контекстного меню: **New** → **Statechart Diagram** (Новая → Диаграмма *состояний* ).
- Выполнить операцию главного меню: **Browse** → **State Machine Diagram** (Обзор → Диаграмма *состояний* ), после чего следует выбрать представление и тип разрабатываемой диаграммы.

В результате выполнения этих действий появляется новое окно с чистым *рабочим листом* диаграммы *состояний* и специальная *панель инструментов*, содержащая кнопки с изображением графических элементов модели, необходимых для разработки диаграммы *состояний* (табл. 9.1). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 9.1. Назначение кнопок специальной панели инструментов диаграммы состояний

| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
|---|---|---|
|  | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
|  | *Text Box* | Добавляет на диаграмму текстовую область |
|  | Note | Добавляет на диаграмму примечание |
|  | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
|  | State | Добавляет на диаграмму *состояние* |
|  | Start State | Добавляет на диаграмму начальное *состояние* |
|  | End State | Добавляет на диаграмму конечное *состояние* |
|  | State Transition | Добавляет на диаграмму *переход* |
|  | Transition to Self | Добавляет на диаграмму *рефлексивный переход* |
|  | *Horizontal Synchronization* | Добавляет на диаграмму горизонтально расположенный символ синхронизации (по умолчанию отсутствует) |
|  | *Vertical Synchronization* | Добавляет на диаграмму вертикально расположенный символ синхронизации (по умолчанию отсутствует) |
|  | Decision | Добавляет на диаграмму символ принятия решения для альтернативных *переходов* (по умолчанию отсутствует) |

*По* умолчанию на специальной панели инструментов могут отсутствовать кнопки с тремя последними графическими элементами из таблицы 9.1. При необходимости их можно добавить на специальную панель

диаграммы *состояний* аналогично способу, рассмотренному ранее в лекции 3 (рис. 3.1). Продолжая разработку проекта *по* моделированию системы управления банкоматом, можно приступить к разработке новой диаграммы *состояний*. С этой целью для диаграммы *состояний* модели банкомата зададим имя *Диаграмма состояний ATM*, а в секцию ее документации введем текст "*Диаграмма состояний* описывает *конечный автомат* банкомата".

Добавление состояния на диаграмму состояний и редактирование его свойств

Для добавления *состояния* на диаграмму *состояний* необходимо с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *состояния* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение *состояния* с маркерами изменения его геометрических размеров и предложенным средой именем *по* умолчанию, которое разработчику следует изменить.

Для диаграммы *состояний* модели банкомата в качестве имени первого добавленного *состояния* изменим предложенное программой *по* умолчанию имя NewState на Ожидание карточки (рис. 9.1). Задать имя *состояния* можно либо непосредственно при добавлении нового *состояния* на диаграмму *состояний*, либо открыв окно спецификации свойств нового *состояния*.

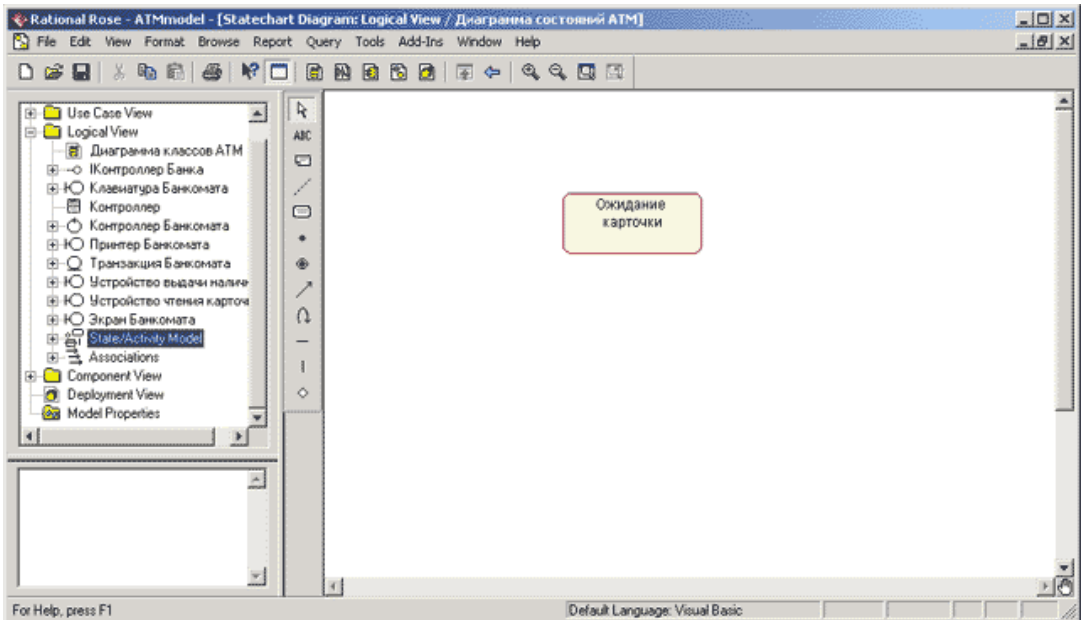



**Рис. 9.1.** Диаграмма состояний после добавления на нее состояния Ожидание карточки

Для добавленного *состояния* можно открыть *диалоговое окно* его свойств двойным щелчком левой кнопкой мыши на изображении этого элемента на диаграмме. В этом случае активизируется *диалоговое окно* со специальными вкладками, в поля которых можно занести всю информацию *по* данному *состоянию* (рис. 9.2).

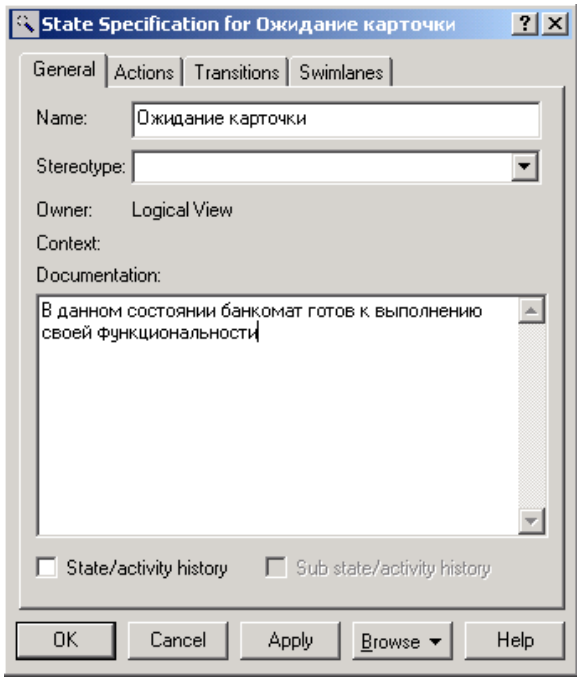



**Рис. 9.2.** Диалоговое окно спецификации свойств состояния

При необходимости в диалоговом окне спецификации свойств выбранного *состояния* можно задать вложенное *историческое состояние*. Для этого следует выставить отметку у свойства **State/activity history** ( *Историческое состояние /деятельность*) и нажать кнопку **Apply**. В результате внутри исходного *состояния* появится вложенное *историческое состояние* (рис. 9.3, а).


**Рис. 9.3.** Добавление вложенного исторического состояния (а) и состояния глубокой истории (б) для состояния Ожидание карточки

Чтобы обычное *историческое состояние* превратить в *состояние* глубокой истории, следует дополнительно выставить отметку у свойства **Sub state/activity history** (Историческое под-состояние/*деятельность*), которое становится доступным для редактирования после выбора первого свойства, и нажать кнопку **Apply**. В результате внутри исходного *состояния* появится вложенное *состояние* глубокой истории (рис. 9.3, б).

Чтобы обычное *состояние* превратить в *композит*, следует при добавлении нового *состояния* поместить его внутри границы того *состояния*, которое необходимо сделать *композитным*. В результате внутри исходного *состояния* появится новое вложенное *состояние* с именем **NewState**, которое при перемещении композита в области диаграммы *состояний* всегда будет находиться внутри своего композита (рис. 9.4).

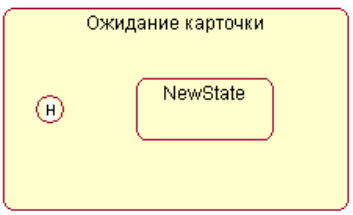



**Рис. 9.4.** Превращение состояния Ожидание карточки в композитное состояние

Рассмотренные выше действия приведены только с целью иллюстрации особенностей спецификации исторических и вложенных подсостояний и не относятся к разрабатываемой модели банкомата.

Дополнительно можно определить следующие свойства *состояний*: задать текстовый стереотип *состояния*, определить внутренние действия на входе и выходе, а также внутреннюю *деятельность*. Эти свойства доступны для редактирования на вкладке **General** (Общие) и **Actions** (Действия). На вкладке **Transitions** ( *Переходы* ) можно определять и редактировать *переходы*, которые входят и выходят из рассматриваемого состояния. Последняя вкладка **Swimlanes** (Дорожки) служит для спецификации дорожек, которые, в контексте языка *UML*, определяются для диаграммы деятельности.

Добавление перехода и редактирование его свойств

Для добавления *перехода* между двумя *состояниями* нужно с помощью левой кнопки мыши нажать кнопку с изображением *перехода* на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходного состояния на диаграмме и отпустить ее на изображении целевого состояния. В результате этих действий на диаграмме появится изображение *перехода*, соединяющего два выбранных состояния. Продолжая разработку модели системы управления банкоматом, добавим на диаграмму *состояний* начальное *состояние* ( **Start State** ) и соединим его *переходом* с *состоянием* Ожидание карточки (рис. 9.5).

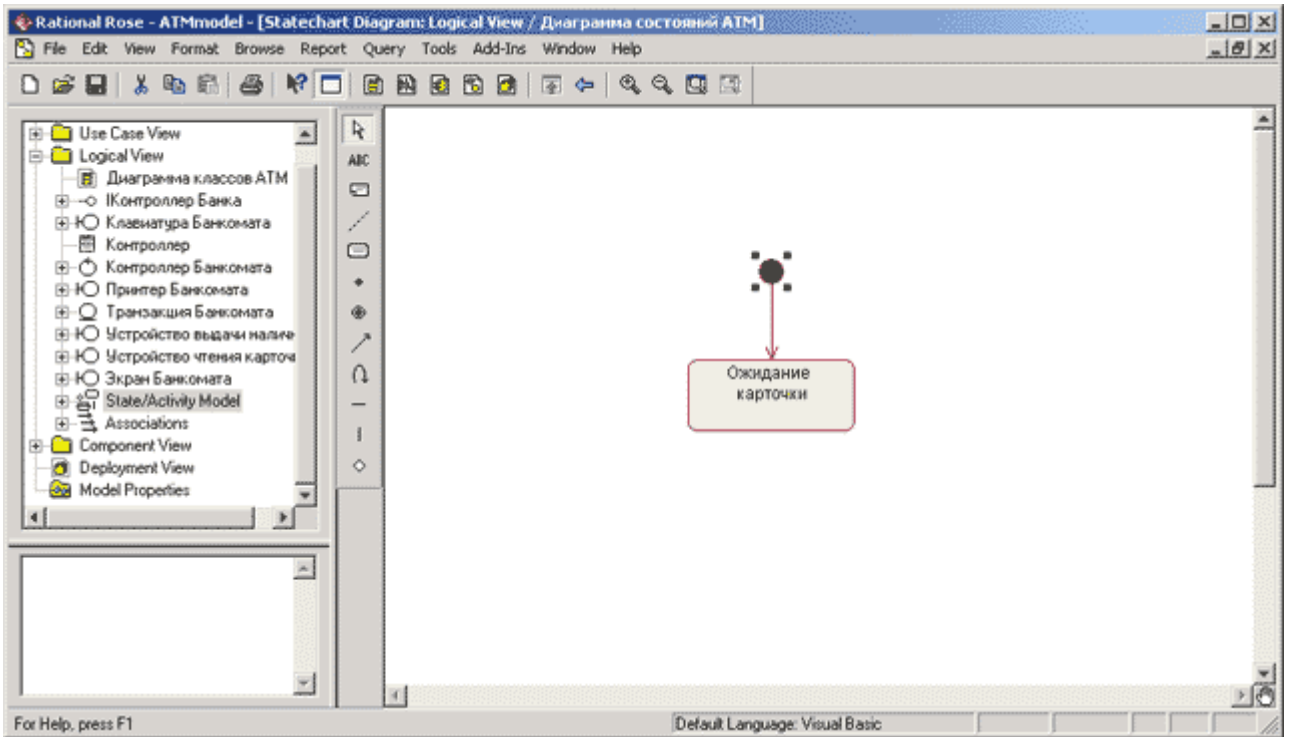



**Рис. 9.5.** Диаграмма состояний после добавления на нее перехода из начального состояния в состояние Ожидание карточки

После добавления *перехода* на диаграмму *состояний* можно открыть *диалоговое окно* его свойств и специфицировать дополнительные свойства, доступные на соответствующих вкладках (рис. 9.6). Следует обратить внимание на две первые строки вкладки **Detail** (Подробно), которые представляются наиболее важными из свойств *перехода*. Первое *поле ввода* **Guard Condition** служит для задания ***сторожевого условия***, которое определяет правило срабатывания соответствующего *перехода*. Во втором *поле ввода* **Action** можно специфицировать *действие, которое происходит при срабатывании перехода* до того, как моделируемая система попадет в целевое *состояние*.

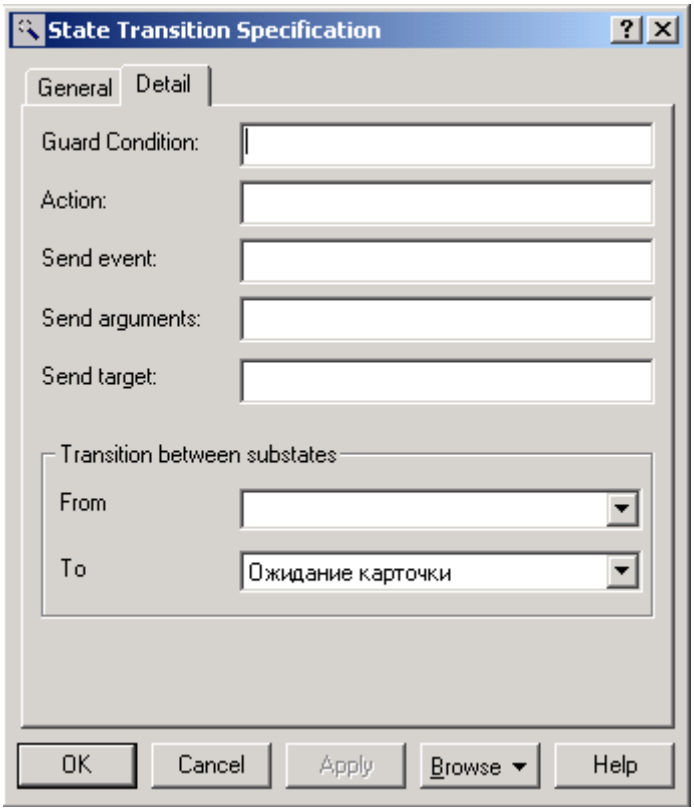



**Рис. 9.6.** Диалоговое окно спецификации свойств перехода, открытое на вкладке Detail (Подробно)

При необходимости можно определить сообщение о событии, происходящем при срабатывании *перехода*, а также визуализировать вложенность *состояний* и подключить историю отдельных *состояний*

Окончательное построение диаграммы состояний модели банкомата

Для завершения построения диаграммы *состояний* рассматриваемого примера следует описанным выше способом добавить оставшиеся *состояния* и *переходы*. С этой целью следует выполнить следующие действия:

1. Добавить *состояния* с именами: Ожидание ввода ПИН-кода, Проверка ПИН-кода, Ожидание выбора клиента, Обработка запроса на снятие наличных, Обработка запроса на получение справки, Выдача наличных, Печать, Возврат карточки, Завершение транзакции и финальное *состояние*.
2. Добавить *переход*: карточка вставлена, направленный от *состояния* Ожидание карточки к *состоянию* Ожидание ввода ПИН-кода.
3. Добавить *переход*: ПИН-код введен, направленный от *состояния* Ожидание ввода ПИН-кода к *состоянию* Проверка ПИН-кода.
4. Добавить *переход*: отмена транзакции, направленный от *состояния* Ожидание ввода ПИН-кода к *состоянию* Возврат карточки.
5. Добавить *переход* со *сторожевым условием*: [ПИН-код верный], направленный от *состояния* Проверка ПИН-кода к *состоянию* Ожидание выбора клиента.
6. Добавить *переход* со *сторожевым условием*: [ПИН-код неверный], направленный от *состояния* Проверка ПИН-кода к *состоянию* Ожидание ввода ПИН-кода.
7. Добавить *переход*: три неудачи с *действием на переходе* конфискация карточки, направленный от *состояния* Проверка ПИН-кода к *состоянию* Завершение транзакции. Для задания *действия на данном переходе* следует ввести текст конфискация карточки в поле ввода **Action** (Действие) на вкладке **Detail** (Подробно) окна спецификации свойств данного *перехода* (рис. 9.7).

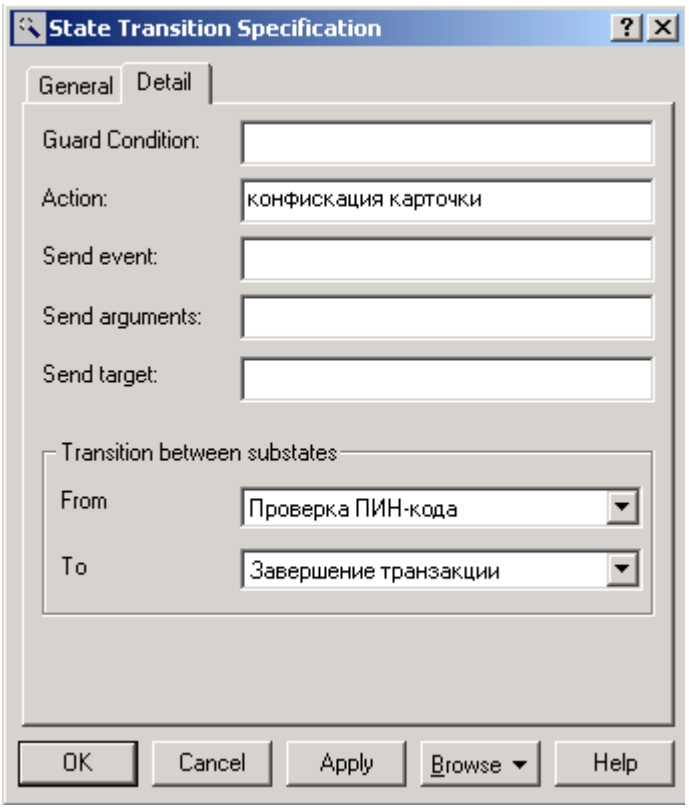



**Рис. 9.7.** Диалоговое окно спецификации свойств перехода три неудачи при задании действия на переходе

Для продолжения построения диаграммы *состояний* следует выполнить следующие действия:

8. Добавить *переход*: выбор суммы со *сторожевым условием*: [сумма введена], направленный от *состояния* Ожидание выбора клиента к *состоянию* Обработка запроса на снятие наличных.
9. Добавить *переход*: выбор справки, направленный от *состояния* Ожидание выбора клиента к *состоянию* Обработка запроса на получение справки.
10. Добавить *переход*: отмена транзакции, направленный от *состояния* Ожидание выбора клиента к *состоянию* Возврат карточки.
11. Добавить *переход* со *сторожевым условием*: [кредит не превышен], направленный от *состояния* Обработка запроса на снятие наличных к *состоянию* Выдача наличных.
12. Добавить *переход* со *сторожевым условием*: [кредит превышен] с *действием на переходе* сообщение, направленный от *состояния* Обработка запроса на снятие наличных к *состоянию* Возврат карточки.
13. Добавить *переход*: наличные выданы со *сторожевым условием*: [выбрана печать чека], направленный от *состояния* Выдача наличных к *состоянию* Печать.
14. Добавить *переход*: наличные выданы со *сторожевым условием*: [печать чека не выбрана], направленный от *состояния* Выдача наличных к *состоянию* Возврат карточки.
15. Добавить *переход*: справка сформирована, направленный от *состояния* Обработка запроса на получение справки к *состоянию* Печать.
16. Добавить *переход*: печать закончена, направленный от *состояния* Печать к *состоянию* Возврат карточки.
17. Добавить *переход*: карточка возвращена, направленный от *состояния* Возврат карточки к *состоянию* Завершение транзакции.
18. Добавить *переход*: транзакция завершена, направленный от *состояния* Завершение транзакции к *состоянию* Ожидание карточки.
19. Добавить *переход*, направленный от *состояния* Ожидание карточки к финальному *состоянию*.

*Диаграмма состояний* для рассматриваемой модели банкомата будет иметь следующий вид (рис. 9.8).

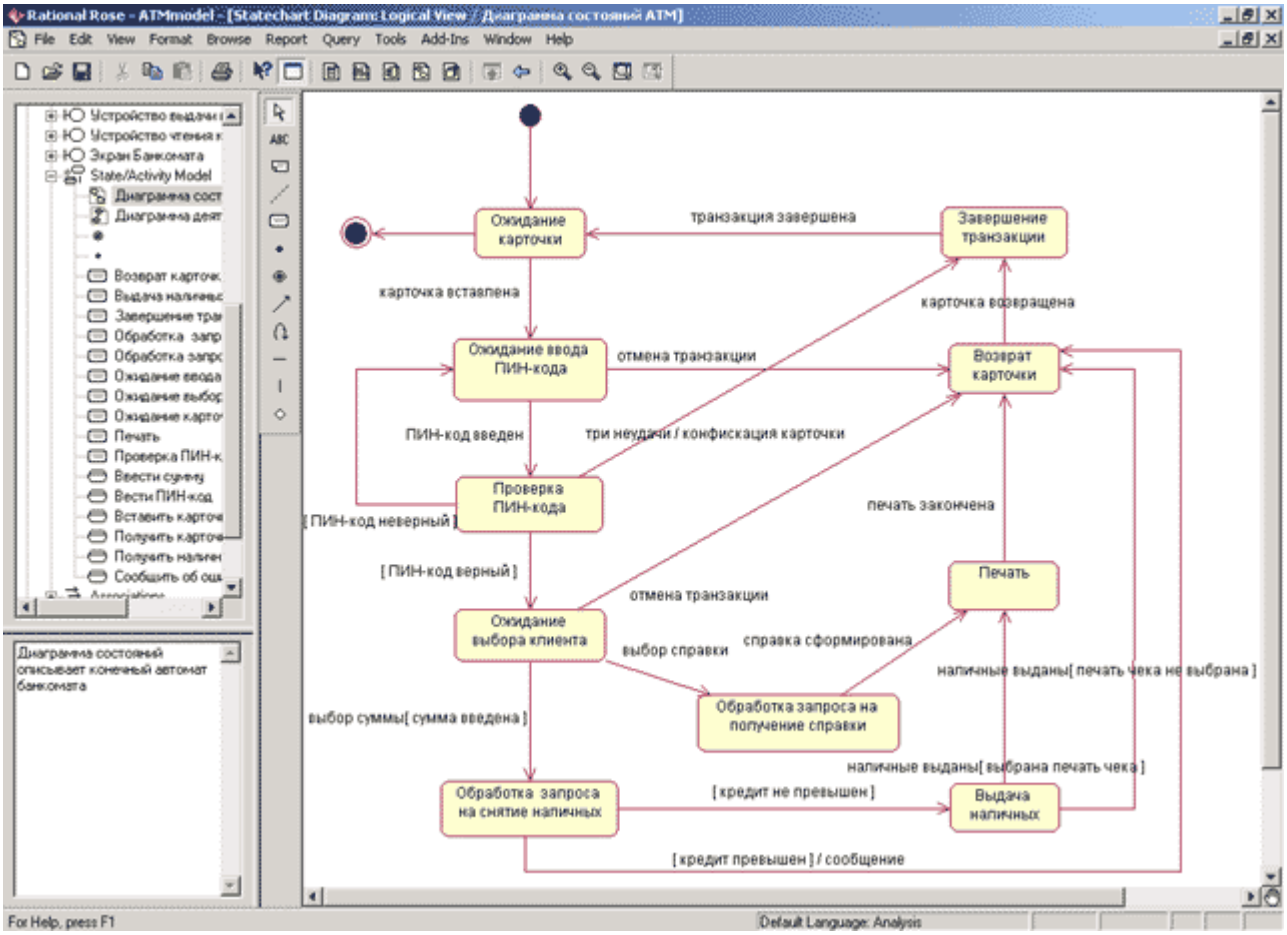



**Рис. 9.8.** Окончательный вид диаграммы состояний для моделирования поведения банкомата

Следует заметить, что в разрабатываемой модели *диаграмма состояний* является единственной и описывает поведение системы управления банкоматом в целом. Главное достоинство данной диаграммы *состояний* - возможность моделировать условный характер реализации всех вариантов использования в форме изменения отдельных *состояний* разрабатываемой системы. В то же время в среде *IBM* Rational Rose 2003 данная *диаграмма* не является необходимой для генерации программного кода. Поэтому в случае дублирования информации, представленной на диаграммах кооперации и последовательности, разработку диаграммы *состояний*, особенно в условиях дефицита времени, отпущенного на выполнение проекта, иногда опускают.

## Лабораторная работа 19. Разработка диаграммы деятельности и редактирование свойств ее элементов.

Особенности разработки диаграммы деятельности в среде IBM Rational Rose 2003

*Диаграмма деятельности* в среде *IBM* Rational Rose 2003, так же как и *диаграмма* состояний, может относиться к отдельному классу, *операции* класса, варианту использования, пакету или представлению. Общие рекомендации *по* построению диаграммы *деятельности* были рассмотрены в лекции 11 курса "Основы *объектно-ориентированного моделирования* в нотации *UML*". Для того чтобы построить диаграмму *деятельности*, ее вначале необходимо создать и активизировать.

Начать построение диаграммы *деятельности* для выбранного элемента модели или моделируемой системы в целом можно одним из следующих способов:

- Щелкнуть на кнопке с изображением *диаграммы состояний* на *стандартной панели инструментов*, после чего следует выбрать представление и тип разрабатываемой диаграммы - диаграмма *деятельности*.
- Выделить логическое представление ( **Logical View** ) или представление вариантов использования ( **Use Case View** ) в браузере проекта и выполнить операцию контекстного меню: **New** → **Activity Diagram** (Новая → Диаграмма *деятельности* ).
- Раскрыть логическое представление ( **Logical View** ) в браузере проекта и выделить рассматриваемый класс, *операцию класса*, пакет, или раскрыть представление вариантов использования ( **Use Case View** ) и выбрать

вариант использования, после чего выполнить операцию контекстного меню: **New** → **Activity Diagram** (Новая → Диаграмма *деятельности* ).

- Выполнить операцию главного меню: **Browse** → **State Machine Diagram** (Обзор → Диаграмма состояний), после следует чего выбрать представление и тип разрабатываемой диаграммы - диаграмма *деятельности*.

В результате выполнения этих действий появляется новое окно с чистым *рабочим листом* диаграммы *деятельности* и специальная *панель инструментов*, содержащая кнопки с изображением графических элементов, необходимых для разработки диаграммы *деятельности* (табл. 10.1). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 10.1. Назначение кнопок специальной панели инструментов диаграммы деятельности

| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
|---|---|---|
|  | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
|  | *Text Box* | Добавляет на диаграмму текстовую область |
|  | Note | Добавляет на диаграмму примечание |
|  | *Anchor* Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
|  | State | Добавляет на диаграмму состояние |
|  | Activity | Добавляет на диаграмму *деятельность* |
|  | Start State | Добавляет на диаграмму начальное состояние |
|  | End State | Добавляет на диаграмму конечное состояние |
|  | State Transition | Добавляет на диаграмму *переход* |
|  | Transition to Self | Добавляет на диаграмму *рефлексивный переход* |
|  | *Horizontal Synchronization* | Добавляет на диаграмму горизонтально расположенный символ синхронизации |
|  | *Vertical Synchronization* | Добавляет на диаграмму вертикально расположенный символ синхронизации |
|  | *Decision* | Добавляет на диаграмму *символ принятия решения* для альтернативных *переходов* |
|  | Swimlane | Добавляет на диаграмму дорожку |
|  | Object | Добавляет на *диаграмму объект* (по умолчанию отсутствует) |
|  | Object Flow | Добавляет на диаграмму стрелку *потока объектов* (по умолчанию отсутствует) |
|  | Business Activity | Добавляет на диаграмму бизнес-деятельность (по умолчанию отсутствует) |
|  | Business Transaction | Добавляет на диаграмму бизнес-транзакцию (по умолчанию отсутствует) |

Как видно из этой таблицы, *по* умолчанию на панели инструментов отсутствуют некоторые графические элементы, а именно - кнопки с пиктограммами объекта и *потока объектов*. При необходимости их можно добавить на специальную панель диаграммы *деятельности* стандартным способом, который был описан ранее.

Для разрабатываемого проекта системы управления банкоматом *диаграмма деятельности* описывает последовательность действий клиента при использовании банкомата. Для удобства можно включить эту диаграмму в логическое *представление*, для чего необходимо в браузере проекта выделить логическое *представление* ( **Logical View** ) и выполнить операцию контекстного *меню*: **New** → **Activity Diagram** (Новая → *Диаграмма деятельности* ). Продолжая разработку проекта *по* моделированию системы управления банкоматом, можно приступить к разработке новой диаграммы *деятельности*. С этой целью для диаграммы *деятельности* модели банкомата зададим имя Диаграмма *деятельности ATM*, а в секцию ее документации введем текст "*Диаграмма деятельности* описывает последовательность действий клиента при использовании банкомата".

Добавление деятельности на диаграмму деятельности и редактирование ее свойств

Для добавления *деятельности* на диаграмму *деятельности* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *деятельности* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте *рабочего листа* диаграммы. На диаграмме появится изображение *деятельности* с маркерами изменения его геометрических размеров и предложенным средой именем *по* умолчанию, которое разработчику следует изменить. Добавить *деятельность* на диаграмму можно также с помощью *операции* главного *меню*: **Tools** → **Create** → **Activity** или с помощью *операции* контекстного *меню*: **New** → **Activity**, предварительно выделив диаграмму *деятельности* в браузере проекта.

В результате этих действий на диаграмме появится изображение *деятельности* с именем **NewActivity**, предложенное программой *по* умолчанию. Начиная построение диаграммы *деятельности* модели банкомата, для первой добавленной *деятельности* зададим имя Вставить карточку (рис. 10.1).

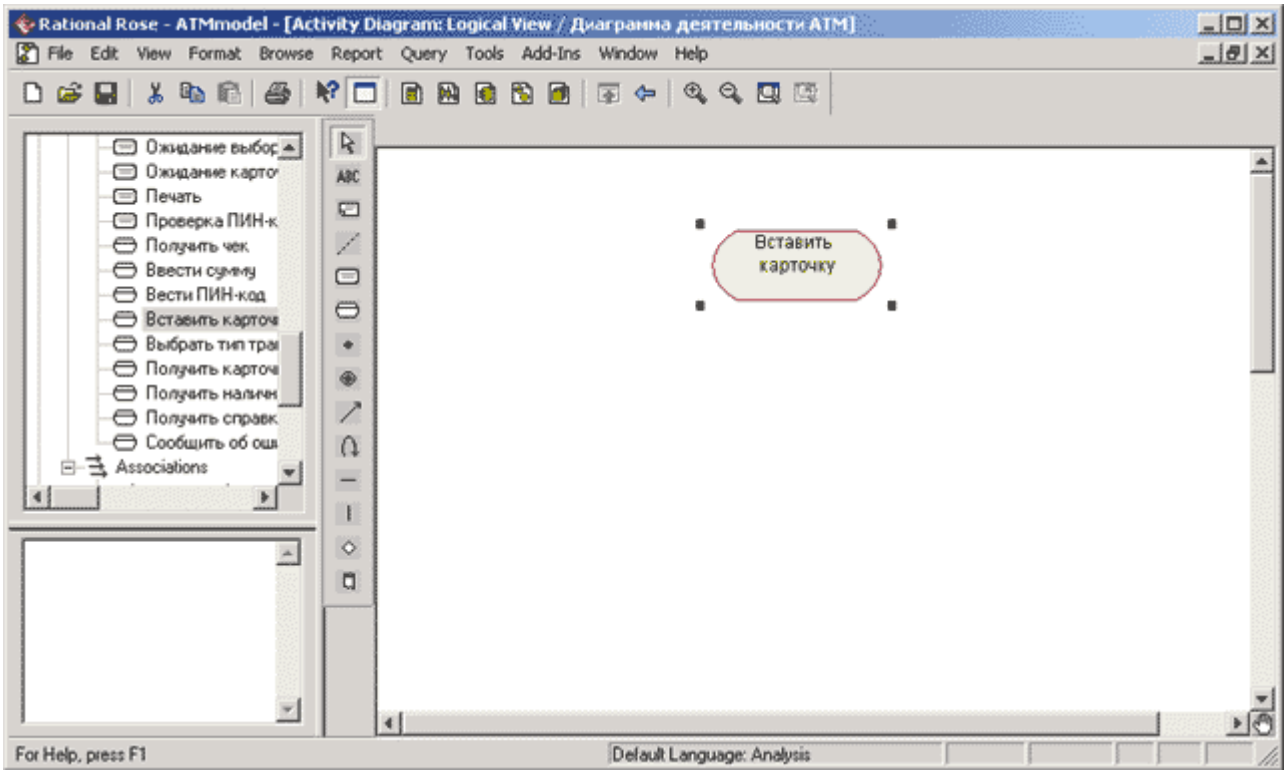



**Рис. 10.1.** Диаграмма деятельности после добавления на нее деятельности Вставить карточку

После добавления *деятельности* на диаграмму *деятельности* можно открыть *диалоговое окно* спецификации ее свойств и определить дополнительные свойства *деятельности*, доступные на соответствующих вкладках (рис. 10.2).

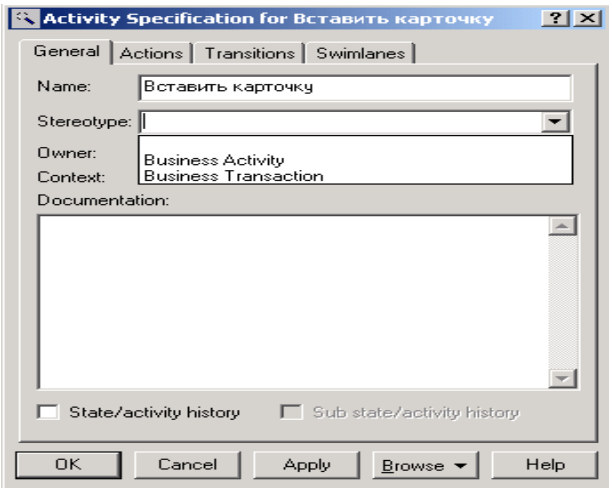



**Рис. 10.2.** Диалоговое окно спецификации свойств деятельности

При этом для *деятельности* становятся доступными для выбора два стереотипа: **Business Activity** (Бизнес-*деятельность*) и **Business Transaction** (Бизнес-транзакция), которые имеют собственное графическое изображение (см. табл. 10.1). На вкладке **Transitions** ( *Переходы* ) окна спецификации свойств *деятельности* можно определять и редактировать *переходы*, которые входят и выходят из рассматриваемой *деятельности*. Последняя вкладка **Swimlanes** (Дорожки) служит для спецификации дорожки, на которую помещается рассматриваемая *деятельность*.

Хотя *программа IBM* Rational Rose 2003 позволяет определить свойства *деятельности*, доступные на вкладке **Actions** (Действия), следует помнить, что внутренние действия являются свойствами общего понятия состояния, а внутренняя *деятельность* служит именем собственно *деятельности*, помещаемой на диаграмму *деятельности*. Поэтому для *деятельности* во избежание недоразумений лучше оставить эту вкладку пустой.

Добавление перехода и редактирование его свойств

Добавление *перехода* на диаграмму *деятельности* полностью аналогично диаграмме состояний. А именно, для добавления *перехода* между двумя *деятельностями* нужно с помощью левой кнопки мыши нажать кнопку с изображением *перехода* на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходной *деятельности* на диаграмме и отпустить ее на изображении целевой *деятельности*. В результате этих действий на диаграмме появится изображение *перехода*, соединяющего две выбранных *деятельности*. Если в качестве одной из деятельностей является символ ветвления или соединения, то порядок добавления *перехода* сохраняется прежним.

Следует заметить, что при наличии в проекте законченной *диаграммы состояний* попытка добавить начальное состояние на диаграмму *деятельности* с помощью кнопки специальной панели инструментов окажется безуспешной. В этом случае *программа IBM* Rational Rose 2003 фиксирует наличие в модели начального состояния и не позволит добавить его с помощью соответствующей кнопки на разрабатываемые *диаграммы состояний* или *деятельности*. Решить данную проблему можно посредством перетаскивания с помощью мыши начального состояния из браузера проекта на любую из вновь разрабатываемых диаграмм.

После добавления *перехода* на диаграмму *деятельности* становятся доступными для редактирования его свойства в специальном диалоговом окне (рис. 10.3), которое можно открыть *по* двойному щелчку левой кнопкой мыши на изображении *перехода*.

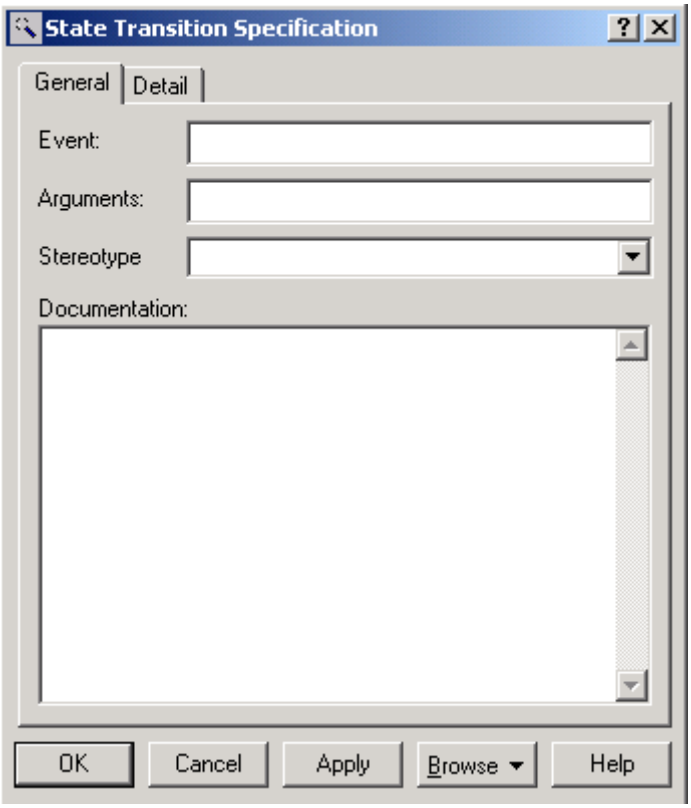



**Рис. 10.3.** Диалоговое окно спецификации свойств перехода

При спецификации свойств *переходов* следует помнить, что все *переходы* на диаграмме *деятельности* является нетриггерными, т.е. не имеют имен событий. *По* этой причине *поле ввода* с именем **Event** (Событие) для всех *переходов* должно оставаться пустым. Но все *переходы*, выходящие из *символов ветвления (решения)*, должны иметь *сторожевые условия*, которые специфицируются на вкладке **Detail** (Подробно) диалогового окна спецификации свойств *перехода*.

Окончательное построение диаграммы деятельности модели банкомата

Для завершения построения диаграммы *деятельности* рассматриваемого примера следует описанным выше способом добавить оставшиеся *деятельности* и *переходы*. С этой целью следует выполнить следующие действия:

1. Добавить *деятельности* с именами: Ввести ПИН-код, Выбрать тип транзакции, Ввести сумму, Получить справку о состоянии счета, Получить наличные, Получить чек, Получить карточку и финальное состояние.
2. Добавить *символы ветвления (решения)*, расположив их между *деятельностями* с именами: Ввести ПИН-код и Выбрать тип транзакции, Выбрать тип транзакции и Ввести сумму, Ввести сумму и Получить справку о состоянии счета, Получить наличные и Получить чек, Получить чек и Получить карточку. При этом последний *символ решения* будет использоваться в качестве символа соединения.
3. Добавить *переход*, направленный от *деятельности* Ввести ПИН-код к *символу решения*.
4. Добавить *переход* со *сторожевым условием*: [ПИН-код верный], направленный от *символа решения* к *деятельности* Выбрать тип транзакции. Для задания *сторожевого условия* данного *перехода* следует ввести текст ПИН-код верный в поле ввода **Guard Condition** ( *Сторожевое условие* ) на вкладке **Detail** (Подробно) окна спецификации свойств данного *перехода* (рис. 10.4). При этом текст *сторожевого условия* следует вводить без скобок.

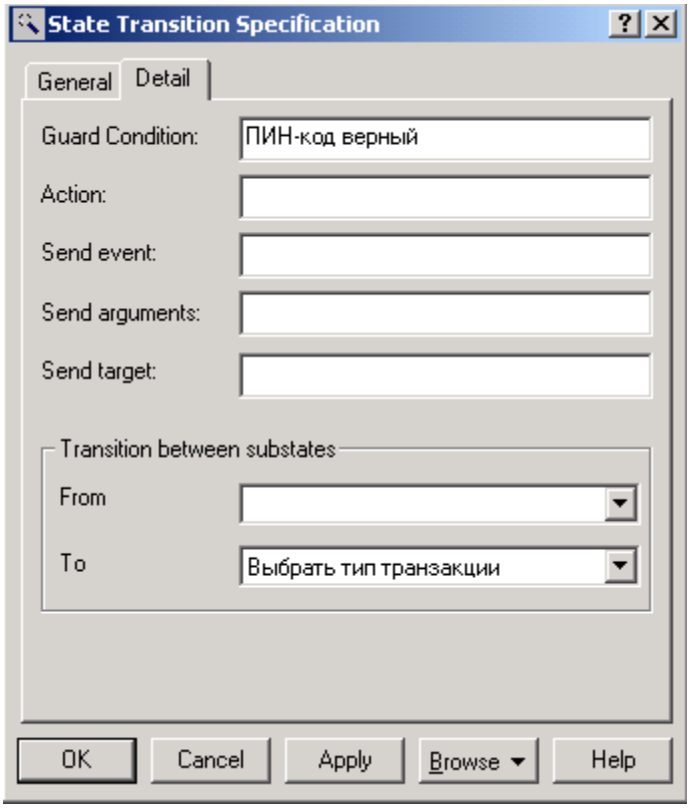



**Рис. 10.4.** Диалоговое окно спецификации свойств перехода при задании сторожевого условия

Для продолжения построения диаграммы *деятельности* следует выполнить следующие действия:

5. Добавить *переход* со *сторожевым условием*: [ПИН-код неверный], направленный от *символа решения* к символу соединения.
6. Добавить *переход*, направленный от *деятельности* Выбрать тип транзакции к *символу решения*.
7. Добавить *переход* со *сторожевым условием*: [выбор снятия суммы] , направленный от *символа решения* к *деятельности* Ввести сумму.
8. Добавить *переход* со *сторожевым условием*: [выбор получения справки], направленный от *символа решения* к *деятельности* Получить справку о состоянии счета.
9. Добавить *переход*, направленный от *деятельности* Ввести сумму к *символу решения*.
10. Добавить *переход* со *сторожевым условием*: [сумма не превышает *кредит*], направленный от *символа решения* к *деятельности* Получить наличные.
11. Добавить *переход* со *сторожевым условием*: [сумма превышает *кредит*], направленный от *символа решения* к символу соединения.
12. Добавить *переход*, направленный от *деятельности* Получить наличные к *символу решения*.
13. Добавить *переход* со *сторожевым условием*: [выбрана печать чека], направленный от *символа решения* к *деятельности* Получить чек.
14. Добавить *переход* со *сторожевым условием*: [печать чека не выбрана], направленный от *символа решения* к символу соединения.
15. Добавить *переход*, направленный от *деятельности* Получить чек к символу соединения.
16. Добавить *переход*, направленный от *деятельности* Получить справку о состоянии счета к символу соединения.
17. Добавить *переход*, направленный от символа соединения к *деятельности* Получить карточку.
18. Добавить *переход*, направленный от *деятельности* Получить карточку к финальному состоянию.

Построенная таким образом *диаграмма деятельности* будет иметь следующий вид (рис. 10.5).

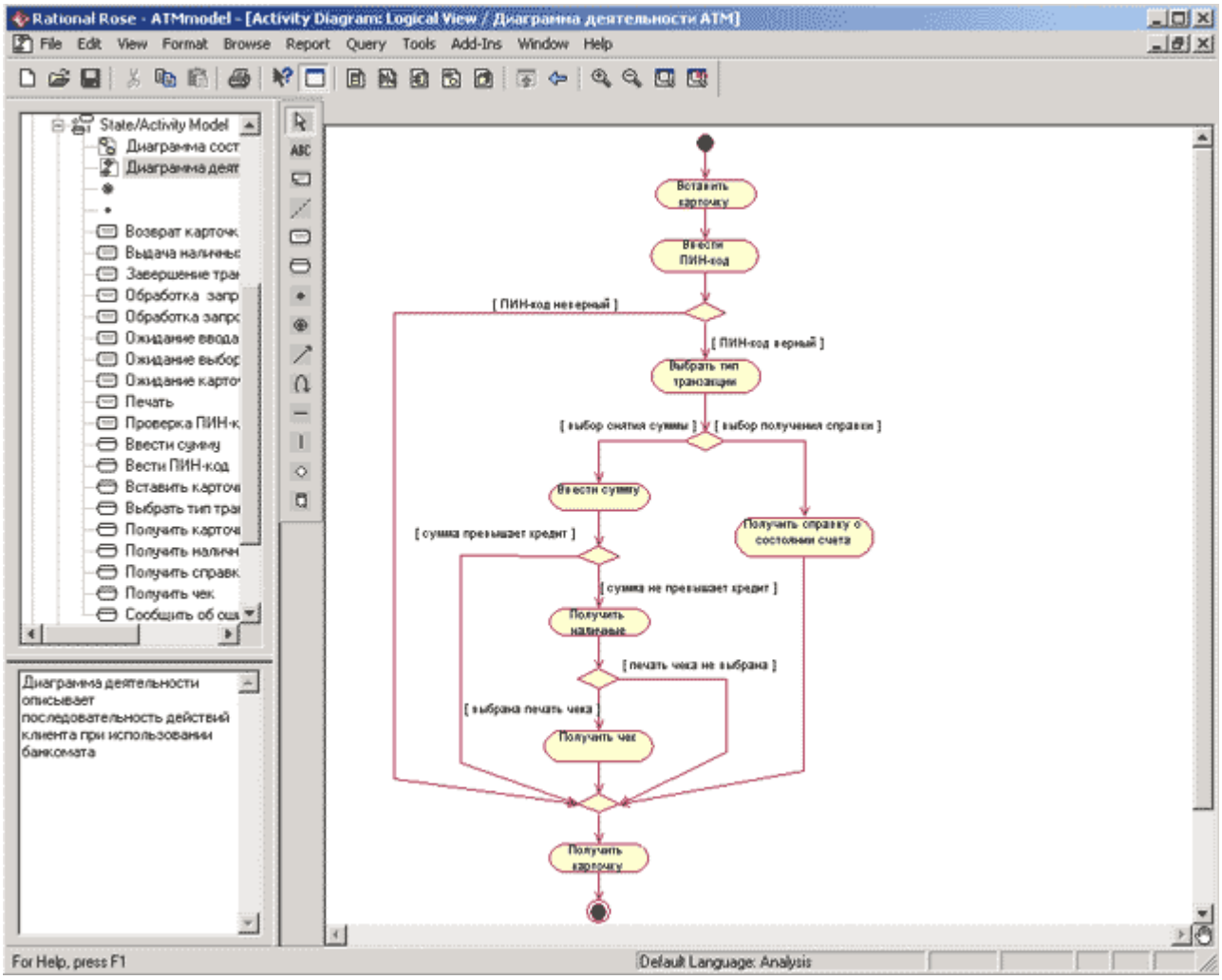



**Рис. 10.5.** Окончательный вид диаграммы деятельности для модели банкомата

Следует заметить, что в разрабатываемой модели *диаграмма деятельности* не описывает ситуацию блокирования карточки при трижды неверно введенном ПИН-коде. Дополнить данную диаграмму *деятельности*, которая учитывает данное условие в форме проверки отдельного условия, предлагается читателям самостоятельно в качестве упражнения.

Следует помнить, что в среде *IBM* Rational Rose 2003 *диаграмма деятельности* не является необходимой для генерации программного кода. Поэтому разработку диаграмм этого типа, особенно в условиях дефицита времени, отпущенного на выполнение проекта, иногда опускают. В то же время следует отметить, что в проектах реинжиниринга и документирования бизнес-процессов *диаграмма деятельности* является основным средством визуализации бизнес-процессов в контексте языка *UML*.

## Лабораторная работа 20. Разработка диаграммы компонентов и редактирование свойств ее элементов

Особенности разработки диаграммы компонентов в среде IBM Rational Rose 2003

*Диаграмма компонентов* служит частью физического представления модели, играет важную роль в процессе ООАП и является необходимой для генерации программного кода. Общие рекомендации *по* построению диаграммы *компонентов* были рассмотрены в лекции 12 курса "Основы *объектно-ориентированного моделирования* в нотации *UML*". Для разработки диаграмм *компонентов* в браузере проекта предназначено отдельное *представление компонентов* ( **Component View** ), в котором уже содержится *диаграмма компонентов* с пустым содержанием и именем *по* умолчанию **Main** (Главная).

Активизация диаграммы *компонентов* может быть выполнена одним из следующих способов:

- Щелкнуть на кнопке с изображением диаграммы *компонентов* на *стандартной панели инструментов*.
- Раскрыть представление *компонентов* в браузере ( **Component View** ) и дважды щелкнуть на пиктограмме **Main** (Главная).
- Через пункт меню **Browse** → **Component Diagram** (Браузер → Диаграмма *компонентов* ).

В результате выполнения этих действий появляется новое окно с чистым *рабочим листом* диаграммы *компонентов* и специальная *панель инструментов*, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы *компонентов* (табл. 12.1).

Таблица 12.1. Назначение кнопок специальной панели инструментов диаграммы компонентов

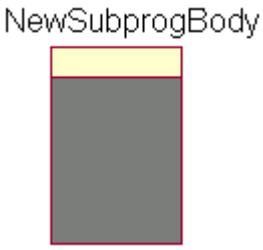
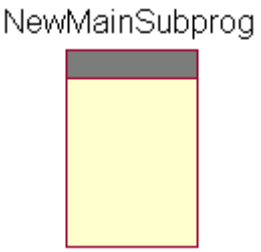
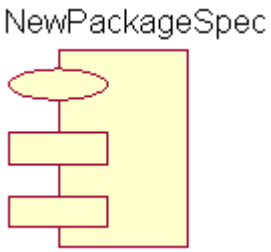
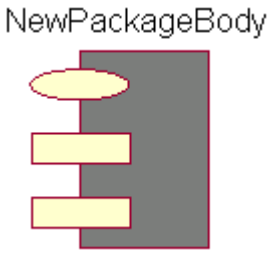
| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
|---|---|---|
|  | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
|  | *Text Box* | Добавляет на диаграмму текстовую область |
|  | Note | Добавляет на диаграмму примечание |
|  | *Anchor* Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
|  | Component | Добавляет на диаграмму *компонент* |
|  | Package | Добавляет на *диаграмму пакет* |
|  | *Dependency* | Добавляет на диаграмму отношение зависимости |
|  | *Subprogram* Specification | Добавляет на диаграмму спецификацию подпрограммы |
|  | *Subprogram* Body | Добавляет на диаграмму тело подпрограммы |
|  | Main Program | Добавляет на диаграмму главную программу |
|  | Package Specification | Добавляет на диаграмму *спецификацию пакета* |
|  | Package Body | Добавляет на диаграмму тело пакета |
|  | Task Specification | Добавляет на диаграмму спецификацию задачи |
|  | Task Body | Добавляет на диаграмму тело задачи |
|  | Generic *Subprogram* | Добавляет на диаграмму типовую подпрограммы(по умолчанию отсутствует) |
|  | Generic Package | Добавляет на диаграмму типовой пакет (по умолчанию отсутствует) |
|  | Database | Добавляет на диаграмму базу данных (по умолчанию отсутствует) |

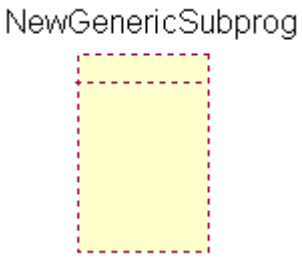
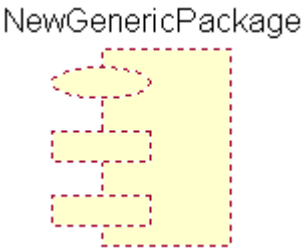
Как видно из этой таблицы, *по* умолчанию на панели инструментов отсутствуют только три графических элемента из рассмотренных ранее элементов диаграммы *компонентов*, а именно - кнопки с пиктограммами типовой подпрограммы, типового пакета и *базы данных*. При необходимости их можно добавить на специальную панель диаграммы *компонента* стандартным способом.

*Программа IBM* Rational Rose 2003 не поддерживает графические *стереотипы*, рассмотренные в лекции 12 курса "Основы *объектно-ориентированного моделирования* в нотации *UML*", и предлагает *целый* ряд собственных *стереотипов*. Графическое изображение этих *стереотипов* и их краткая характеристика приводятся в

следующей таблице (табл. 12.2). При этом каждому из *компонентов*, как правило, соответствует отдельный *файл* исходной сборки программного приложения.

Таблица 12.2. Графическое изображение *стереотипов компонентов* и их характеристика

| **Графическое изображение и имя по умолчанию** | **Название стереотипа** | **Характеристика стереотипа *компонента*** |
|---|---|---|
|  | *Subprogram* Specification | Спецификация подпрограммы. Содержит *описание переменных*, процедур и функций и не содержит определений классов |
|  | *Subprogram* Body | Тело подпрограммы. Содержит реализацию процедур и функций, не относящихся к каким-то классам, при этом не содержит определений классов или реализаций операций других классов |
|  | Main Program | Главная программа. Реализует базовую логику работы программного приложения и содержит ссылки на другие *компоненты* модели |
|  | Package Specification | *Спецификация пакета*. Содержит определение класса, его атрибутов и операций. В языке программирования C++ *спецификации пакета* соответствует отдельный файл с расширением "h" |
|  | Package Body | Тело пакета. Содержит код реализации *операций класса*. В языке программирования C++ *спецификации пакета* соответствует отдельный файл с расширением "*cpp*" |
|  | Task Specification | Спецификация задачи. Может содержать определение класса, его атрибутов и операций, которые предполагается использовать в независимом *потоке управления* |

| | | |
|---|---|---|
|  | Task Body | Тело задачи. Может содержать реализацию *операций класса*, которые имеют независимый поток управления. |
|  | Generic *Subprogram* | Типовая подпрограмма. Содержит *описание переменных*, процедур и функций, которые могут быть использованы в нескольких программных приложениях. При этом типовая подпрограмма не содержит определений классов |
|  | Generic Package | Типовой пакет. Содержит определение класса, его атрибутов и операций, которое может быть использовано в нескольких программных приложениях |
|  | Database | База данных. Содержит определение одного или нескольких классов, их атрибутов и, возможно, операций. При этом соответствующие классы могут быть реализованы в форме одной или нескольких таблиц базы данных |

Использование рассмотренных *стереотипов* существенно увеличивают наглядность графического представления диаграммы *компонентов* и позволяют архитектору уточнить характер реализации модели программистом на выбранном языке программирования.

Добавление компонента на диаграмму компонентов и редактирование его свойств

Для добавления *компонента* на диаграмму *компонентов* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *компонента* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. Добавить *компонент* на диаграмму можно также с помощью *операции* главного *меню*: **Tools** → **Create** → **Component** или с помощью *операции* контекстного *меню*: **New** → **Component**, предварительно выделив *представление компонентов* в браузере проекта.

В результате этих действий на диаграмме появится изображение *компонента* с маркерами изменения его геометрических размеров и предложенным средой именем *по* умолчанию, которое разработчику следует изменить. Продолжая разработку модели системы управления банкоматом, построим для нее диаграмму *компонентов*. С этой целью изменим имя диаграммы, предложенное *по* умолчанию Main, на *Диаграмма компонентов* ATM, а для первого добавленного *компонента* зададим имя MainATM.exe (рис. 12.1).

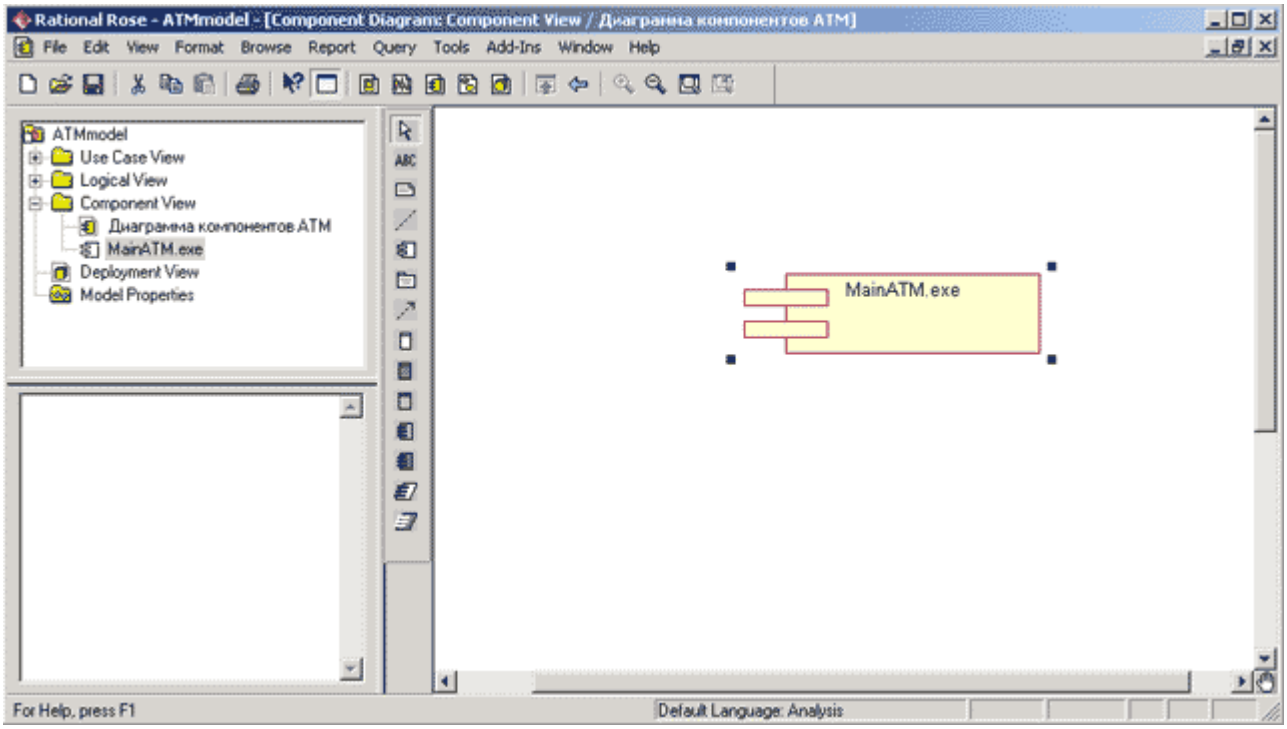



**Рис. 12.1.** Диаграмма компонентов после добавления компонента MainATM.exe

Для каждого *компонента* можно определить различные свойства, такие как стереотип, *язык программирования*, декларации, реализуемые классы. Редактирование этих свойств для произвольного *компонента* осуществляется с помощью диалогового окна спецификации свойств (рис. 12.2).

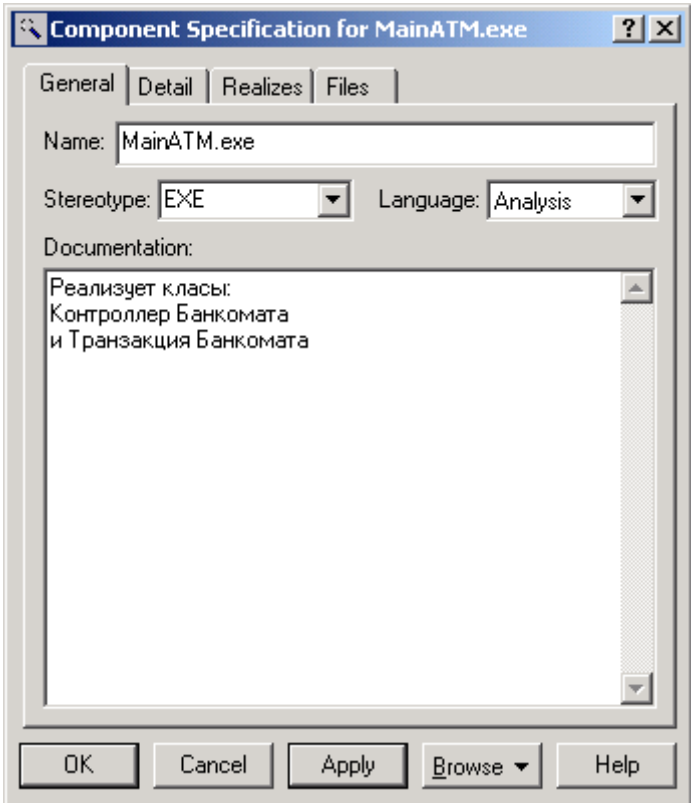



**Рис. 12.2.** Диалоговое окно спецификации свойств компонента MainATM.exe

В частности, для *компонента* MainATM.exe можно выбрать стереотип <<EXE>> из предлагаемого вложенного списка, поскольку применительно к разрабатываемой модели предполагается реализация этого *компонента* в форме исполнимого файла. При этом на вкладке Realizes (Реализует) содержатся все классы, включая и актеров, которые на данный момент присутствуют в модели (рис. 12.3). Следует заметить, что классы будут показаны в этом окне только при выбранном свойстве **Show all classes** (Показать все классы).

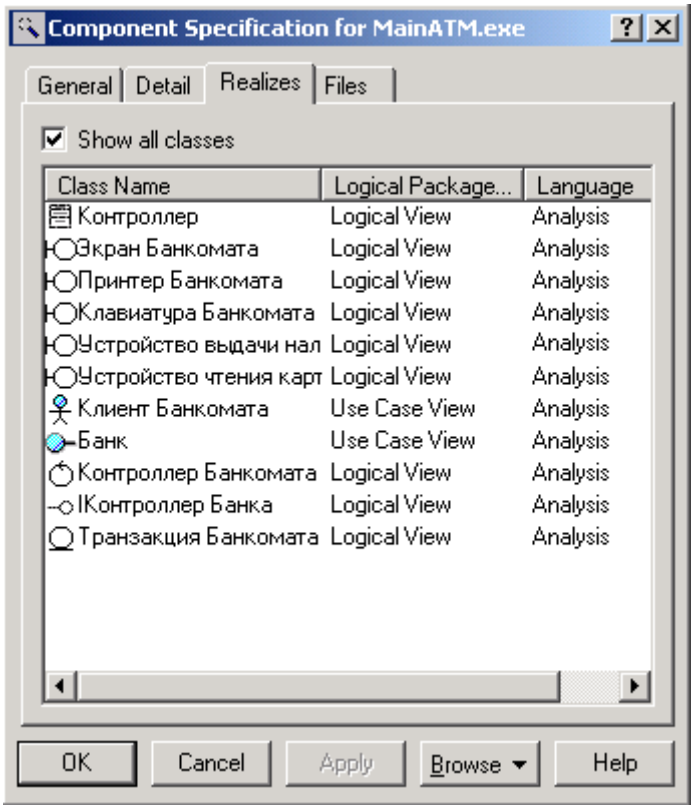



**Рис. 12.3.** Диалоговое окно спецификации свойств компонента MainATM.exe, открытое на вкладке Realizes (Реализует)

*По* умолчанию в среде *IBM* Rational Rose 2003 для всех добавляемых на диаграмму *компонентов* в качестве языка реализации используется язык анализа, который в последствии следует изменить на тот *язык программирования*, который предполагается использовать для написания программного кода. В дальнейшем при генерации программного кода необходимо будет дополнительно выбрать те классы, которые реализует тот или иной *компонент* модели. *Программа IBM* Rational Rose 2003 поддерживает возможность использования различных языков программирования для реализации различных *компонентов* модели.

Добавление отношения зависимости и редактирование его свойств

Добавление отношения зависимости на диаграмму *компонентов* аналогично добавлению соответствующего отношения на диаграмму вариантов использования. Продолжая разработку модели банкомата, на диаграмму *компонентов* предварительно следует добавить второй *компонент* с именем MainBank, для которого выбрать стереотип **Main Program**. Для добавления зависимости между двумя *компонентами* нужно с помощью левой кнопки мыши нажать кнопку с изображением зависимости на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходного *компонента* на диаграмме и отпустить ее на изображении целевого *компонента*. В результате этих действий на диаграмме появится изображение отношения зависимости в форме пунктирной линии со стрелкой, соединяющей два выбранных *компонента*.

Применительно к диаграмме *компонентов* модели банкомата рассмотренным способом следует добавить *отношение* зависимости от *компонента* с именем MainATM.exe к *компоненту* с именем MainBank. В *дополнение* к этому для наглядности можно указать в форме примечаний те классы модели, которые предполагается реализовать в данных *компонента* х (рис. 12.4).

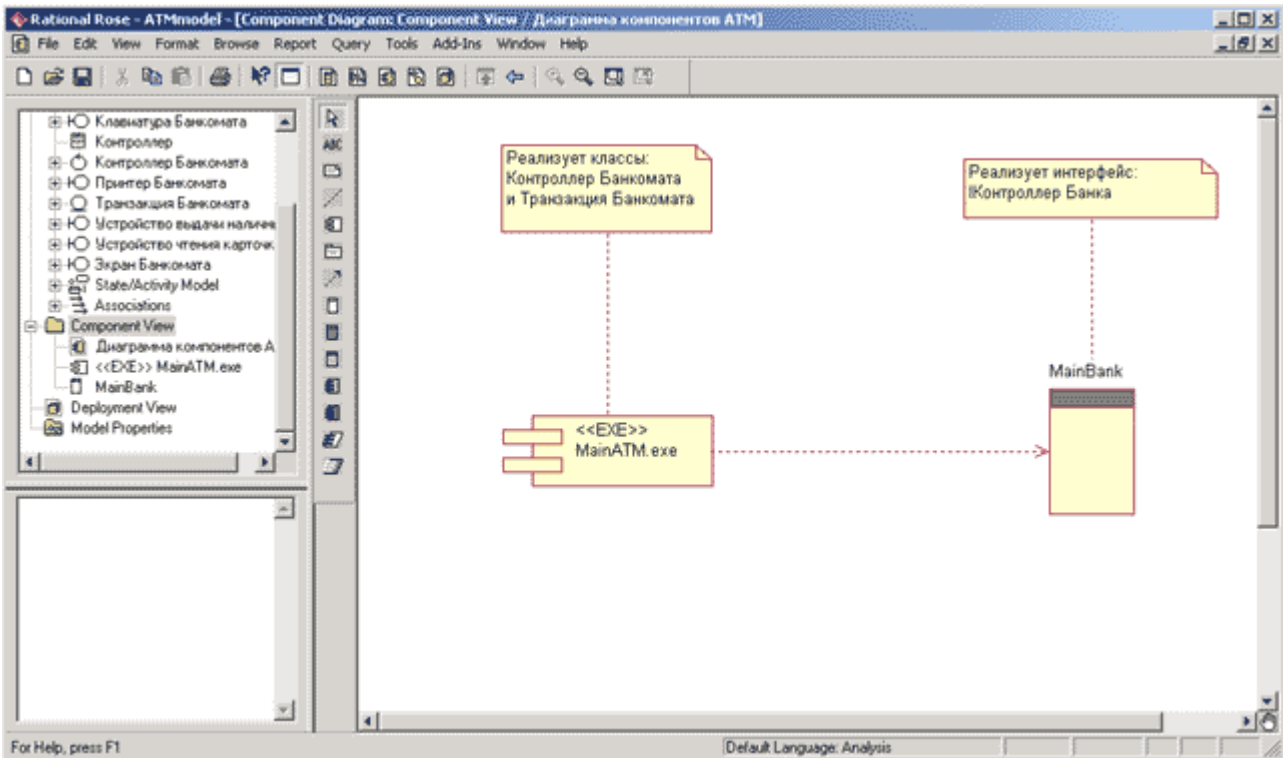



**Рис. 12.4.** Диаграмма компонентов после добавления отношения зависимости между компонентами MainATM.exe и MainBank

Следует заметить, что *отношение* зависимости в среде *IBM* Rational Rose 2003 не имеет собственного окна спецификации свойств. Именно *по* этой причине специфицировать свойства данного отношения, такие как имя и стереотип, можно только с помощью текстовой области, что нельзя признать удобным с практической точки зрения.

Окончательное построение диаграммы компонентов модели банкомата

Для завершения построения диаграммы *компонентов* рассматриваемого примера следует описанным выше способом добавить оставшиеся *компоненты* и зависимости. С этой целью следует выполнить следующие действия:

1. Добавить *компонент* с именем: Устройства Банкомата, для которого задать стереотип **Task Specification**.
2. Добавить *компоненты* с именами: Устройство чтения карточки, Клавиатура Банкомата, Принтер Банкомата, Экран Банкомата, Устройство выдачи наличных, для которых задать стереотип **Task Body**.
3. Добавить *зависимость* от *компонента* с именем MainATM.exe к *компоненту* с именем Устройства Банкомата.
4. Добавить *зависимость* от *компонента* с именем Устройство чтения карточки к *компоненту* с именем Устройства Банкомата.
5. Добавить *зависимость* от *компонента* с именем Клавиатура Банкомата к *компоненту* с именем Устройства Банкомата.
6. Добавить *зависимость* от *компонента* с именем Принтер Банкомата к *компоненту* с именем Устройства Банкомата.
7. Добавить *зависимость* от *компонента* с именем Экран Банкомата к *компоненту* с именем Устройства Банкомата.
8. Добавить *зависимость* от *компонента* с именем Устройство выдачи наличных к *компоненту* с именем Устройства Банкомата.

Построенная таким образом *диаграмма компонентов* будет иметь следующий вид (рис. 12.5).

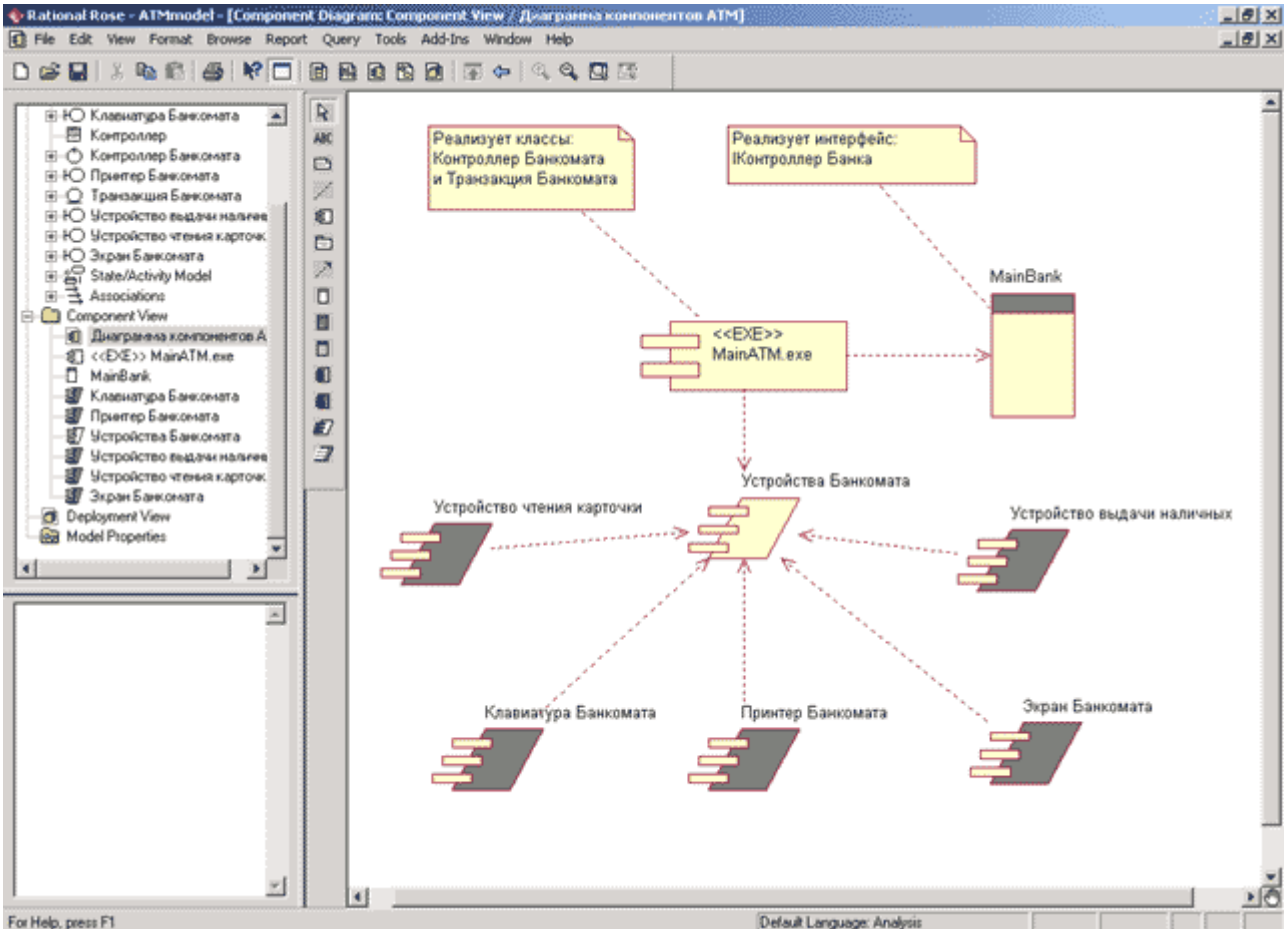



**Рис. 12.5.** Окончательный вид диаграммы компонентов разрабатываемой модели управления

Следует заметить, что различные графические *стереотипы компонентов* не оказывают влияния на особенности генерации программного кода. Поэтому при разработке диаграммы *компонентов* присутствует некоторая неоднозначность выбора соответствующих *стереотипов*, связанная с особенностями предполагаемой реализации программного приложения. При работе с диаграммой *компонентов* можно также создавать пакеты и размещать в них *компоненты*, изменять их спецификацию и отношения зависимости между различными элементами диаграммы. Выполнить эти действия предлагается читателям самостоятельно в качестве упражнения.

Лабораторная работа 21. Разработка диаграммы развертывания и редактирование свойств ее элементов.

Особенности разработки диаграммы развертывания в среде IBM Rational Rose 2003

*Диаграмма развертывания* является второй составной частью физического представления модели и разрабатывается, как правило, для территориально распределенных систем. Для разработки *диаграмм компонентов* в браузере проекта предназначено отдельное *представление* развертывания ( **Deployment View** ), в котором уже содержится *диаграмма развертывания* с пустым содержанием и без собственного имени.

Активизация *диаграммы развертывания* может быть выполнена одним из следующих способов:

- Щелкнуть на кнопке с изображением *диаграммы развертывания* на *стандартной панели инструментов*.
- Дважды щелкнуть на пиктограмме представления развертывания ( **Deployment View** ) в браузере проекта.
- Выполнить операцию главного меню: **Browse** → **Deployment Diagram** (Обзор → *Диаграмма развертывания*).

В результате выполнения этих действий появляется новое окно с чистым *рабочим листом диаграммы развертывания* и специальная *панель инструментов*, содержащая кнопки с изображением графических примитивов, необходимых для разработки *диаграммы развертывания* (табл. 13.1).

Таблица 13.1. Назначение кнопок специальной панели инструментов диаграммы развертывания

| **Графическое** | **Всплывающая** | **Назначение кнопки** |
|---|---|---|

| изображение | подсказка | |
|---|---|---|
|  | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
|  | *Text Box* | Добавляет на диаграмму текстовую область |
|  | Note | Добавляет на диаграмму примечание |
|  | *Anchor* Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
|  | Processor | Добавляет на диаграмму *процессор* |
|  | Connection | Добавляет на диаграмму отношение соединения |
|  | Device | Добавляет на диаграмму *устройство* |

Как видно из этой таблицы, по умолчанию на панели инструментов присутствуют все графические элементы из рассмотренных ранее элементов *диаграммы развертывания*, поэтому изменять специальную панель нет необходимости. Работа с диаграммой развертывания состоит в создании *процессоров* и *устройств*, их спецификации, установлении связей между ними, а также добавлении и спецификации процессов.

Добавление узла на диаграмму развертывания и редактирование его свойств

Для добавления *узла* на диаграмму развертывания нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы требуемого *узла* ( *процессора* или *устройства* ) на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. Добавить *процессор* на диаграмму развертывания можно также с помощью *операции* главного *меню*: **Tools** → **Create** → **Processor** или с помощью *операции* контекстного *меню*: **New** → **Processor**, предварительно выделив *представление* развертывания в браузере проекта. Аналогично добавить *устройство* на диаграмму можно также с помощью *операции* главного *меню*: **Tools** → **Create** → **Device** или с помощью *операции* контекстного *меню*: **New** → **Device**, предварительно выделив *представление* развертывания в браузере проекта.

В результате этих действий на диаграмме развертывания появится изображение *узла* требуемого типа с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить. При этом следует иметь в виду, что в среде *IBM* Rational Rose 2003 под процессором понимается ресурсоемкий *узел*, а под ***устройством*** - нересурсоемкий *узел*.

Продолжая разработку модели системы управления банкоматом, построим для нее диаграмму развертывания. С этой целью в качестве первого *узла* выберем тип *процессор* и зададим ему имя Банкомат №1, для которого в форме примечания укажем *помеченное значение*: {*адрес* = ул. Садовая, д.5}. Это *значение* служит для спецификации конкретного адреса одного из банкоматов системы (рис. 13.1).

Лабораторная работа 22. Особенности генерации программного кода.

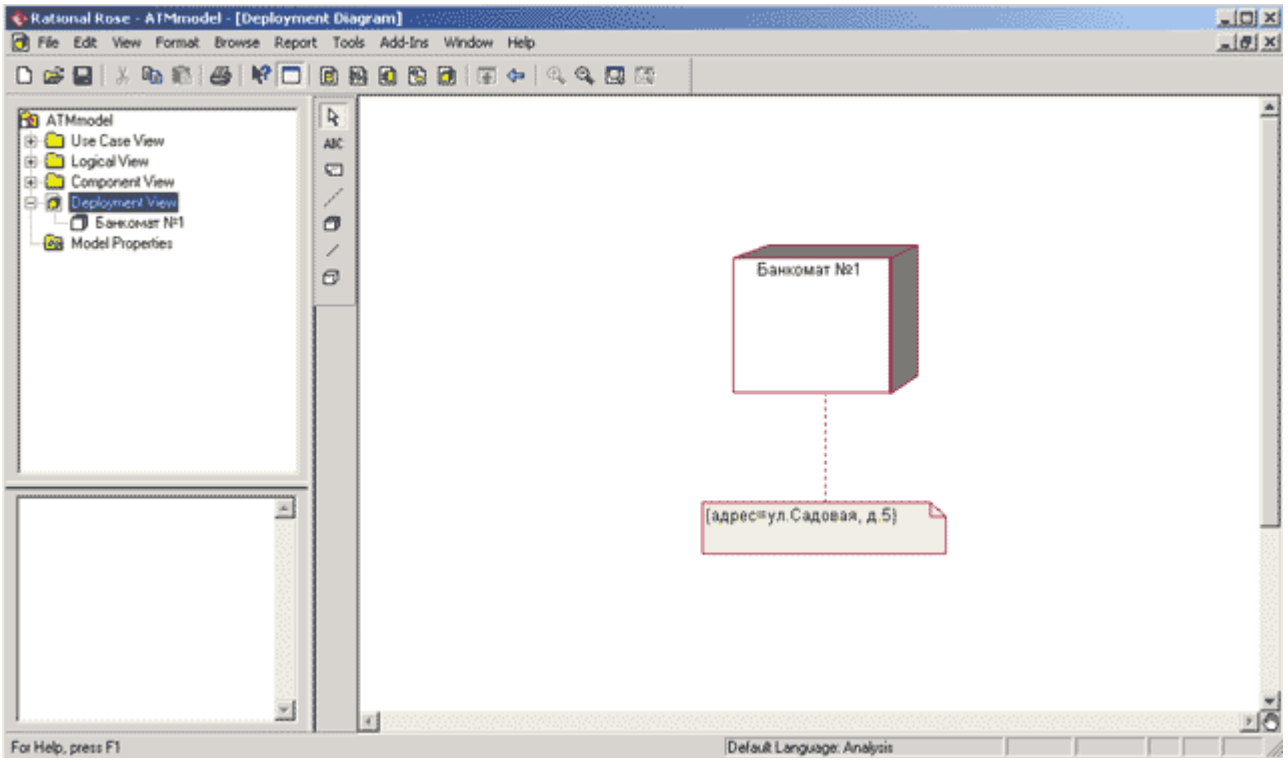



**Рис. 13.1.** Диаграмма развертывания после добавления узла Банкомат № 1

Для каждого *процессора* можно специфицировать различные свойства, такие как стереотип, характеристику, процессы и их приоритет. Спецификация этих свойств осуществляется с помощью диалогового окна спецификации свойств *процессора* (рис. 13.2).

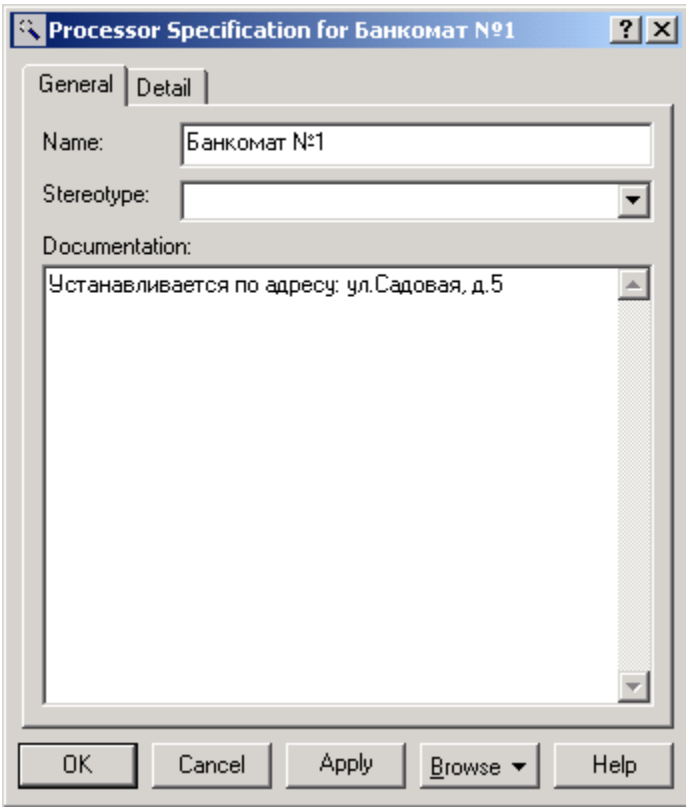



**Рис. 13.2.** Диалоговое окно спецификации свойств узла Банкомат № 1

При этом на вкладке **General** (Общие) можно только изменить имя *процессора*, ввести текст стереотипа, предложенный самим разработчиком, и текст документации, поясняющий особенности физического размещения данного компонента. На вкладке **Detail** (Подробно) окна спецификации свойств *процессора* можно определить его характеристики, выбрать процессы и вариант *планирования* его работы (рис. 13.3).

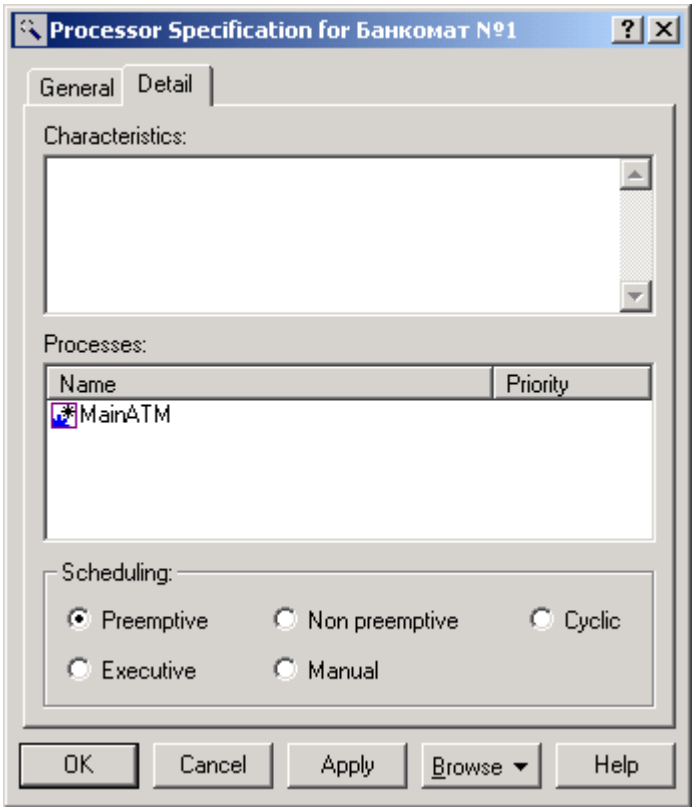



**Рис. 13.3.** Диалоговое окно спецификации свойств узла Банкомат № 1, открытое на вкладке Detail (Подробно)

Характеристики *процессора*, такие как его *быстродействие* и объем оперативной памяти, могут быть записаны в форме текста в многостраничное *поле* с именем **Characteristics**. В *поле* **Processes** (Процессы) можно задать некоторый процесс, который предполагается реализовать на данном *процессоре*. С этой целью необходимо выполнить операцию контекстного *меню* **Insert** (Вставить) и ввести текст имени процесса. Далее можно задать *приоритет процесса*, введя некоторое число в соответствующее *поле* ввода.

При наличии у *процессора* нескольких процессов может быть дополнительно определена процедура *планирования* их выполнения. Для спецификации процедуры *планирования процессора* могут быть использованы следующие варианты выбора в группе **Scheduling**:

- **Preemptive** (С приоритетом) - определяет процедуру *планирования*, при которой процесс с большим приоритетом будет иметь преимущество при использовании ресурсов *процессора* по сравнению с менее *приоритетными процессами*.
- **Non preemptive** (Без приоритета) - определяет процедуру *планирования*, при которой все *приоритеты процессов* игнорируются. При этом текущий процесс выполняется до своего завершения, после чего может быть начато выполнение следующего процесса.
- **Cyclic** (Циклический) - определяет процедуру *планирования*, при которой *приоритеты процессов* также игнорируются. Все процессы выполняются циклически по кругу, при этом каждому из них выделяется фиксированное время на выполнение, по прошествии которого управление передается следующему процессу.
- **Executive** (Исполнительный) - определяет процедуру *планирования*, для которой существует некоторый алгоритм, предназначенный для управления отдельными процессами.
- **Manual** (Вручную) - определяет процедуру *планирования*, при которой *планирование выполнения процессов* осуществляется пользователем.

Для отображения информации о процессах, выполняемых на отдельных *процессорах*, представленных на диаграмме развертывания, следует выполнить операцию контекстного *меню* **Show Processes** (Показать процессы). Для отображения информации о процедуре *планирования отдельных процессов* на выбранном *процессоре* следует выполнить операцию контекстного *меню* **Show Scheduling** (Показать планирование).

Продолжая разработку *диаграммы развертывания* для модели банкомата, следует добавить второй *узел* типа *устройство* ( **Device** ) с именем *Сеть*, для которого задать стереотип <<закрытая *сеть*>>. При этом для задания стереотипа следует ввести его текст без угловых кавычек в строку с именем **Stereotype**.

Для *устройства* набор редактируемых свойств меньше, поэтому для него с помощью соответствующего окна спецификации свойств можно определить: имя, стереотип, документацию и характеристику (рис. 13.4). Этот факт согласуется с определением ***устройства*** как нересурсоемкого *узла*, на котором отсутствует *процессор*.

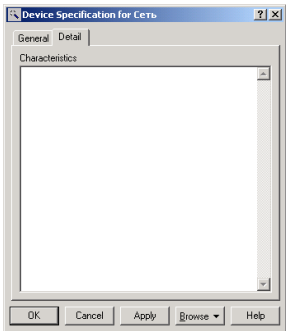



**Рис. 13.4.** Диалоговое окно спецификации свойств устройства Сеть, открытое на вкладке Detail (Подробно)
Лабораторная работа 23. Особенности генерации программного кода.

Для модели обработки заказов сгенерируйте программный код на языке C++.

***Этапы выполнения упражнения***

**Ввод тел пакетов на диаграмму компонентов системы**

1. Откройте диаграмму компонентов системы (*System*). Выберите в браузере пакет компонентов *Entities*: тело пакета *Order*. Перетащите тело пакета *Order* на диаграмму компонентов системы.

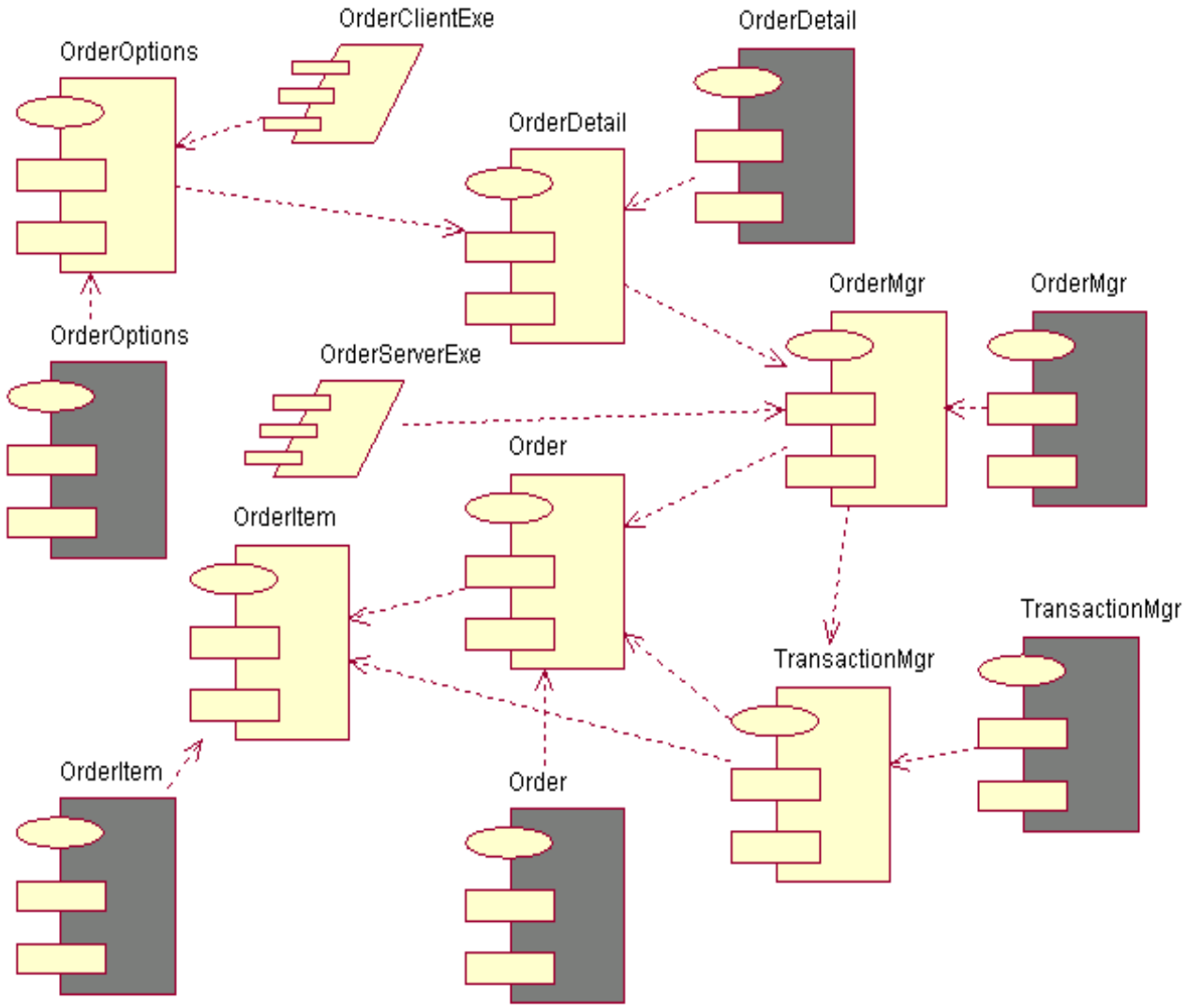
2. Повторите п. 1 для следующих компонентов (см. рис.11.1):

*Entities*: тело пакета *OrderItem*, *Boundaries*: тело пакета *OrderOptions*, *Boundaries*: тело пакета *OrderDetail*, *Control*: тело пакета *TransactionMgr* и *Control*: тело пакета *OrderMgr*.

**Проверка модели Rose**

Выберите в меню *Tools* > *Check Model*. Проанализируйте и исправьте все найденные ошибки в окне журнала.

**Обнаружение нарушений правил доступа**




Откройте главную диаграмму классов модели (Main) в логическом представлении (Logical view) модели. Выберите в меню *Report* > *Show Access Violations.* Проанализируйте и исправьте все нарушения правил доступа, показанные в соответствующем диалоговом окне.

*Рис.11.1.* Диаграмма компонентов системы Order

**Установка языка C++**

Выберите в меню *Add-Ins* > *Add-In Manager*. В диалоговом окне *Add-In Manager* поставьте флажок для варианта *Rose C++* и нажмите кнопку *Apply* (Применить), затем *OK*.

**Назначение компонентам системы языка генерации программного кода**

1. Откройте спецификацию компонента *Order* (спецификацию пакета) в пакете компонентов *Entities.* Выберите в качестве языка реализации *C++*.
2. Повторите п. 1 для остальных компонентов системы (см. рис.11.1).

**Назначение свойств генерации программного кода**

1. Любым способом создайте корневой каталог для генерируемого программного кода, где будут созданы все каталоги и файлы C++ (например, *c:\order*).

2. Выберите в меню *Tools* > *Options* на вкладке C++ *Type: Project* для установки свойств генерации программного кода проекта.

3. Нажмите кнопку *Clone* для создания своей копии набора свойств проекта. Введите имя для нового набора свойств проекта (например, *MyPropertySet*) и выберите его в раскрывающемся списке *Set* диалогового окна *Options*.

4. Для свойства *Directory* в столбце *Value* необходимо набрать имя созданного корневого каталога для генерируемого программного кода проекта (например, *c:\order*). Нажмите кнопку *Apply* (Применить) и *OK*.

**Генерация программного кода C++**

1. Откройте диаграмму компонентов системы (*System*). Выберите все компоненты на диаграмме *System* посредством меню *Edit >Select All*.

2. Выберите в меню *Tools* > *C++* > *Code Generation*. Просмотрите созданный программный код в соответствующем каталоге для генерируемого программного кода проекта.

**Лабораторная работа 24. Проектирование информационной системы в рамках RUP.**

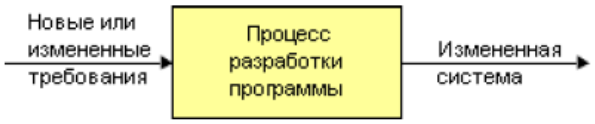
Цель работы – научиться разрабатывать модели потока работ; понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Сегодня трудно найти книгу или статью, посвященную технологии создания больших программных систем, которая не цитировала бы убийственную характеристику нынешней ситуации: “По нашим оценкам только 26% проектов создания ИС заканчиваются успешно”. (Standish Group CHAOS Report 1998)

# Rational Unified Process как технология

Rational Unified Process - это процесс разработки программного обеспечения.

**Процесс** – частично упорядоченный набор шагов, которые нужно проделать для достижения цели; при разработке программного обеспечения цель состоит в формировании или расширении существующего программного изделия.




Процесс разработки программного обеспечения - процесс разработки системы из требований, новых (начальный цикл развития) или измененных (цикл эволюции).

Когда программная система разрабатывается на пустом месте, ее разработка – это процесс создания системы из требований. Но как только система приняла какую-то форму (или в наших терминах, когда система прошла начальный цикл развития), то любая доразработка – это процесс приспособления системы к новым или изменившимся требованиям. Разработка и все последующие доразработки составляют **жизненный цикл системы**.

Rational Unified Process обеспечивает строгий подход к назначению задач и ответственности в пределах группы разработки. Его **цель** состоит в том, чтобы гарантировать высокое качество программного продукта, отвечающего потребностям конечных пользователей, в пределах предсказуемого временного графика и бюджета.

- Rational Unified Process – это **итеративный** процесс. Создавать современные сложные программные системы последовательно, т.е. сначала определять все проблемы, затем принимать все проектные решения, формировать программное обеспечение и, наконец, проверять изделие, невозможно. **Итерационный подход** позволяет улучшать понимание проблемы через последовательные усовершенствования и с приращением конкретизировать эффективные решения. Этот подход обеспечивает большую гибкость при учете новых требований или тактических изменений в деловых целях, и позволяет проекту заранее **идентифицировать и разрешать риски**.
- Rational Unified Process – это **управляемый** процесс. Итерационный подход предполагает **управление требованиями** и **управление изменениями**, чтобы по всем пунктам вовремя

гарантировать общее понимание ожидаемых функциональных возможностей, ожидаемый уровень качества и гарантировать наилучшее управление связанными затратами и графиками выполнения.

- Rational Unified Process заключается в создании и обслуживании **моделей**. Rational Unified Process фокусирует внимание не на создании большого количества бумажных документов, а на развитии и эксплуатации моделей - семантически богатых представлений программной системы при ее разработке.
- Rational Unified Process сосредотачивает внимание на первоначальной разработке и компоновке устойчивой **архитектуры** программы, которая облегчает параллельную разработку, минимизирует переделки, увеличивает возможность многократного использования и надежность эксплуатации. Эта архитектура используется для планирования использования и управления развитием программных **компонентов**.
- Действия при выполнении Rational Unified Process **управляются прецедентами**. Понятия прецедентов и сценария управляют технологическим маршрутом от делового моделирования и требований до испытаний, и обеспечивают связанные и доступные для анализа маршруты разработки и поставки системы.
- Rational Unified Process поддерживает **объектно-ориентированную технологию**. Некоторые из моделей являются объектно-ориентированными моделями, которые базируются на понятиях объектов, классов и зависимостей между ними. Эти модели, подобно многим другим техническим искусственным объектам (артефактам), используют Унифицированный язык моделирования (UML) как общую систему обозначений.
- Rational Unified Process поддерживает **компонентно-ориентированное программирование**. Компоненты - это нетривиальные модули или подсистемы, которые выполняют конкретную функцию и могут быть смонтированы в строго очерченной архитектуре, специальной или некоторой общедоступной инфраструктуре компонентов, типа Internet, CORBA, COM/DCOM, для которых появляется индустрия многократно используемых компонентов.
- Rational Unified Process – это процесс **с перестраиваемой конфигурацией**. ***Никакой одиночный процесс не подходит для всех случаев разработки программного обеспечения.*** Rational Unified Process удовлетворяет и маленькие группы разработчиков и большие организации. Rational Unified Process основан на простой и корректной архитектуре, которая обеспечивает общность для семейства процессов, и все же может быть изменена ради приспособления к конкретным ситуациям. Он содержит рекомендации по конфигурированию процесса для удовлетворения потребностей данной организации.
- Rational Unified Process поощряет объективно осуществляемое **управление качеством**. Оценка качества всех действий и их участников, формируемая в процессе, использует объективные измерения и критерии.
- Rational Unified Process поддерживается **инструментальными средствами**, которые автоматизируют большинство действий процесса. Инструментальные средства используются для создания и обслуживания различных артефактов процесса разработки программного обеспечения: визуального моделирования, программирования, испытаний и так далее. Они неоценимы в поддержке всей бухгалтерии, связанной с управлением изменениями и управлением конфигурацией, которыми сопровождается каждая итерация.

9. Настройка среды разработки (Enviroment).


Рис. Дисциплины RUP

Любая дисциплина RUP включают в себя все элементы описания бизнес процесса:

- цели;
- концепции;
- поток работ;
- описание видов деятельности;
- описание артефактов;
- рекомендации;
- шаблоны документов.

Таблица.1.1.- Этапы работ по RUP, модели и диаграммы UML в Rational Rose

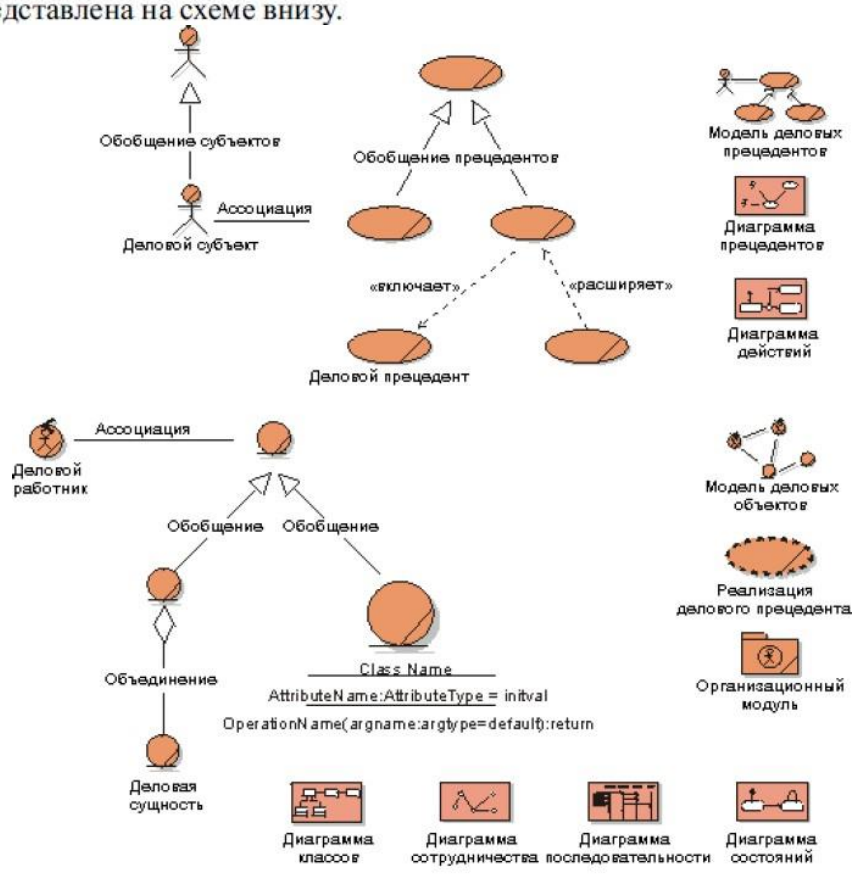
| Этап работ по RUP | Модели | Диаграммы UML | Примечания |
|---|---|---|---|
| Бизнес моделирование (Business Modeling) | Бизнес процессы (business use case model) | Use case diagram | Модель отображает процессы, подлежащие автоматизации, связи между процессами, цели, которые они поддерживают, субъектов и объектов, взаимодействующих с бизнес процессами и являющихся внешними по отношению к ним, например клиентами и партнерами. Модель используется для определения целей системы и разбиения системы на подсистемы. Каждому бизнес процессу ставится в соответствие подсистема |
| | Описание бизнес процессов (business object model RUP 2002 или business analysis model RUP 2003) | Activity diagram | Модель отображает поток работ по бизнес процессу. Модель используется для определения модулей подсистем и их функций. |

| | | | |
|---|---|---|---|
| | Описание бизнес сущностей (business object model RUP 2002 или business analysis model RUP 2003) | Class diagram, Use case diagram | Модель отображает сущности реального мира (business entity), их атрибуты. Модель используется для формирования альбомов входных и выходных форм системы, проектирования пользовательского интерфейса, баз данных, классов, реализующих функции |
| | Описание состояния бизнес сущности (business object model RUP 2002 или business analysis model RUP 2003) | Activity diagram, Statechartdia gram. | Модель отображает состояния сущности реального мира. Модель используется для определения скрытых атрибутов бизнес сущностей и при определении функций системы. |
| | Роли и автоматизируемые виды деятельности (business object model RUP 2002 или business analysis model RUP 2003) | Class diagram, Use case diagram | Модель отображает роли и их автоматизируемые виды деятельности. Модель используется при определении функций системы |
| | Структура предприятия (business object model RUP 2002 или business analysis model RUP 2003) | Class diagram, Use case diagram | Модель отображает структуру автоматизируемого предприятия. Модель используется для определения функций системы |
| | Бизнес правила | Class diagram, Activity diagram | Модель отображает ограничения, накладываемые на бизнес процессы. Модель используется для определения правил системы |
| Определение требований (Requirements) | Функции системы (Use case model) | Use case diagram | Модель отображает функции системы |
| | Экранные формы | Class diagram | Модель отображает экранные формы системы |
| | Сценарии работы пользователя с системой | Activity diagram | Модель отображает сценарии работы пользователя с системой |
| Анализ и проектирование (Analysis & Design) | Модель размещения (Deployment model) | Deployment diagram | Модель отображает технические средства и, размещенные на них, программные средства системы и прочие программные средства |
| | Модель данных (Data modal) | Class diagram | Модель отображает логическую и физическую структуру данных. |

| | | | |
|---|---|---|---|
| | Модель анализа (Analysis modal) | Class diagram | Модель описывает реализацию требований и служит абстракцией для модели проектирования |
| | Модель проектирования (Design modal) | Class diagram, Sequence diagram, Activity diagram, Collaboration diagram | Модель описывает реализацию требований и служит абстракцией модели реализации и исходного кода |
| Реализация (Implementation) | Модель реализации (Implementation model) | Component diagram | Модель отображает подсистемы и компоненты, из которых они состоят |
| Тестирование (Test) | Модель тестирования (Test suite) | Class diagram, Activity diagram | Модель отображает контрольные примеры, тесты, последовательность выполнения тестов, ожидаемые и полученные результаты тестов |
| Размещение (Deployment) | Модель размещения (Deployment model) | Deployment diagram | Модель отображает технические средства и размещенные на них программные средства системы и прочие программные средства |

# Основные понятия

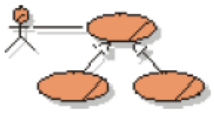
Номенклатура основных понятий делового моделирования (модели, диаграммы, объекты моделей и их связи) представлена на схеме внизу.




Основные понятия делового моделирования. Каждое понятие представлено своим стандартным обозначением в Rational Unified Process

Деловое моделирование в Rational Unified Process рассматривает деловую сферу с двух точек зрения: **внешней** (что полезного делает деловая сфера для своих контрагентов – **модель деловых прецедентов**) и **внутренней** (как деловая сфера это делает с позиций внутреннего устройства и функционирования – **модель деловых объектов**). На иллюстрации этим моделям отведены, соответствеено, верхняя и нижняя части.
Рассмотрим некоторые из понятий делового моделирования более подробно.

### Модель деловых прецедентов



**Модель деловых прецедентов** - модель, которая описывает процессы деловой сферы и их взаимодействие с внешней средой, например, с заказчиками и партнерами.

Основная цель создания модели деловых прецедентов состоит в том, чтобы описать, как используется деловая сфера заказчиками и партнерами. При этом могут быть показаны действия, которые непосредственно касаются заказчика или партнера, а также действия по поддержке или управлению задачами, которые косвенно касаются внешней стороны дела.

Модель описывает деловую сферу в терминах деловых субъектов и деловых прецедентов.

### Деловой субъект

**Деловой субъект** представляет роль, которую кто-то или что-то играет в деловой среде.

Деловой субъект имеет **имя**.

Чтобы понять цель деятельности, Вы должны знать **кто** взаимодействует с деловой сферой; то есть, кто размещает запросы или заинтересован в получении ответа. Различные типы «взаимодействующих субъектов» представляются как **деловые субъекты**.

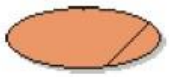
Термин «субъект» означает роль, которую играет кто-то или что-то при взаимодействии в деловой среде. Ниже перечислены типы пользователей, которые потенциально могут быть деловыми субъектами:

- Заказчики
- Поставщики
- Партнеры
- Потенциальные заказчики («рынок»)
- Властные органы
- Коллеги из не моделированных частей деловой сферы.

Чаще всего субъект соответствует пользователю-человеку. Однако, бывают ситуации, когда роль субъекта играет, например, информационная система. Если сетевые услуги вашего банка настолько хороши, что Вы можете управлять большинством операций банка с персонального компьютера из своего помещения, то ваш прецедент, взаимодействующий с субъектом «поставщик денег» (банк), фактически взаимодействует с информационной системой.

Субъект представляет специфический тип делового пользователя, а не реального физического пользователя. Несколько физических пользователей деловой сферы могут играть относительно нее одну и ту же роль; то есть они действуют как экземпляры одного и того же субъекта. В то же время один и тот же субъект может действовать как несколько различных субъектов. Это означает, что один и тот же человек может воплощаться в экземплярах различных субъектов.

## Деловой прецедент



**Экземпляр делового прецедента** - последовательность действий, выполняемых в деловой сфере, которая производит видимый результат, имеющий значение для индивидуального субъекта деловой сферы.

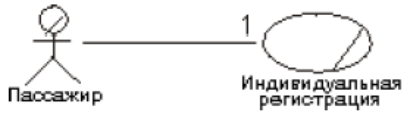
**Деловой прецедент** определяет набор экземпляров деловых прецедентов. Деловой прецедент имеет **имя**.

Процессы деловой сферы определяются как ряд различных деловых прецедентов, каждый из которых представляет определенный поток работ в деловой сфере. Деловой прецедент определяет то, что должно случиться в деловой сфере, когда он выполнен; он описывает эффект последовательности действий, которые производят результат, ценный для конкретного делового субъекта.
С точки зрения индивидуального субъекта деловой прецедент определяет законченный поток работ, который произвел нужные результаты. Это подобно тому, что называют «деловым процессом», но «деловой прецедент» имеет намного более строгое определение.

Совокупность деловых прецедентов представляет все возможные способы использования деловой сферы.

**Пример:**

Когда экземпляр делового субъекта Пассажир подходит к стойке регистрации пассажиров и предъявляет свои билет и багаж, он посылает сообщение экземпляру прецедента Индивидуальная регистрация. В конце процедуры регистрации деловой прецедент распечатает и выдаст пассажиру посадочный талон и одну или более багажных квитанций. Пассажир может связываться только с одним экземпляром прецедента Индивидуальная регистрация. Таким образом, множественность связи равна [1].




При регистрации в аэропорту Пассажир взаимодействует с прецедентом Индивидуальная регистрация.

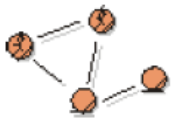
**Пример:**

Субъекты Пассажир бизнес-класса и Турист одинаково являются внешними объектами делового прецедента, который выполняет регистрацию. Их общая роль, смоделированная деловым субъектом Пассажир, унаследована двумя первоначальными деловыми субъектами. Мы отображаем эту ситуацию с использованием связей обобщения.


Субъекты Пассажир бизнес-класса и Турист наследуют все атрибуты Пассажира. Таким образом, оба эти субъекта могут действовать как пассажиры.

**Пример:**

## Модель деловых объектов



**Модель деловых объектов** – это объектная модель, описывающая реализацию деловых прецедентов. Она служит абстракцией того, как должны быть связаны деловые работники и деловые сущности, и как они должны сотрудничать в деловой сфере.

Модель деловых объектов описывает прецедент деловой сферы с внутренней точки зрения деловых работников. Модель определяет, **как** люди, которые работают в деловой сфере, и вещи, которые они обрабатывают и используют (классы и объекты деловой сферы), должны быть связаны друг с другом, статически и динамически, чтобы произвести ожидаемые результаты. Все вместе объекты классов модели должны быть способны выполнить все прецеденты деловой сферы.

Оформление и обработка специального багажа включается в индивидуальную регистрацию только в тех случаях, когда пассажир имеет такой багаж и должен перейти к специальной стойке оформления специального багажа.


Поток работ прецедента Оформление и обработка специального багажа вставлен в прецедент Индивидуальная регистрация с использованием связи расширения.

## Деловая сущность

**Деловая сущность** представляет «вещь», которую обрабатывает или использует деловой работник.

Деловые сущности представляют «вещи», обрабатываемые или используемые деловыми работниками при выполнении делового прецедента. Деловая сущность часто представляет что-то ценное для нескольких деловых прецедентов или экземпляров прецедентов, так что объекты деловой сущности могут быть довольно долгоживущими. Вообще, считается хорошей практикой, если деловая сущность не содержит никакой информации о том, как и для кого она используется.

Как правило, деловая сущность представляет документ или существенную часть изделия. Иногда она представляет что-то менее материальное, вроде важной информации относительно рынка или заказчика. Примеры деловых сущностей в ресторане - Меню и Напиток; в аэропорту – Билет и Посадочный талон.

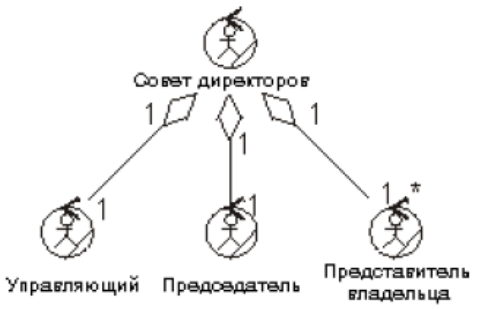
Вы должны моделировать как деловые сущности только те явления, к которым должны обращаться другие классы в модели деловых объектов. Другие «вещи» могут быть смоделированы как **атрибуты** соответствующих классов или только описаны в этих классах в виде текста.

Теоретически любое явление может быть смоделировано как класс. Однако, использование атрибутов может сократить число поддерживаемых классов и сделать объектную модель более простой для понимания.

**Операция** определяет инструмент, которым управляется деловая сущность. **Операция** имеет **название** и, может быть, **параметры**. Доступ к объекту деловой сущности показывается как **сообщение**, посылаемое объекту деловой сущности.

**Пример:**

Совет директоров компании состоит из председателя, управляющего высшего ранга и нескольких представителей владельца.




Класс-объединение включает другие классы.

**Пример:**

Пассажиры, проходящие регистрацию в аэропорту, имеют различные виды багажа: обычный багаж, ручную кладь и специальный багаж. С точки зрения авиакомпании, все эти виды багажа имеют несколько общих свойств. Кроме того, что все они являются багажом, каждый из них имеет, например, владельца и вес. Эти общие свойства могут быть смоделированы атрибутами и операциями в отдельном классе под названием Багаж. Обычный багаж, Ручная кладь и Специальный багаж наследуют свойства этого класса.


Классы Обычный багаж, Ручная кладь и Специальный багаж имеют общие свойства. Все они являются специализациями обобщённого понятия Багаж.

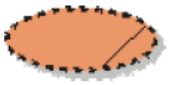
## Организационный модуль

**Организационный модуль** включает деловых работников, деловые сущности и другие организационные модули, которые объединяются в соответствии с некоторым критерием. Организационный модуль имеет **название**.

Как правило, деловую сферу структурируют, организуя служащих в департаменты, отделы, группы и так далее, согласно различным критериям. Эта ситуация моделируется распределением деловых работников и деловых сущностей по организационным модулям.

Организационный модуль - конструкция для структурирования деловой сферы путем деления ее на меньшие части. Организационные модули могут содержать не только деловых работников и деловые сущности, но также и другие организационные модули, что позволяет Вам формировать иерархию организационных модулей. Деловой работник или деловая сущность могут быть непосредственными элементами только одного организационного модуля.

### Реализация делового прецедента



**Реализация делового прецедента** описывает, как реализован конкретный деловой прецедент в пределах модели деловых объектов, в терминах сотрудничества объектов.

Модель деловых прецедентов описывает деловую сферу в терминах деловых субъектов и деловых прецедентов, соответствующих заказчикам и деловым процессам. Модель деловых прецедентов содержит описания потока работ, которые идентифицируют то, **что** делается. То, **как** выполняется работа в каждом деловом прецеденте, описывается в модели деловых объектов.

Набор работников, которые выполняют работу делового прецедента, и деловых объектов, к которым они обращаются и которыми управляют в процессе работы, называется реализацией делового прецедента. Объекты одного и того же класса могут участвовать в нескольких различных реализациях деловых прецедентов, отражая тот факт, что один и тот же вид ресурса одновременно участвует в различных процессах.

### Диаграмма прецедентов

**Диаграмма прецедентов** показывает деловые субъекты, деловые прецеденты, пакеты деловых прецедентов и связи между ними.

Никаких строгих правил относительно того, что нужно показывать в диаграммах прецедентов, не существует. Вы показываете те связи в модели, которые считаете интересными. Могут быть полезны следующие диаграммы:

- Деловые субъекты, принадлежащие к одному и тому же пакету прецедентов.
- Деловой субъект и все деловые прецеденты, с которыми он взаимодействует. Диаграмма этого типа может функционировать как локальная диаграмма делового субъекта и, вероятно, должна быть с ним связана.
- Деловые прецеденты, обрабатывающие одну и ту же информацию.
- Деловые прецеденты, используемые одной и той же группой субъектов.
- Деловые прецеденты, которые часто выполняются в одной последовательности.
- Деловые прецеденты, принадлежащие одному пакету прецедентов.
- Наиболее важные деловые прецеденты. Диаграмма этого типа может функционировать как резюме модели.
- Определенный деловой прецедент и его связи с деловыми субъектами и другими деловыми прецедентами. Диаграмма этого типа может функционировать как локальная диаграмма делового прецедента и, вероятно, должна быть с ним связана.

### Диаграмма действий

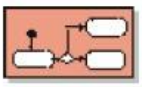


**Диаграмма действий** в модели деловых прецедентов иллюстрирует поток работ делового прецедента.

Поток работ делового прецедента описывает последовательность действий, которые вместе что-то производят для делового субъекта. Поток работ часто состоит из основного потока и одного или нескольких альтернативных потоков.

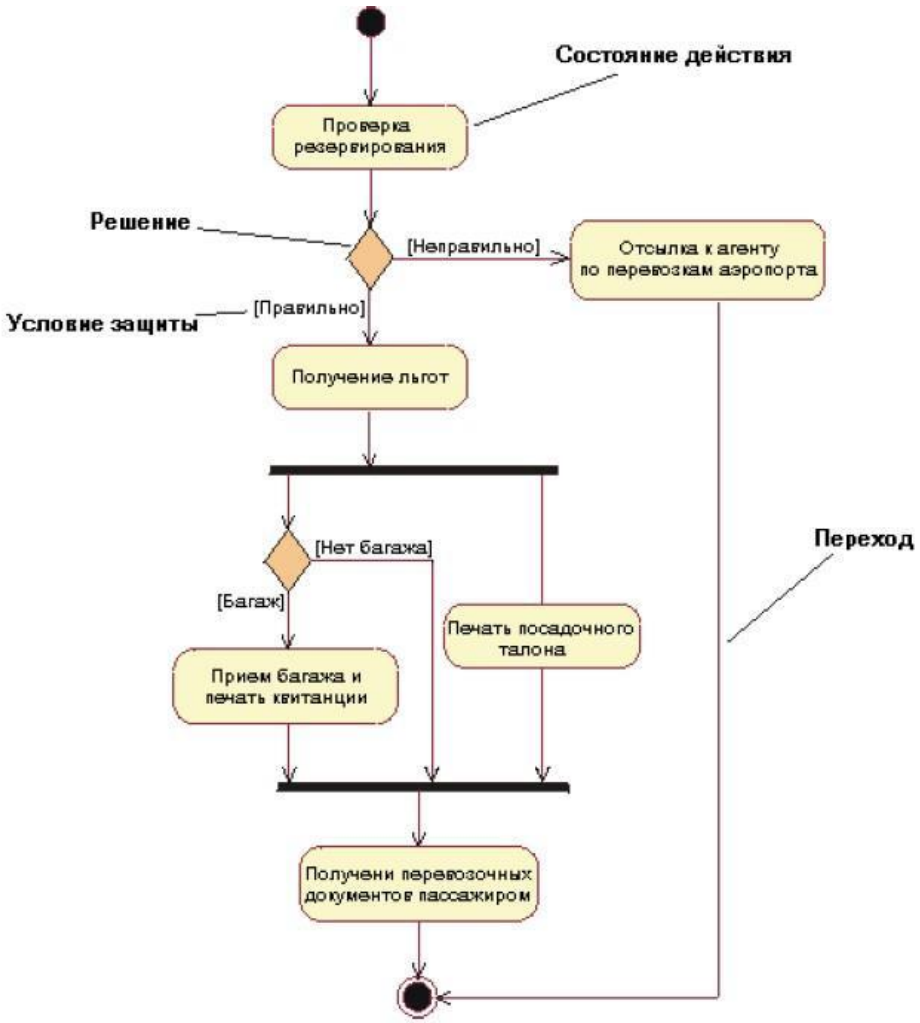



Диаграмма действий для делового прецедента Индивидуальная регистрация в модели деловых прецедентов регистрации в аэропорту.

## Диаграмма классов

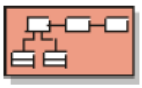


**Диаграмма классов** показывает совокупность декларативных (статических) элементов модели, таких как классы и пакеты, а также их содержание и связи.

Диаграммы классов показывают связи ассоциации, объединения и обобщения между деловыми работниками и деловыми сущностями. Могут представлять интерес следующие виды диаграмм классов:

- Иерархии наследований.
- Объединения деловых работников и деловых сущностей.
- Как деловые работники и сущности связаны посредством ассоциаций.

## Диаграмма сотрудничества

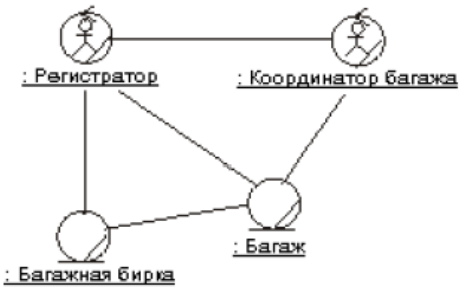
**Диаграмма сотрудничества** описывает экземпляр взаимодействия между объектами; она показывает объекты, участвующие во взаимодействии, с их связями друг с другом и с сообщениями, которые они посылают друг другу.

Различают групповые и рабочие диаграммы сотрудничества.

Групповая диаграмма сотрудничества содержит деловых работников, деловые сущности и связи между ними. Если это сделает диаграмму более информативной, Вы можете также включать в диаграмму деловых субъектов, которые связаны с деловыми работниками.

В групповую диаграмму сотрудничества не включают сообщения и порядковые номера сообщений. Вы не можете показать, как передается между работниками управление поведением. Часто, но не всегда, это достаточный уровень детализации для определения внутреннего поведения делового прецедента. Если уровень детализации не достаточен, разработайте диаграмму последовательности или рабочую диаграмму сотрудничества критической части.

**Пример:**




Групповая диаграмма сотрудничества для прецедента Индивидуальная регистрация.

## Диаграмма последовательности

**Диаграмма последовательности** описывает экземпляр взаимодействий между объектами, размещая их в хронологическом порядке; она показывает «линии жизни» объектов, участвующих во взаимодействии, и сообщения, которые они посылают друг другу.

Диаграмма последовательности графически изображает подробности взаимодействия деловых работников и деловых субъектов и то, как обращаются к деловым сущностям в ходе выполнения делового прецедента. Диаграмма последовательности кратко описывает то, что делают участвующие деловые работники, как управляются деловые сущности (в терминах запусков), и как они связываются, посылая сообщения друг другу.

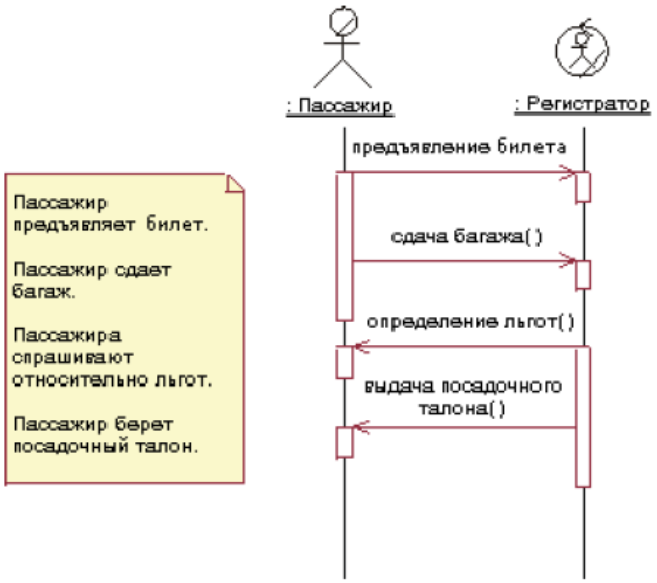



Диаграмма последовательности части делового прецедента Индивидуальная регистрация.

## Диаграммы состояний

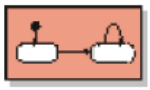


**Диаграмма состояний** представляет машину состояний, поведение которой определяется последовательностью состояний, проходимых объектом в течение его жизни в ответ на события, вместе с его ответами и действиями.

Диаграмму состояний можно использовать для иллюстрации состояний, которые может иметь деловой работник или деловая сущность, событий, которые вызывают переход из одного состояния в другое, и действий, которые вытекают из изменения состояния. Диаграмма состояний часто упрощает проверку правильности конструкции класса.

Для каждого состояния, которое может иметь объект класса, диаграмма показывает сообщения, которые он может принимать, действия, которые будут выполнены и состояние, в которое перейдет объект класса после этого.

**Пример:**


Диаграмма состояний деловой сущности Багаж.

***ОПИСАНИЕ ДИСЦИПЛИНЫ БИЗНЕС МОДЕЛИРОВАНИЯ***

Цели бизнес моделирования

С точки зрения RUP целями бизнес моделирования являются:

1. Описание бизнес процессов автоматизируемой организации для формирования единого их понимания со стороны заинтересованных в автоматизации организации лиц.
2. Определение проблем автоматизируемой организации и способов их решения.
3. Определение требований к автоматизированной системе организации со стороны заинтересованных лиц.
4. Понимание процесса размещения программного обеспечения в организации.

Для достижения этих целей в RUP описаны виды деятельности проектной команды при проведении бизнес моделирования, главными из которых являются разработка моделей бизнес процессов (Business Use-Case Model) и моделей анализа бизнеса (Business Analysis Model), описывающих реализации бизнес процессов. В некоторых версиях RUP модели анализа бизнеса, описывающие реализацию бизнес процессов, называются объектными моделями бизнеса (Business Object Model).

Результаты работы, полученные после проведения бизнес моделирование, являются основой для проведения работ по определению требований и разработки архитектуры автоматизированной системы.

С точки зрения RUP, наиболее значимыми артефактами, связанными с бизнес моделированием являются модели бизнес процессов (Business Use-Case Model), модели анализа бизнеса или объектные модели, описывающие реализации бизнес процессов (Business Analysis Model), а также набор документов, в котором отражены результаты бизнес моделирования.
Модели бизнес процессов описывают процессы, связанные с оказанием услуг организацией (business use case), и действующих лиц или систем, внешних по отношению к бизнес процессу (business actor). Действующие лица и системы либо инициируют бизнес процесс, либо заинтересованы в получении некоторых результатов бизнес процесса.
Модели анализа показывает, как каждый бизнес процесс реализуется некоторым набором участников бизнес процесса: работниками (business worker), действующими лицами внешними по отношению к бизнес процессу (business actor) и связанными с ними бизнес сущностями (business entity).
Для описания реализаций бизнес процессов могут использоваться следующие диаграммы языка UML:

- диаграммы деятельностей (Activity diagrams);
- диаграммы классов (Class diagrams);
- диаграммы состояний (Statechart Diagram);
- диаграммы последовательностей действий (Sequence diagrams);
- диаграммы взаимодействия (Collaboration diagrams).

Основными документами, в которых должны быть отражены результаты бизнес моделирования по RUP, явля-ются следующие документы:

- документ Оценка автоматизируемой организации (Target-Organization Assessment);
- документ Архитектура бизнеса (Business Architecture Document);
- документ Словарь терминов предметной области (Business Glossary);
- документ Бизнес правила (Business Rule);
- документ Концепция развития организации (Business Vision);
- документ Описание бизнес процесса (Business Use Case);
- документ Дополнительные требования к деятельности организации (Supplementary Business Specifications).

В RUP принято документы, модели, элементы модели называть артефактами.

## *РАЗРАБОТКА МОДЕЛЕЙ ПОТОКОВ РАБОТ*

Цель моделирование потока работ
Как указывалось выше, реализация бизнес процессов по RUP описывается с использованием различных объектных моделей бизнеса или моделей анализа бизнеса. Диаграмма деятельности (activity diagram) языка UML используется в бизнес моделировании для отображения потока работ рассматриваемого бизнес процесса. По модели потока работ определяются виды деятельности, подлежащие автоматизации, то есть бизнес требования. На основе бизнес требований в дальнейшем на этапе определения требований будут определяться функции разрабатываемой системы.

Использование диаграммы деятельности для разработки модели потока работ
Для разработки модели потока работ или, по другому, модели описания бизнес процессов должна использоваться диаграмма деятельности языка UML (activity diagram).
Для разработки модели потока работ следует использовать следующие элементы диаграммы деятельности:

- начальное состояние (start state);
- конечное состояние (end state);
- деятельность (activity);
- состояние (state);
- переход (state transition);
- решение (decision);

- горизонтальные синхронизаторы (horizontal synchronization);
- вертикальные синхронизаторы (vertical synchronization);
- разделительные линии (swimlane);
- объект (object);
- поток объектов (object flow);
- заметка.

При построении модели потока работ используется нотация диаграммы деятельности, поддерживаемая Rational Rose, которая отличается от нотации этой же диаграммы изложенной в описании UML OMG.

Диаграмма деятельности (activity diagram) должна иметь только одно начальное состояние. Конечных же состояний может существовать множество.

Новые начальные состояния могут быть только на диаграммах, декомпозирующих отдельные виды деятельности.

Элементарная «деятельность» (activity) должна использоваться для описания одного действия, например, формирование отчета сделок, передача тикетов, получение тикетов.

Нельзя в одном элементе деятельность (activity) указывать несколько видов деятельности, например, формирует отчет по сделкам и регистрирует сделки в журнале.

Если описывается автоматизированная деятельность, то следует отдельно описать деятельность пользователя системы и соответствующую ей деятельность системы, как функции действующих лиц пользователя и системы.

Элемент «состояние» (state) может использоваться для описания определенных состояний какого-либо субъекта или объекта. С этим элементом должно быть связано имя. Имя должно отражать состояние субъекта или объекта. Состояние должно именоваться в зависимости от контекста.

Переход (state transition) должен использоваться для описания связи между элементами диаграммы «деятельность» (activity), «состояние» (state). Переход (state transition) обозначается сплошной линией со стрелкой. Стрелка указывает на следующее действие или состояние.

Для отображения условий может использоваться элемент решение (decision). Элемент решение (decision) обозначается в виде ромба.

Разделительные линии (swimlane) следует использовать для разделения поля диаграммы на части, например, с целью отображения на диаграммах, ответственных за выполнение определенных действий. Пример разделительной лини (swimlane) представлен на рис..

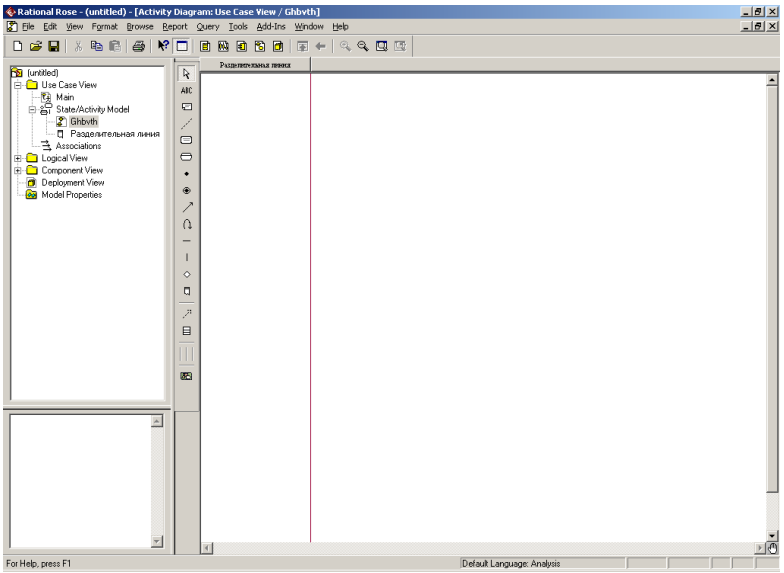
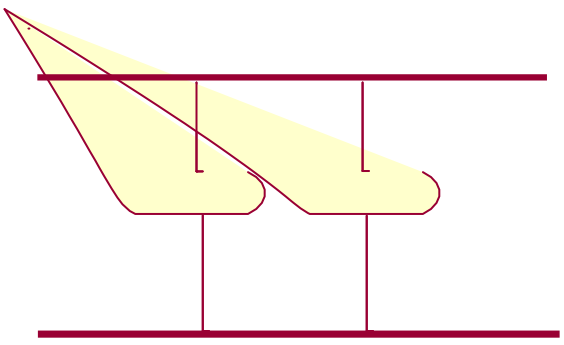



Рис. Пример разделительной линий (swimlane)

Синхронизаторы (synchronization) должны использоваться для отражения деятельностей, выполняемых параллельно или множественного выбора. Пример использования



синхронизаторов (synchronization) для отображения деятельностей, выполняемых параллельно представлен на рис.

Рис. Пример горизонтальных синхронизаторов (synchronization) для отображения деятельностей, выполняемых параллельно

В RUP существуют следующие рекомендации по разработке модели потока работ с использованием диаграммы деятельности (activity diagram).
Модель потока работ разрабатывается в два этапа.
В начале разрабатывается модель, отображающая основные виды деятельности (macro activity) по описываемому бизнес процессу в целом.
Далее каждая деятельность декомпозируется с использованием другой диаграммы деятельности. Поле этой диаграммы может разбиваться на части с использованием разделительных линий (swimlane), где разделительные линии могут представлять работников участвующих в бизнес процессе или подразделения.
Построение диаграммы с основными видами деятельности рекомендуется проводить таким образом, чтобы в дальнейшем каждому основному виду деятельности можно было сопоставить на этапе определения требований к системе модуль, компоненту или подсистему в разрабатываемой системе.
Однако для построения распределенных систем более удобным является использование следующих разделительных линий:

- входная/выходная информация;
- деятельность;
- роль;
- подразделение;
- должность;
- бизнес правило.

В разделе деятельность следует отражать шаги бизнес процесса или деятельность процесса.
В разделе входная/выходная информация – входные/выходные бизнес сущности, связанные с шагом бизнес процесса, в разделе роль – роль, ответственную за выполнение шага бизнес процесса, в разделе должность и подразделения – должности действующих лиц бизнес процесса и подразделения предприятия, связанные с шагом бизнес процесса, в разделе бизнес правила - описание бизнес правил или ссылки на модели бизнес правил для рассматриваемого шага бизнес процесса.
Такое использование разделительных линий обусловлено тем, что на основе видов деятельности, отображающих шаги бизнес процесса, будут определяться функции системы, на основе входных/выходных сущностей будет разрабатываться интерфейса пользователя, альбом входных/выходных форм, БД, классы, реализующие соответствующие функции. Информация о ролях, должностях и подразделениях будет использована при рассмотрении вопросов, связанных с разграничением доступа. На основе бизнес правил будут определяться ограничения накладываемые на функции системы.
Для изображения входной/выходной информации и роли должен использоваться элемент объект с соответствующим состоянием (object).
Для изображения шага бизнес процесса должен использоваться элемент деятельность(activity).
Для изображения подразделений, должностей, ссылок на бизнес правила – заметки (note).
Роли, входная и выходная информацию должны связываться с деятельностью через потоки объектов (object flow).
Роли, подразделения должны связываться между собой через связь - пунктирная линия (anhor note to item), прикрепляющую заметки к элементам диаграммы.
На рис. представлен пример диаграммы деятельности, используемой для декомпозиции обобщенной деятельности.

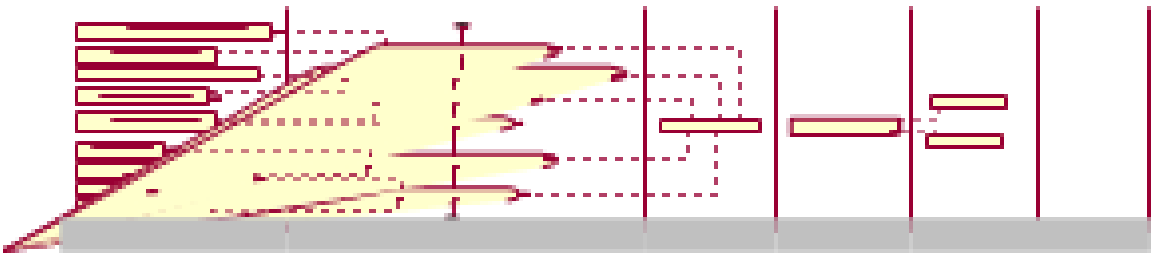


Рис. Детальное описание шага процесса кредитования Предварительное ознакомление с клиентом и его хозяйственной деятельностью и целью кредитования

Порядок построения модели потока работ бизнес процессов в Rational Rose
Порядок создания моделей с описанием потока работ в Rational Rose должен включать следующие шаги:Построение диаграммы, отображающей основные виды деятельности.

1) Построение диаграмм детализирующих основные виды деятельности.
2) Построение ссылки из диаграммы процессов на диаграмму потока работ. Построение диаграммы, отображающей основные виды деятельности

Диаграмма деятельности (activity diagram) с основными шагами бизнес процесса должна строиться как поддиаграмма соответствующего бизнес процесса, представленного на диаграмме функций (use case diagram).

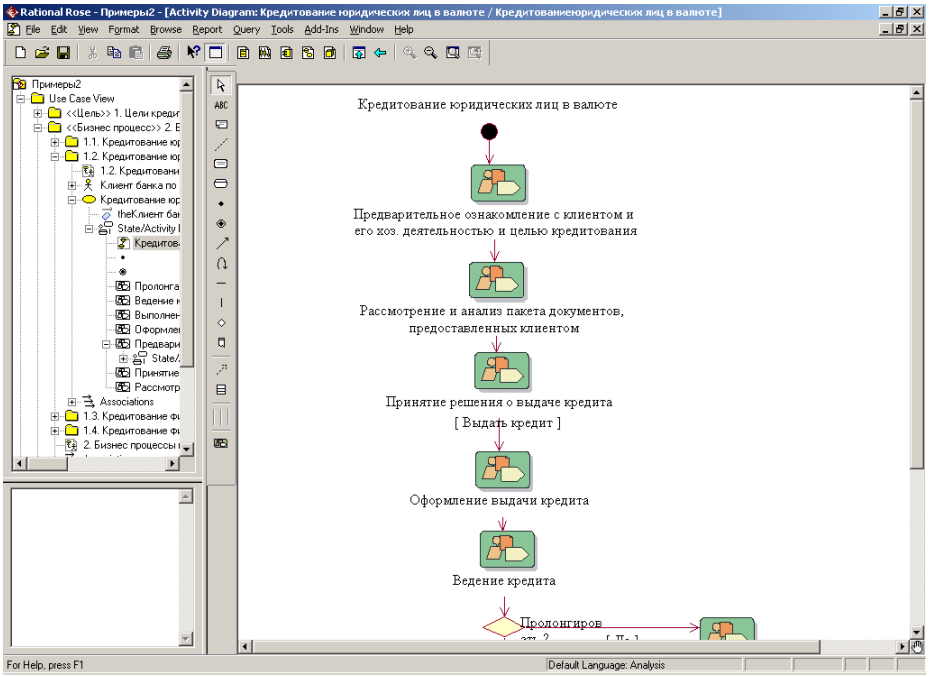


Рис. Пример изображения диаграммы с основными шагами процесса кредитования юридических лиц в валюте в Rational Rose

Построение диаграмм, детализирующих основные виды деятельности

Диаграмма деятельности (activity diagram) с детальным описанием каждого основного шага процесса должна строиться как поддиаграмма соответствующего основного шага. На рис. показано, что в окне просмотра элементов модели в Rational Rose диаграмма деятельности (activity diagram), отображающая детальное описание основного шага процесса Предварительное ознакомление с клиентом и его хоз. деятельностью и целью кредитования, находиться под соответствующим шагом Предварительное ознакомление с клиентом и его хоз. деятельностью и целью кредитования. Деятельности на этой диаграмме, которые подлежат автоматизации, отмечены цветом.

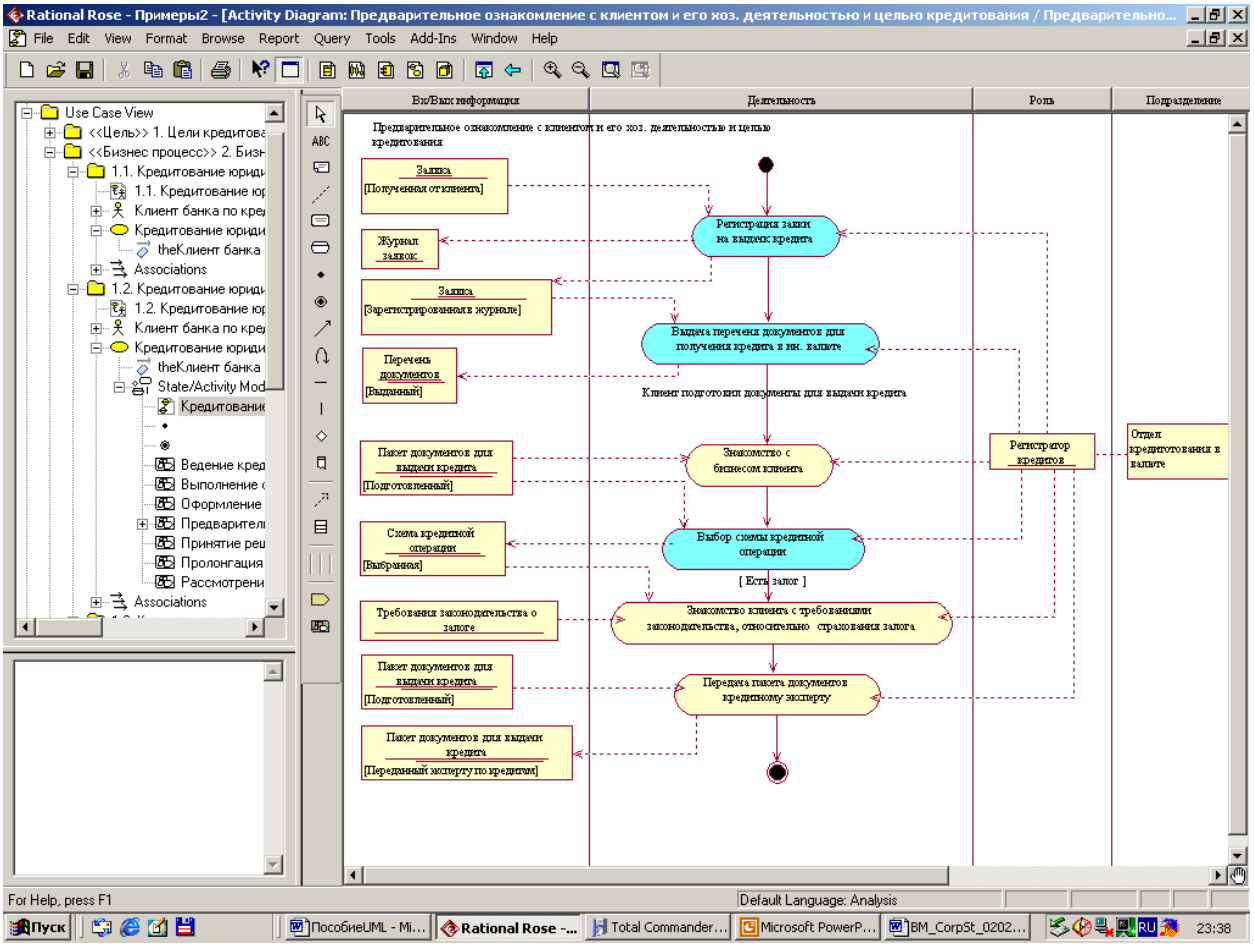



Рис. Пример изображения диаграммы с детальным описанием шага Предварительное ознакомление с клиентом и его хоз. деятельностью и целью кредитования в Rational Rose

Построение ссылки из диаграммы функций на диаграмму потока работ

Для построения ссылки на диаграмму деятельности (activity diagram) с изображением основных шагов бизнес процесса следует рядом с изображением бизнес процесса на диаграмме функций (use case diagram) поместить заметку. Прикрепить заметку к изображению бизнес процесса. Перетащить на эту заметку из окна просмотра элементов модели диаграмму деятельности (activity diagram) с изображением основных шагов бизнес процесса как представлено на рис..

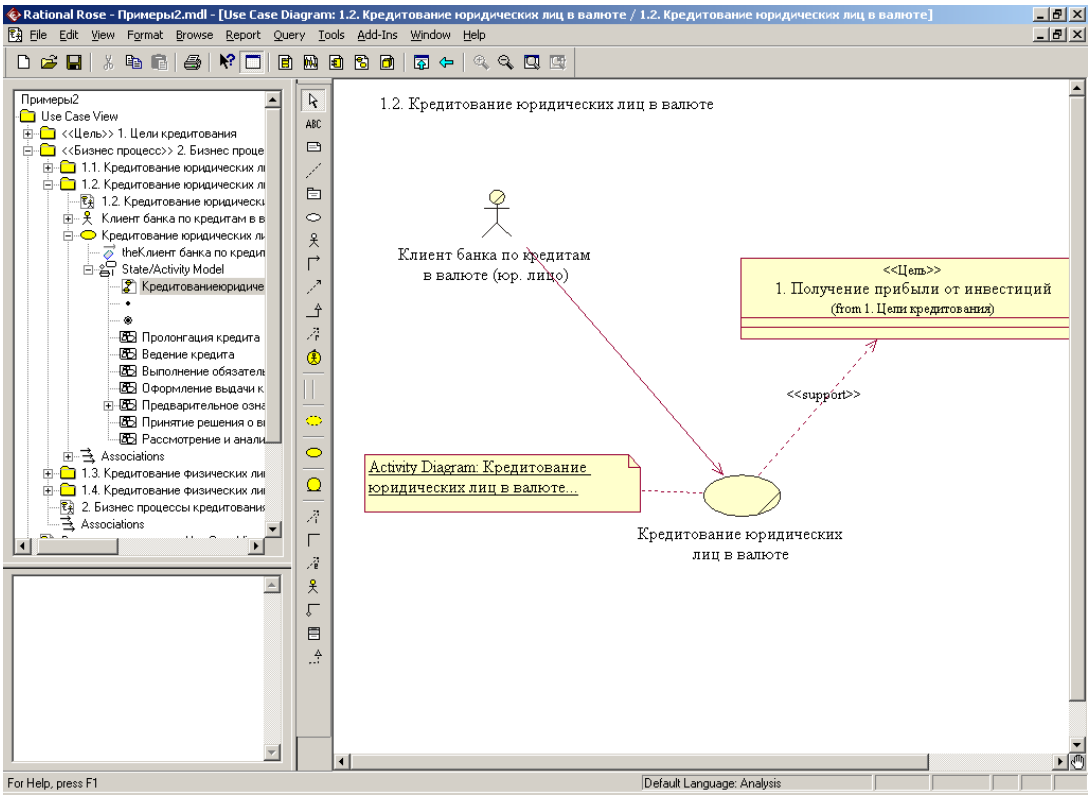



Рис. Пример диаграммы функций с изображением процесса Кредитования юридических лиц в валюте и ссылкой на диаграмму деятельности с изображением основных шагов соответствующего бизнес процесса в Rational Rose

Индивидуальные задания для лабораторной работы №24

Вариант №1

Разработать модели потока работ для информационной интернет подсистемы риэлтерской фирмы, включающей регистрацию клиента, просмотр (после регистрации) предложений, дать объявление; понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №2

Разработать модели потока работ информационной подсистемы интернет-регистрации на курсы водителей (предполагается регистрация на различные типы категорий, подача заявки и ее обработка, выбор дней и времени занятий и дней вождения); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №3

Разработать модели потока работ информационной подсистемы интернет регистрации на курсы иностранных языков (предполагается выбор языка, выбор уровня (возможное тестирование), выбор времени и дней обучения); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе

Вариант №4

Разработать модели потока работ информационной подсистемы интернет магазина (предполагается просмотр, регистрация при заказе, оформление заказа, подтверждение получения заказа); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №5
Разработать модели потока работ информационной подсистемы интернет кафе; понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №6
Разработать модели потока работ информационной подсистемы интернет-склад (предполагается просмотр интернет-клиентом ассортимента, возможность доставки, наличие на складе, оформление заявки); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №7
Разработать модели потока работ информационной подсистемы аптечной справочной службы (предполагается регистрация интернет-клиента, проверка в наличии, оформление заказа, дата доставки); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №8
Разработать модели потока работ информационной подсистемы телефонной справочной службы (предполагается регистрация интернет-клиента, проверка в наличии, оформление заказа, дата доставки); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №9
Разработать модели потока работ информационной подсистемы покупки билетов (ж/д и авиа) онлайн; понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №10
Разработать модели потока работ информационной подсистемы регистрации в аэропорту с прохождением пограничного контроля; понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №11
Разработать модели потока работ информационной подсистемы сайт кафедры ИС; понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №12
Разработать модели потока работ информационной подсистемы интернет-хранилища учебных книг свободного доступа (предполагается обязательная регистрация для скачивания, ограниченное количество скачиваний, возможность выкладывать свой материал); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №13
Разработать модели потока работ автоматизированной системы работы библиотеки (в частности рабочее место библиотекаря); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №14
Разработать модели потока работ информационной подсистемы «регистратура поликлиники» (в частности рабочее место регистратора: принятие заявок на вызова разным врачам, выдача карточек к врачам с уточнением времени работы врачей); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

Вариант №15
Разработать модели потока работ информационной веб-ориентованой подсистемы рекламного агентства (предполагается предоставление рекламных услуг, оформление заявки, подтверждение принятия заявки и сроки выполнения); понять место данной модели при определении функций разрабатываемой системы на этапе определения требований к системе.

**4.1.2. Тестирование**
3 семестр
Тема 1. Теоретические основы проектирования информационных систем
Тема 2. Методологические основы проектирования информационных систем
Тема 3. Каноническое и типовое проектирование информационных систем
Тема 4. Структурные методы анализа и проектирования информационных систем

4 семестр
Тема 5. Проектирование документальных баз данных
Тема 6. Проектирование фактографических баз данных
Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем
Тема 8. Обеспечение совместного доступа к базам данных и программам

**4.1.2.1. Порядок проведения.**

Тестирование проходит в письменной форме или с использованием компьютерных средств. Обучающийся получает определённое количество тестовых заданий. На выполнение выделяется фиксированное время в зависимости от количества заданий. Оценка выставляется в зависимости от процента правильно выполненных заданий. Ниже приведены примерные задания.

**4.1.2.2 Критерии оценивания**
**18-20 баллов** ставится, если у обучающегося:
86% правильных ответов и более.
**14-17 баллов** ставится, если у обучающегося:
От 71% до 85 % правильных ответов.
**11-13 баллов** ставится, если у обучающегося:
От 56% до 70% правильных ответов.
**0-10 баллов** ставится, если у обучающегося:
55% правильных ответов и менее.

**4.1.2.3. Содержание оценочного средства**

Вариант 1

1. Какие основные стороны участвуют в жизненном цикле программных средств?
   а) заказчик, поставщик, разработчик, оператор и персонал сопровождения
   б) заказчик, поставщик, разработчик, пользователь и администратор
   в) заказчик, поставщик, разработчик, оператор и программист
   г) заказчик, поставщик, разработчик, оператор и пользователь
2. Какие основные процессы жизненного цикла программных средств существуют?
   а) процесс заказа, процесс поставки, процесс разработки, процесс эксплуатации и процесс переноса
   б) процесс разработки, процесс эксплуатации, процесс сопровождения и процесс контроля качества
   в) процесс заказа, процесс поставки, процесс разработки, процесс эксплуатации и процесс сопровождения
   г) процесс заказа, процесс поставки, процесс разработки, процесс эксплуатации и процесс снятия с эксплуатации
3. Какой процесс охватывает перенос и снятие с эксплуатации программного продукта?
   а) процесс эксплуатации
   б) процесс решения проблемы
   в) процесс сопровождения
   г) процесс разработки

4. Сколько всего вспомогательных процессов ЖЦ?
   а) 8
   б) 9
   в) 4
   г) 5
5. Какие процессы жизненного цикла относятся к организационным процессам?
   а) процесс управления, процесс создания инфраструктуры и процесс верификации
   б) процесс решения проблемы, процесс обучения и процесс усовершенствования
   в) процесс управления, процесс создания инфраструктуры и процесс обучения
   г) процесс управления, процесс усовершенствования и процесс аудита
6. Какие процессы применяются на предприятии для создания и реализации основной структуры?
   а) основные
   б) вспомогательные
   в) организационные
7. Какие принципы лежат в основе процессов жизненного цикла информационной системы?
   а) модульность и собственность
   б) эффективность и результативность

в) гибкость и адаптивность
г) иерархичность и последовательность

**8.** Какой стадии НЕТ в жизненном цикле информационных систем?
а) проектирование.
б) апробация
в) создание
г) ввод в эксплуатацию

**9.** Какой принцип ЖЦ можно описать словами: максимальная слаженность функций процесса и минимальная связь между процессами?
а) собственность
б) определенность
в) модульность

**10.** Что такое модель жизненного цикла информационной системы (ИС)?
а) список всех ИС на рынке
б) набор этапов для разработки, внедрения и обслуживания ИС
в) результат тестирования ИС
г) учет расходов на разработку ИС

**11.** Какой подход фокусируется на создании работающего прототипа для получения отзывов от пользователей?
а) agile
б) экстремальное программирование
в) модель быстрого прототипирования
г) scrum

**12.** Какой из следующих методов включает в себя частые итерации разработки, известные как спринты?
а) agile
б) модель быстрого прототипирования
в) экстремальное программирование
г) scrum

**13.** Что является основным преимуществом итеративной разработки?
а) необходимость планирования на длительный срок
б) снижение рисков и возможность получения ранней обратной связи
в) фиксация всех требований изначально
г) сложность в управлении проектом

**14.** Что из перечисленного относится к практике экстремального программирования (XP)?
а) ожидание завершения всех требований
б) непрерывная интеграция кода
в) использование исключительно ручного тестирования
г) отсутствие общения с пользователями

**15.** Какой методологии разработки программного обеспечения характерны принципы гибкости и адаптивности?
а) водопадная модель
б) agile
в) модель быстрого прототипирования
г) тестирование перед разработкой

**16.** Какой из принципов итеративной разработки позволяет вносить изменения на основе полученной обратной связи?
а) гибкость
б) устойчивость
в) изоляция этапов
г) подробное тестирование в конце

**17.** Что означает термин «рефакторинг» в контексте разработки?
а) удаление ненужного кода
б) изменение требований
в) упрощение и улучшение кода без изменения его функциональности
г) завершение проекта

**18.** Какой подход к разработке позволяет улучшить качество продукта благодаря постоянному тестированию?
а) водопадная модель
б) Scrum
в) итеративная разработка
г) модель быстрой реализации

**19.** Упорядочите в порядке следования этапы разработки информационной системы:
   а) определяются сроки и ресурсы
   б) разрабатывается код
   в) распределяются роли в команде
   г) оценивается успешность итерации

**20.** На каком этапе производится разработка и тестирование кода?
   а) оценка
   б) проектирование
   в) развертывание
   г) планирование

Ответы: 1-а; 2-в; 3-в; 4-а; 5-в; 6-а; 7-а; 8-б; 9-в; 10-б; 11-в; 12-г; 13-б; 14-б; 15-б; 16-а; 17-в; 18-в; 19- вабг; 20-б.

Вариант 2

**1.** Что происходит на этапе развертывания?
   а) внесение корректив в проект
   б) новый функционал становится доступным пользователям
   в) оценка результатов итерации
   г) определение новых задач

**2.** Что означает оценка в итеративном процессе?
   а) планирование следующей итерации на основе обратной связи
   б) реализация всех задач проекта
   в) разработка новых функций
   г) установка дополнительных ресурсов

**3.** Какая роль в Scrum отвечает за бизнес-требования?
   а) участник команды
   б) Scrum Master
   в) Product Owner
   г) разработчик

**4.** Какой элемент Scrum включает в себя планирование задач на спринт?
   а) демонстрация
   б) ретроспектива
   в) спринт планирование
   г) ежедневные совещания

**5.** Гибридные модели жизненного цикла позволяют:
   а) комбинировать элементы различных методологий для достижения лучших результатов
   б) исключить необходимость в тестировании и проверке продукта
   в) применять гибкие подходы в строгих рамках традиционного управления проектами
   г) упрощать процесс разработки за счет полного соблюдения одной конкретной модели

**6.** Выстройте иерархию от самой важной модели жизненного цикла к менее важной:
   а) внешние факторы
   б) опыт команды
   в) мода на подходы
   г) наличие финансирования

**7.** Что является одним из ключевых преимуществ итеративной разработки?
   а) долгосрочное планирование
   б) быстрая адаптация к меняющимся требованиям
   в) невозможность внесения изменений
   г) увеличение количества сотрудников

**8.** Что характеризует архитектурный стиль?
   а) сходство в подходах к реализации поставленных задач
   б) стандартные языки описания архитектур
   в) жесткую иерархию компонентов системы
   г) технологическую реализацию системы

**9.** На сколько групп подразделяются архитектурные стили?
   а) 3
   б) 4
   в) 5
   г) 6

**10.** К какой группе архитектурных стилей относятся системы пакетно-последовательной

обработки и «конвейеры и фильтры»?

а) потоки данных
б) вызов с возвратом
в) независимые компоненты
г) централизованные данные

**11.** Что характеризует архитектуру «программа-сопрограмма»?

а) наличие главной управляющей программы и сопрограмм
б) параллельное функционирование основной программы и сопрограмм
в) работа основной программы как диспетчера процессов
г) все перечисленное выше

**12.** Что является преимуществом объектно-ориентированных систем?

а) возможность вносить изменения в объект без уведомления конечных пользователей
б) открытость реализации объектов
в) полное отсутствие связей между объектами
г) простота разработки и поддержки

**13.** Какой тип архитектурных стилей характеризуется наличием произвольного количества уровней вложенности?

а) потоки данных
б) вызов с возвратом
в) независимые компоненты
г) централизованные данные

**14.** Какой архитектурный стиль предполагает наличие отдельных модулей, выполняющих различные процессы?

а) системы пакетно-последовательной обработки
б) «конвейеры и фильтры»
в) «программа-сопрограмма»
г) объектно-ориентированные системы

**15.** Какая особенность характеризует архитектуру «конвейеры и фильтры»?

а) линейная структура модулей
б) результаты выполнения модулей передаются в разные модули
в) отсутствие обратных связей
г) все модули выполняют один и тот же процесс

**16.** Что является примером реализации архитектуры «конвейеры и фильтры»?

а) бухгалтерская информационная система
б) компилятор
в) система управления базами данных
г) офисное приложение

**17.** Какие типы систем являются частным случаем архитектуры «программа-сопрограмма»?

а) клиент-серверные системы
б) иерархические многоуровневые системы
в) объектно-ориентированные системы
г) все перечисленные

**18.** Что является недостатком объектно-ориентированных систем?

а) необходимость знать расположение вызываемого объекта
б) невозможность вносить изменения в объекты
в) сложность разработки и поддержки
г) отсутствие инкапсуляции данных

**19.** Какая особенность характеризует клиент-серверные системы?

а) синхронность программных архитектур
б) асинхронность программных архитектур
в) отсутствие уровней вложенности
г) отсутствие взаимодействия между компонентами

**20.** Что подразумевает архитектура «ведущий-ведомый»?

а) независимость модулей друг от друга
б) полную децентрализацию
в) наличие основной управляющей программы и сопрограмм
г) отсутствие взаимосвязей между компонентами

Ответы: 1-б; 2-а; 3-в; 4-в; 5-а, в; 6- бгав; 7-б; 8-а; 9-в; 10-а; 11-а; 12-а; 13-б; 14-б; 15-б; 16-б; 17-в; 18-а; 19-а; 20-в

**4.1.3. Реферат**
3 семестр
Тема 1. Теоретические основы проектирования информационных систем
Тема 2. Методологические основы проектирования информационных систем
Тема 3. Каноническое и типовое проектирование информационных систем
Тема 4. Структурные методы анализа и проектирования информационных систем
4 семестр
Тема 5. Проектирование документальных баз данных
Тема 6. Проектирование фактографических баз данных
Тема 7. Объектно-ориентированные методы анализа и проектирования информационных систем
Тема 8. Обеспечение совместного доступа к базам данных и программам

**4.1.3.1. Порядок проведения.**

Обучающиеся самостоятельно пишут работу на заданную тему и сдают преподавателю в письменном виде. В работе производится обзор материала в определённой тематической области либо предлагается собственное решение определённой теоретической или практической проблемы. Оцениваются проработка источников, изложение материала, формулировка выводов, соблюдение требований к структуре и оформлению работы, своевременность выполнения. В случае публичной защиты реферата оцениваются также ораторские способности.
Требования к реферату
При оформлении текста реферата следует придерживаться следующих параметров:
поля: левое – 35 мм, правое – 15 мм, верхнее – 25 мм, нижнее – 25 мм;
ориентация страницы: книжная;
шрифт: TimesNewRoman;
кегль: 14 пт (пунктов);
красная строка:1 мм;
междустрочный интервал: полуторный;
выравнивание основного текста и сносок: по ширине.
Иллюстрации в виде рисунков, фотоснимков, схем и т.п. могут располагаться органично с текстом (возможно ближе к иллюстрируемой части) либо на отдельных листах. В любом случае выполняется нумерация (сквозная для всех разделов), которая располагается вверху. Подрисуночную нумерацию и надпись располагать внизу.
Заканчивается пояснительная записка библиографическим списком источников, к которым обращался студент во время работы над разрабатываемой темой.
Объем информационно-технологической документации не регламентируется – он диктуется достаточностью для практического применения. Карточки задания для самоконтроля (если таковы имеются) вкладываются в прозрачные файлы.
Реферат по своему структурному содержанию должен содержать следующие элементы:
- титульный лист;
- содержание;
- введение;
- базовое понятия;
- историческая справка (особенности зарождения и развития, основоположники и т.д.);
- классификация (виды, формы и т.д.);
- общее и частное положения по применению в учебно-воспитательном процессе;
- глоссарий;
- список использованных источников
- приложения

**4.1.3.2 Критерии оценивания**
**17-20 баллов ставится, если обучающийся:**
Продемонстрировал высокий уровень знаний и умений, необходимых для выполнения задания. Работа полностью соответствует требованиям профессиональной деятельности. Отличная способность применять имеющиеся знания и умения для решения практических задач. Высокий уровень креативности, самостоятельности. Соответствие выбранных методов поставленным задачам.
**2114-16 баллов ставится, если обучающийся:**
Продемонстрировал средний уровень знаний и умений, необходимых для выполнения задания. Работа в основном соответствует требованиям профессиональной деятельности. Хорошая способность применять имеющиеся знания и умения для решения практических задач. Средний уровень креативности, самостоятельности. Выбранные методы в целом соответствуют поставленным задачам.
**11-15 баллов ставится, если обучающийся:**
Продемонстрировал низкий уровень знаний и умений, необходимых для выполнения задания. Работа частично соответствует требованиям профессиональной деятельности. Удовлетворительная способность применять

имеющиеся знания и умения для решения практических задач. Низкий уровень креативности, самостоятельности. Выбранные методы частично соответствуют поставленным задачам.

**0-10 баллов ставится, если обучающийся:**

Продемонстрировал неудовлетворительный уровень знаний и умений, необходимых для выполнения задания. Работа не соответствует требованиям профессиональной деятельности. Неудовлетворительная способность применять имеющиеся знания и умения для решения практических задач. Недостаточный уровень креативности, самостоятельности. Выбранные методы не соответствуют поставленным задачам.

**4.1.3.3. Содержание оценочного средства**

Примерные темы:

1. Проектирование подсистемы регистрации командировочных удостоверений в информационной системе.
2. Проектирование ИС автотранспортного предприятия
3. Проектирование АС учета договоров и контроля за их исполнением
4. Проектирование АС учета и оптимизации транспортных расходов на предприятии
5. Проектирование АС учета сдельной оплаты труда
6. Проектирование АРМ экономиста по прогнозу закупок на предприятии оптовой торговли
7. Проектирование ИС поддержки биржевых торгов
8. Проектирование АС учета материальных ресурсов предприятия
9. Проектирование подсистемы автоматизации складского учета
10. Проектирование подсистемы автоматизации учета платежей по договорам
11. Проектирование системы автоматизации учета поступления и реализации товаров в розничной торговле
12. Проектирование подсистемы учета реализации товаров в оптовой торговле
13. Проектирование системы автоматизации кассовых операций торгового предприятия
14. Проектирование системы автоматизации учета выбытия денежных средств с расчетного счета организации
15. Проектирование системы автоматизации учета повременно-премиальной оплаты труда в организации
16. Проектирование системы автоматизации учета поступления и выбытия малоценных и быстроизнашивающихся предметов в коммерческой организации
17. Проектирование системы автоматизации учета поступления и выбытия, основных средств на предприятии
18. Проектирование АС учета обмена валют
19. Проектирование АС учета запасов предприятия".

**4.2. Оценочные средства промежуточной аттестации**

По дисциплине предусмотрен зачет в 3 семестре, экзамен в 4 семестре. Экзамен / зачет проходит по билетам. В каждом билете два теоретических вопроса. Экзамен / зачет проводится в устной / письменной и компьютерной форме. Оценивается владение материалом, его системное освоение, способность применять нужные знания, навыки и умения при анализе проблемных ситуаций.

**4.2.1. Устный или письменный ответ на вопрос**

**4.2.1.1. Порядок проведения.**

Устный или письменный ответ на вопрос направлен на проверку знаний основных разделов по дисциплине «Проектирование информационных систем на транспорте».

**4.2.1.2. Критерии оценивания.**

**43-50 баллов ставится, если обучающийся:**

В ответе качественно раскрыл содержание темы. Ответ хорошо структурирован. Прекрасно освоен понятийный аппарат. Продемонстрирован высокий уровень понимания материала. Превосходное умение формулировать свои мысли, обсуждать дискуссионные положения.

**36-42 баллов ставится, если обучающийся:**

Основные вопросы темы раскрыл. Структура ответа в целом адекватна теме. Хорошо освоен понятийный аппарат. Продемонстрирован хороший уровень понимания материала. Хорошее умение формулировать свои мысли, обсуждать дискуссионные положения.

**28-35 баллов ставится, если обучающийся:**

Тему частично раскрыл. Ответ слабо структурирован. Понятийный аппарат освоен частично. Понимание отдельных положений из материала по теме. Удовлетворительное умение формулировать свои мысли, обсуждать дискуссионные положения.

**0-27 баллов ставится, если обучающийся:**

Тему не раскрыл. Понятийный аппарат освоен неудовлетворительно. Понимание материала фрагментарное или отсутствует. Неумение формулировать свои мысли, обсуждать дискуссионные положения.

**4.2.1.3. Оценочные средства.**

**Вопросы для устного или письменного ответа**

**3 семестр**

1. Проектирование ИС, проект ИС, основные подходы в проектировании ИС.
2. Методология построения ИС.

3. Основные компоненты технологии проектирования ИС.
4. Методы проектирования ИС.
5. Средства проектирования ИС.
6. Краткая характеристика применяемых технологий проектирования.
7. Жизненный цикл ИС. Виды моделей жизненного цикла ИС.
8. Требования к выбираемой технологии проектирования.
9. Каноническое проектирование. Стадии и этапы.
10. Состав и содержание работ на предпроектной стадии создания ИС.

**4 семестр**

1. Технология параметрически-ориентированного проектирования.
2. Технология модельно-ориентированного проектирования.
3. Основные понятия и классификация CASE-технологий.
4. Проектирование фактографических БД: методы проектирования.
5. Проектирование документальных баз данных.
6. Концептуальное проектирование БД.
7. Логическое проектирование БД.
8. Физическое проектирование БД.
9. Базовые принципы и понятия технологии разработки объектно-ориентированных информационных систем на основе UML.
10. Возможности и достоинства UML. Инструментальные средства визуального моделирования.

Краткие ответы:

**3 семестр**

**1. Проектирование ИС, проект ИС, основные подходы в проектировании ИС.**

Проектирование информационных систем (ИС) — это процесс преобразования входной информации об объекте проектирования, о методах проектирования и об опыте проектирования объектов аналогичного назначения в соответствии со стандартами в проект ИС.

Проект ИС охватывает три основные области:

Проектирование объектов данных, которые будут реализованы в базе данных.

Проектирование программ, экранных форм, отчётов, которые будут обеспечивать выполнение запросов к данным.

Учёт конкретной среды или технологии: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределённой обработки данных и т.п.

Основные подходы в проектировании ИС:

Структурный подход. Требует синтезировать варианты системы из компонентов (блоков) и оценивать варианты при их частичном переборе с предварительным прогнозированием характеристик компонентов.

Блочно-иерархический подход. Использует идеи декомпозиции сложных описаний объектов и средств их создания на иерархические уровни и аспекты, вводит понятие стиля проектирования (восходящее и нисходящее), устанавливает связь между параметрами соседних иерархических уровней. 1

Объектно-ориентированный подход. Имеет ряд важных структурных принципов, используемых при разработке информационных систем, и преимуществ в решении проблем управления сложностью и интеграции ПО.

**2. Методология построения ИС.**

Методология построения информационных систем (ИС) заключается в организации процесса построения системы и обеспечении управления этим процессом для того, чтобы гарантировать выполнение требований как к самой системе, так и к характеристикам процесса разработки.

Типовая методология построения информационных систем содержит три основных компонента:

Набор моделей для описания требований к ИС, проектных и программных решений. Каждая модель обычно содержит как определение конструкций (нотацию), так и правила их использования (синтаксис).

Методика применения набора моделей для построения информационной системы. Методика обычно использует фиксированный набор моделей и определяет последовательность их построения для описания различных аспектов создаваемой системы.

Процесс организации проектных работ. Включает различные технологии — планирования, управления проектом, контроля качества и т. д.

Некоторые особенности одной из методологий построения ИС:

Итерационная спиральная модель жизненного цикла ИС. Методология описывает процесс создания и сопровождения информационных систем в виде последовательности стадий, каждая из которых разбита на этапы, и выполняемых на них процессов. Для каждого этапа определяются последовательность выполняемых работ,

получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т.д.

Комплекс развивающихся систем согласованных моделей. Методология определяет процесс создания корпоративных информационных систем как процесс построения и последовательного развития систем согласованных моделей, начиная от системы моделей, описывающих деятельность организации, и заканчивая готовой информационной системой.

Методология анализа ИС на основе бизнес-процессов. Отправной точкой процесса создания ИС являются модели бизнес-процессов, протекающих в организации и реализующих её цели и задачи.

Методология проектирования от данных. Процесс проектирования основан на извлечении всех данных из моделей бизнес-процессов, построении и развитии моделей данных (концептуальной модели данных, модели архитектуры ИС, полной реляционной модели данных и т.д.).

Комплекс согласованных инструментальных средств. Реализация методологии базируется на применении комплекса согласованных между собой инструментальных средств, обеспечивающих высокий уровень автоматизации всех процессов, выполняемых в соответствии с методологией на протяжении жизненного цикла ИС.

**3. Основные компоненты технологии проектирования ИС.**

Основные компоненты технологии проектирования информационных систем (ИС):

Методология. Предлагает принципы проектирования, определяет общие подходы к оценке и выбору варианта системы, последовательность стадий и этапов проектирования, а также позволяет выбрать метод проектирования.

Метод проектирования. Конкретизирует порядок разработки отдельных элементов, комплексов задач, подсистем и системы в целом.

Инструментальные средства проектирования. Поддерживают метод проектирования.

Кроме того, к основным компонентам технологии проектирования относят методы и средства организации проектирования (управление процессом создания или модернизации проекта ИС).

Также к средствам проектирования относят нормативно-правовые и нормативно-технические документы, системы классификации и кодирования информации, системы документации, модели входных и выходных потоков информации и методики их анализа и другие.

**4. Методы проектирования ИС.**

Методы проектирования информационных систем (ИС) можно классифицировать по степени использования средств автоматизации, типовых проектных решений и адаптивности к предполагаемым изменениям.

По степени автоматизации методы проектирования разделяются на:

Ручное. Проектирование компонентов ИС осуществляется без использования специальных инструментальных программных средств, а программирование — на алгоритмических языках.

Автоматизированное. Производится генерация или конфигурирование (настройка) проектных решений на основе использования специальных инструментальных программных средств (CASE-средств).

По степени использования типовых проектных решений различают:

Оригинальное (индивидуальное). Проектные решения разрабатываются «с нуля» в соответствии с требованиями к ИС.

Типовое. Предполагает конфигурирование ИС из готовых типовых проектных решений (программных модулей).

По степени адаптивности проектных решений к предполагаемым изменениям выделяют методы:

Реконструкции. Адаптация проектных решений выполняется путём переработки соответствующих компонентов (перепрограммирования программных модулей).

Параметризации. Проектные решения настраиваются (генерируются) в соответствии с изменяемыми параметрами.

Реструктуризации модели. Изменяется модель проблемной области, на основе которой автоматически заново генерируются проектные решения.

Также по использованию подхода (методологии) к проектированию различают структурный и объектный подходы:

Структурный подход предполагает раздельное построение модели функций (чаще всего диаграммы потоков данных) и модели данных (чаще всего диаграммы «сущность — связь»).

Объектный подход содержит набор моделей, связанных с понятием класса/объекта, объединяющего данные (состояние) и поведение.

**5. Средства проектирования ИС.**

Средства проектирования информационных систем (ИС) — это комплекс инструментальных средств, обеспечивающих в рамках выбранной методологии проектирования поддержку полного жизненного цикла ИС, который включает в себя стратегическое планирование, анализ, проектирование, реализацию, внедрение и эксплуатацию.

Некоторые средства проектирования ИС:

CASE-средства. Программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы.

Средства моделирования. В рамках структурного подхода к проектированию ИС используются, например: диаграмма потоков данных (модель бизнес-процессов), диаграмма «сущность — связь», диаграмма переходов состояний, структурная карта, диаграмма вариантов использования, блок-схема (алгоритмы выполнения процедур).

Типовые проектные решения. Предполагают сборку или конфигурацию ИС из готовых типовых компонентов. К ним относятся элементные (по задаче или по отдельному виду обеспечения задачи), подсистемные (отдельные подсистемы, разработанные с учётом функциональной полноты и минимизации внешних информационных связей) и объектные (типовые отраслевые проекты, которые включают полный набор функциональных и обеспечивающих подсистем ИС).

**6. Краткая характеристика применяемых технологий проектирования.**

Некоторые применяемые технологии проектирования и их краткая характеристика:

Каноническое проектирование. Ручная технология индивидуального проектирования, осуществляемого на уровне исполнителей без использования каких-либо инструментальных средств. Область применения — небольшие локальные системы.

Типовое проектирование. Предполагает создание системы из готовых типовых элементов. Каждое типовое проектное решение (ТПР) предполагает наличие документации с детальным описанием ТПР и процедур настройки в соответствии с требованиями разрабатываемой системы.

Параметрически-ориентированное проектирование. Включает определение критериев оценки пригодности пакетов прикладных программ для решения поставленных задач, анализ и оценку доступных программ, выбор и закупку наиболее подходящего пакета, настройку параметров (доработку) закупленного пакета.

Системы автоматизированного проектирования (САПР). Это несколько самодостаточных программ с уникальным функционалом, которые интегрированы друг с другом для комплексной проектной работы. САПР построена на информационных технологиях, возможностях искусственного интеллекта (математическое программирование, дискретная, вычислительная математика, статистика, анализ, CASE-технологии и т. п.).

**7. Жизненный цикл ИС. Виды моделей жизненного цикла ИС.**

Жизненный цикл информационной системы (ИС) — это совокупность стадий и этапов, которые проходит система в своём развитии от момента принятия решения о создании системы до момента прекращения её функционирования.

Стадии жизненного цикла ИС:

Планирование и анализ требований (предпроектная стадия). Исследование и анализ существующей информационной системы, определение требований к создаваемой ИС, оформление технико-экономического обоснования и технического задания на разработку ИС.

Проектирование (техническое проектирование, логическое проектирование). Разработка в соответствии со сформулированными требованиями состава автоматизируемых функций и состава обеспечивающих подсистем, оформление технического проекта ИС.

Реализация (рабочее проектирование, физическое проектирование, программирование). Разработка и настройка программ, наполнение баз данных, создание рабочих инструкций для персонала, оформление рабочего проекта.

Внедрение (тестирование, опытная эксплуатация). Комплексная отладка подсистем ИС, обучение персонала, поэтапное внедрение ИС в эксплуатацию по подразделениям объекта, оформление акта о приёмо-сдаточных испытаниях ИС.

Эксплуатация ИС (сопровождение, модернизация). Сбор рекламаций и статистики о функционировании ИС, исправление ошибок и недоработок, оформление требований к модернизации ИС и её выполнение.

Виды моделей жизненного цикла ИС:

Каскадная модель. Предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе.

Поэтапная модель с промежуточным контролем. Разработка ИС ведётся итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать взаимовлияние результатов разработки на различных этапах.

Спиральная модель. На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка. Особое внимание уделяется начальным этапам разработки — анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования).

**8. Требования к выбираемой технологии проектирования.**

Основные требования к выбираемой технологии проектирования:

Соответствие требованиям заказчика. Созданный с помощью технологии проект должен соответствовать требованиям заказчика.

Максимальное отражение всех этапов жизненного цикла проекта.

Обеспечение минимальных трудовых и стоимостных затрат на проектирование и сопровождение проекта. 15

Наличие связи между проектированием и сопровождением проекта.

Способствование росту производительности труда проектировщика.

Обеспечение надёжности процесса проектирования и эксплуатации проекта.

Простота ведения проектной документации.

Также технология проектирования должна обеспечивать выпуск проектной продукции высокого качества с наименьшими затратами труда, времени, финансовых и материально-технических ресурсов.

**9. Каноническое проектирование. Стадии и этапы.**

Каноническое проектирование — ручная технология индивидуального проектирования, которая применяется для небольших локальных информационных систем (ИС). В её основе лежит каскадная модель жизненного цикла ИС.

Стадии и этапы канонического проектирования:

Стадия 1. Формирование требований к ИС. Обследование объекта и обоснование необходимости создания ИС, формирование требований пользователей к ней, оформление отчёта о выполненной работе и тактико-технического задания на разработку.

Стадия 2. Разработка концепции ИС. Изучение объекта автоматизации, разработка вариантов концепции ИС, удовлетворяющих требованиям пользователей, оформление отчёта и утверждение концепции.

Стадия 3. Техническое задание. Разработка и утверждение технического задания на создание ИС.

Стадия 4. Эскизный проект. Разработка предварительных проектных решений по системе и её частям, эскизной документации на ИС и её части.

Стадия 5. Технический проект. Разработка проектных решений по системе и её частям, документации на ИС и её части, документации на поставку комплектующих изделий.

Стадия 6. Рабочая документация. Разработка и адаптация программного обеспечения, рабочей документации.

Стадия 7. Ввод в действие. Подготовка объекта автоматизации, подготовка персонала, комплектация ИС поставляемыми изделиями, строительно-монтажные и пусконаладочные работы, проведение предварительных, опытной и приёмочной эксплуатации.

Стадия 8. Сопровождение ИС. Выполнение работ в соответствии с гарантийными обязательствами, послегарантийное обслуживание.

**10. Состав и содержание работ на предпроектной стадии создания ИС.**

Состав и содержание работ на предпроектной стадии создания ИС включают:

Предварительное изучение предметной области. Разработчики уточняют границы системы, определяют круг пользователей будущей ИС различных уровней и выделяют классы и типы объектов, подлежащих обследованию и последующей автоматизации.

Выбор технологии проектирования.

Выбор метода проведения обследования.

Выбор метода сбора материалов обследования.

Разработка программы обследования. В ходе неё изучают структуру и бизнес-процессы организации, модель управления, задачи, подлежащие автоматизации, технико-экономические характеристики, ориентировочный состав технических средств.

Разработка плана-графика сбора материалов обследования.

Сбор и формализацию материалов обследования. В процессе этого этапа проектировщики интервьюируют специалистов подразделений изучаемой предметной области, собирают сведения обо всех объектах обследования, в том числе о предприятии в целом, функциях управления, методах и алгоритмах реализации функций. Также собирают сведения о составе обрабатываемых и рассчитываемых показателей, формы документов, отражающих хозяйственные процессы, используемые классификаторы, макеты файлов, сведения об используемых технических средствах и технологиях обработки данных. По окончании этапа вместе с пользователем контролируют правильность собранных материалов и формируют отчёт об обследовании.

По результатам обследования формируется техническое задание на информационную систему.

**4 семестр**

**1. Технология параметрически-ориентированного проектирования.**

Параметрически-ориентированное проектирование — это тип проектирования, который использует математические модели для создания и модификации объектов, материалов и структур.

Технология включает следующие этапы:

Определение критериев оценки пригодности пакетов прикладных программ (ППП) для решения поставленных задач.

Анализ и оценка доступных ППП по сформулированным критериям.

Выбор и закупка наиболее подходящего пакета.

Настройка параметров (доработка) закупленного ППП.

Некоторые области применения параметрически-ориентированного проектирования:

Архитектура. Создание зданий различных форм и размеров, которые более эффективны и экономичны по сравнению с традиционными проектами.

Инженерное дело. Создание объектов, материалов и конструкций, которые более эффективны и экономичны, чем традиционные.

Компьютерная графика. Создание моделей персонажей, автомобилей, деревьев и зданий, которые являются более точными и экономичными, чем традиционные дизайны.

Физика. Создание моделей молекул, звёзд, планет и галактик, которые являются более точными и экономичными, чем традиционные конструкции.

**2. Технология модельно-ориентированного проектирования**.

Модельно-ориентированное проектирование (МОП) — это математический и визуальный метод решения задач, связанных с проектированием систем управления, обработки сигналов и связи.

Основная цель применения МОП — учёт взаимного влияния отдельных компонентов в динамике при разработке целостной системы. Это позволяет оптимально настроить параметры системы для выполнения поставленных задач, провести испытания во всех возможных режимах эксплуатации изделия, а также исполнить целевой бюджет проекта при достижении нужного качества изделия в приемлемые сроки.

Основные этапы МОП:

Построение модели объекта. Построение модели может быть эмпирическим и теоретическим. При эмпирическом построении модели собираются и обрабатываются исходные данные, полученные от реальной системы, и некоторый алгоритм используется для определения математической модели объекта. При теоретическом моделировании строятся блок-схемы модели, которые реализуют известные дифференциально-алгебраические уравнения, описывающие динамику объекта.

Анализ и построение системы управления. Математическая модель, сконструированная на шаге 1, используется для определения динамических характеристик модели объекта. На основе этих характеристик строится система управления.

Офлайн-моделирование и моделирование в реальном времени. Время отклика динамической системы на входные данные, изменяющиеся во времени, исследуется с помощью симуляции модели. Симуляция позволяет немедленно найти характеристики модели, требования, накладываемые на неё, и ошибки построения до начала проектирования.

Реализация регулятора. В идеале это делается с помощью автоматической генерации кода из системы управления, полученной на шаге 2.

МОП предоставляет общую среду разработки, что способствует взаимодействию группы разработчиков в процессе анализа данных и проверки системы. Инженеры могут найти и исправить ошибки на ранних стадиях проектирования системы, когда время и финансовые последствия изменения системы сводятся к минимуму. МОП способствует повторному использованию модели для улучшения системы и создания производных систем с расширенными возможностями.

**3. Основные понятия и классификация CASE-технологий.**

Основные понятия CASE-технологий:

Методология определяет шаги и этапность реализации проекта, а также правила использования методов, с помощью которых разрабатывается проект.

Метод — это процедура или техника генерации описаний компонентов ЭИС (например, проектирование потоков и структур данных).

Нотация — отображение структуры системы, элементов данных, этапов обработки с помощью специальных графических символов диаграмм, а также описание проекта системы на формальных и естественных языках.

Инструментальные средства CASE — специальные программы, которые поддерживают одну или несколько методологий анализа и проектирования ИС.

Репозиторий (словарь данных) — специализированная база данных, предназначенная для отображения состояния проектируемой ЭИС в каждый момент времени.

Классификация CASE-технологий:

По поддерживаемым методологиям проектирования: функционально (структурно) ориентированные, объектно-ориентированные и комплексно-ориентированные (набор методологий проектирования).

По поддерживаемым графическим нотациям построения диаграмм: с фиксированной нотацией, с отдельными нотациями и наиболее распространёнными нотациями.

По степени интегрированности: tools (отдельные локальные средства), toolkit (набор неинтегрированных средств, охватывающих большинство этапов разработки ЭИС) и workbench (полностью интегрированные средства, связанные общей базой проектных данных — репозиторием).

По типу и архитектуре вычислительной техники: ориентированные на ПЭВМ, ориентированные на локальную вычислительную сеть (ЛВС), ориентированные на глобальную вычислительную сеть (ГВС) и смешанного типа.

По режиму коллективной разработки проекта: не поддерживающие коллективную разработку, ориентированные на режим реального времени разработки проекта, ориентированные на режим объединения подпроектов.

По типу операционной системы (ОС): работающие под управлением WINDOWS 3.11 и выше, работающие под управлением UNIX и работающие под управлением различных ОС (WINDOWS, UNIX, OS/2 и др.).

**4. Проектирование фактографических БД: методы проектирования.**

Некоторые методы проектирования фактографических баз данных (БД):

Концептуальное проектирование. Сбор, анализ и редактирование требований к данным. Для этого осуществляются обследование предметной области, изучение её информационной структуры, выявление всех фрагментов с пользовательскими представлениями, информационными объектами и связями между ними, моделирование и интеграция всех представлений. По окончании данного этапа получается концептуальная модель, инвариантная к структуре базы данных. Часто она представляется в виде модели «сущность-связь».

Логическое проектирование. Преобразование требований к данным в структуры данных. На выходе получается СУБД-ориентированная структура базы данных и спецификации прикладных программ. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.

Физическое проектирование. Определение особенностей хранения данных, методов доступа и т. д. На этом этапе выбирают тип носителя, способ организации данных, методы доступа, определяют размер физического блока, управляют размещением данных на внешнем носителе, свободной памятью, определяют целесообразность сжатия данных и используют методы сжатия, оценивают физическую модель данных.

**5. Проектирование документальных баз данных.**

Проектирование документальных баз данных включает следующие этапы:

Постановка задачи. Анализ предметной области и совокупности характеризующих её документов. Выбор критериев эффективности функционирования базы данных и технико-экономическое обоснование целесообразности её создания.

Выбор информационно-поискового языка. Определение требований к техническим средствам и форме представления выходных документов. Утверждение технического задания по проектированию базы данных.

Техническое проектирование. Проектирование поисковых образов документов и поисковых образов запросов. Выбор СУБД для управления базой данных и базой данных документов, если документы представлены на машинном носителе. Разработка логико-семантического комплекса, то есть алгоритма, отражающего логику сопоставления содержания запросов и документов. Проектирование структуры базы данных. Разработка электронного каталога базы данных и формирование «словаря ключевых слов» для пользователей. Утверждение технического проекта.

Рабочее проектирование. Формирование базы данных документов на основе ретроспективного анализа, их индексирование и формирование на этой основе базы данных запросов.

Ввод в действие и сопровождение. Обучение пользователей, тестирование созданной базы данных, состоящей из базы данных документов и их поисковых образов. Утверждение акта приёмки-сдачи базы данных и стадия сопровождения, состоящая в её обновлении и эксплуатации в рабочем режиме.

**6. Концептуальное проектирование БД.**

Концептуальное проектирование базы данных — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Цель концептуального проектирования — создание концептуальной схемы данных на основе представлений о предметной области каждого отдельного типа пользователей.

Концептуальная модель базы данных включает в себя:

описание информационных объектов или понятий предметной области и связей между ними;

описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними.

Некоторые этапы концептуального проектирования:

Выделение локальных представлений, соответствующих обычно относительно независимым данным. Каждое такое представление проектируется как подзадача.

Формулирование сущностей, описывающих локальную предметную область проектируемой БД, и описание атрибутов, составляющих структуру каждой сущности.

Выделение ключевых атрибутов.

Спецификация связей между сущностями, удаление избыточных связей.

Анализ и добавление неключевых атрибутов.

Объединение локальных представлений.

Для проектирования БД активно используются ERD (Entity – Relationship Diagrams, диаграммы «сущность–связь»). С их помощью определяются важные для предметной области объекты (сущности), отношения друг с другом (связи) и их свойства (атрибуты).

**7. Логическое проектирование БД.**

Логическое проектирование базы данных (БД) — это процесс конструирования модели данных на основе конкретной модели данных, но независимо от конкретной системы управления базами данных (СУБД) и других деталей физической реализации.

Некоторые задачи логического проектирования:

определение всех информационных единиц и связей между ними, задание их имён;

определение типа информационных единиц и некоторых количественных характеристик (например, длины поля).

Результат стадии логического проектирования — логическая модель данных, включающая ER-диаграммы (диаграммы отношений), реляционную схему и документацию, в том числе словарь данных.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

**8. Физическое проектирование БД.**

Физическое проектирование базы данных (БД) — это создание схемы базы данных для конкретной системы управления базами данных (СУБД).

Цель физического проектирования — преобразование логической схемы с учётом синтаксиса, семантики и возможностей выбранной целевой СУБД.

Некоторые задачи физического проектирования:

на основе информации, предоставленной глобальной логической моделью данных, создаётся описание набора реляционных таблиц и ограничений для них;

определяются конкретные структуры хранения данных и методы доступа к ним, которые обеспечивают оптимальную производительность системы с БД;

разрабатываются средства защиты создаваемой системы.

Специфика конкретной СУБД при физическом проектировании включает ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п.. Также к этому этапу относится выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.

Результаты этого этапа документируются в форме схемы хранения на языке определения данных (DDL, Data Definition Language) выбранной СУБД.

**9. Базовые принципы и понятия технологии разработки объектно-ориентированных информационных систем на основе UML.**

Некоторые базовые принципы и понятия технологии разработки объектно-ориентированных информационных систем на основе UML:

Принцип онтологизации системы. Система представляется посредством классов, отражающих понятийную структуру предметной области. Классы определяются атрибутами (количественными свойствами) и операциями (методами, поведенческими свойствами).

Принцип декомпозиции. Система представляется совокупностью взаимодействующих между собой объектов. Объекты являются экземплярами классов и взаимодействуют посредством передачи сообщений.

Принцип инкапсуляции. Запрещается любой доступ 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.

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.

Принцип полиморфизма. Декларирует возможность работы с объектом без информации о конкретном классе, экземпляром которого он является.

**10. Возможности и достоинства UML. Инструментальные средства визуального моделирования.**

Возможности и достоинства UML:

Улучшенная коммуникация. Единый язык для разработчиков, аналитиков, тестировщиков, заказчиков, позволяющий им эффективно обмениваться информацией о проекте.

Упрощение проектирования. UML помогает визуализировать, структурировать систему, отслеживая проблемы уже на ранних этапах проектирования.

Повышение качества кода. Визуализация системы выявляет несоответствия, непоследовательности в проектировании.

Создание чёткой, лаконичной документации для проекта. В результате команда и заказчики могут быстро ознакомиться с функциональностью всего проекта.

Повышение повторного использования кода. UML помогает определить модульные компоненты системы, которые могут применяться повторно в других проектах.

Некоторые инструментальные средства визуального моделирования на языке UML:

Lucidchart. Предлагает широкий выбор готовых шаблонов и символов, а также возможность совместной работы в реальном времени.

Visual Paradigm. Полнофункциональный инструмент для моделирования и создания UML-диаграмм.

Creately. Предлагает простой в использовании веб-интерфейс для создания UML-диаграмм.

PlantUML. Инструмент для создания диаграмм, основанный на текстовом описании с использованием языка UML.

*Приложение 2*
*к рабочей программе дисциплины (модуля)*
*Б1.О.05.05 Проектирование информационных систем на транспорте*

**Перечень литературы, необходимой для освоения дисциплины (модуля)**

Направление подготовки: 15.03.06 Мехатроника и робототехника
Профиль подготовки: Физические основы мехатроники и робототехники
Квалификация выпускника: бакалавр
Форма обучения: очная
Язык обучения: русский
Год начала обучения по образовательной программе: 2025

**Основная литература:**

1. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. — Москва : ФОРУМ : ИНФРА-М, 2020. — 368 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-8199-0718-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1066784 – Режим доступа: по подписке.
2. Заботина, Н. Н. Проектирование информационных систем: Учебное пособие / Заботина Н.Н. - Москва : НИЦ ИНФРА-М, 2016. - 331 с. (Высшее образование: Бакалавриат) ISBN 978-5-16-004509-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/542810 – Режим доступа: по подписке.
3. Коваленко, В. В. Проектирование информационных систем : учеб. пособие / В.В. Коваленко. — Москва : ФОРУМ : ИНФРА-М, 2018. — 320 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-00091-628-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/980117 – Режим доступа: по подписке.
4. Гвоздева, В. А. Информатика, автоматизированные информационные технологии и системы: учебник / В. А. Гвоздева. - Москва: ФОРУМ: ИНФРА-М, 2020. - 542 с. - ISBN 978-5-8199-0877-8. - Текст: электронный. - URL: https://znanium.com/catalog/product/1066785 - Режим доступа: по подписке.
5. Балдин, К. В. Информационные системы в экономике: Учебное пособие / К.В. Балдин. - Москва : НИЦ Инфра-М, 2013. - 218 с. (Высшее образование; Бакалавриат). ISBN 978-5-16-005009-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/397677 – Режим доступа: по подписке.

*Приложение 3*
*к рабочей программе дисциплины (модуля)*
*Б1.О.05.05 Проектирование информационных систем на транспорте*

**Перечень информационных технологий, используемых для освоения дисциплины (модуля), включая перечень программного обеспечения и информационных справочных систем**

Направление подготовки: 15.03.06 Мехатроника и робототехника
Профиль подготовки: Физические основы мехатроники и робототехники
Квалификация выпускника: бакалавр
Форма обучения: очная
Язык обучения: русский
Год начала обучения по образовательной программе: 2025

Освоение дисциплины (модуля) предполагает использование следующего программного обеспечения и информационно-справочных систем:
Программное обеспечение: операционная система Windows, Microsoft office, PyCharm, Kaspersky Free для Windows
Электронная библиотечная система «ZNANIUM.COM»
Электронная библиотечная система Издательства «Лань»
Электронная библиотечная система «Консультант студента»