

**Лабораторные работы**  
**по дисциплине «Технология программирования»**  
**ЗНАКОМСТВО С CASE-СИСТЕМОЙ RATIONAL ROSE**

**1. Теоретическая часть**

**1.1. Введение в Rational Rose**

Rational Rose - семейство объектно-ориентированных CASE-средств фирмы Rational Software Corporation, предназначенное для автоматизации процессов анализа и проектирования программного обеспечения (ПО), а также для генерации кодов на различных языках программирования и выпуска проектной документации. Rational Rose использует метод объектно-ориентированного анализа и проектирования, основанный на языке UML. Текущая версия Rational Rose реализует генерацию кодов программ для C++, Visual C++, Visual Basic, Java, PowerBuilder, CORBA Interface Definition Language (IDL), генерацию описаний баз данных для ANSI SQL, Oracle, MS SQL Server, IBM DB2, Sybase, а также позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций. Кроме того, Rational Rose содержит средства реверсного инжиниринга программ и баз данных, обеспечивающие повторное использование программных компонентов в новых проектах.

В основе работы Rational Rose лежит построение диаграмм и спецификаций UML, определяющих архитектуру системы, ее статические и динамические аспекты. В составе Rational Rose можно выделить шесть структурных основных компонентов: репозиторий, графический интерфейс пользователя, средства просмотра проекта (браузер), средства контроля проекта, средства сбора статистики и генератор документов. К ним добавляются генератор кодов (индивидуальный для каждого языка) и анализатор для C++, обеспечивающий реверсный инжиниринг.

*Репозиторий* представляет собой базу данных проекта. *Обзор* обеспечивает «навигацию» по проекту, в том числе перемещение по иерархиям классов и подсистем, переключение от одного вида диаграмм к другому и т.д. *Средства контроля и сбора статистики* дают возможность находить и устранять ошибки по мере развития проекта, а не после завершения его описания. *Генератор отчетов* формирует тексты выходных документов на основе содержащейся в репозитории информации.

*Средства автоматической генерации кодов программ* на языке C++, используя информацию, содержащуюся в диаграммах классов и компонентов, формируют файлы заголовков и файлы описаний классов и объектов. Создаваемая таким образом заготовка (шаблон) программы может быть уточнен в дальнейшем путем прямого программирования на языке C++.

*Анализатор кодов C++* реализован в виде отдельного программного модуля. Его назначение - создавать модули проектов Rational Rose на основе информации, содержащейся в определяемых пользователем исходных текстах на C++. В процессе работы анализатор осуществляет контроль правильности исходных текстов и диагностику ошибок. Модель, полученная в результате его

работы, может целиком или фрагментарно использоваться в различных проектах. Анализатор обладает широкими возможностями настройки по входу и выходу. Например, можно определить типы исходных файлов, базовый компилятор, задать, какая информация должна быть включена в формируемую модель, и какие элементы выходной модели следует выводить на экран. Таким образом, Rational Rose/C++ обеспечивает возможность повторного использования программных компонентов.

В результате разработки проекта с помощью CASE-средства Rational Rose формируются следующие документы:

- диаграммы UML, в совокупности представляющие собой модель разрабатываемой программной системы;
- спецификации классов, объектов, атрибутов и операций;
- заготовки текстов программ.

Тексты программ являются заготовками для последующей работы программистов. Состав информации, включаемой в программные файлы, определяется либо по умолчанию, либо по усмотрению пользователя. В дальнейшем эти исходные тексты развиваются программистами в полноценные программы.

В рамках Rational Rose существуют различные программные инструментари, отличающиеся между собой диапазоном реализованных возможностей. В лабораторной работе используется система Rational Rose 2001, которая существует в четырех основных модификациях:

- Rational Rose Enterprise Edition
- Rational Rose Professional Edition
- Rational Rose Modeler Edition
- Rational Rose для UNIX

Наиболее полными возможностями обладает первая из указанных модификаций этого средства. Из этих возможностей можно отметить: генерацию кодов на различных языках программирования (Java, C++, VisualBasic, PowerBuilder), обратную генерацию диаграмм (реинжиниринг) на основе программного кода и выпуск проектной документации.

## **1.2. Особенности рабочего интерфейса Rational Rose**

В CASE-средстве Rational Rose реализованы общепринятые стандарты на рабочий интерфейс программы, подобно известным средам визуального программирования. После установки Rational Rose на компьютер запуск этой программы в среде MS Windows приводит к появлению на экране главного окна программы (рис. 1.1.).

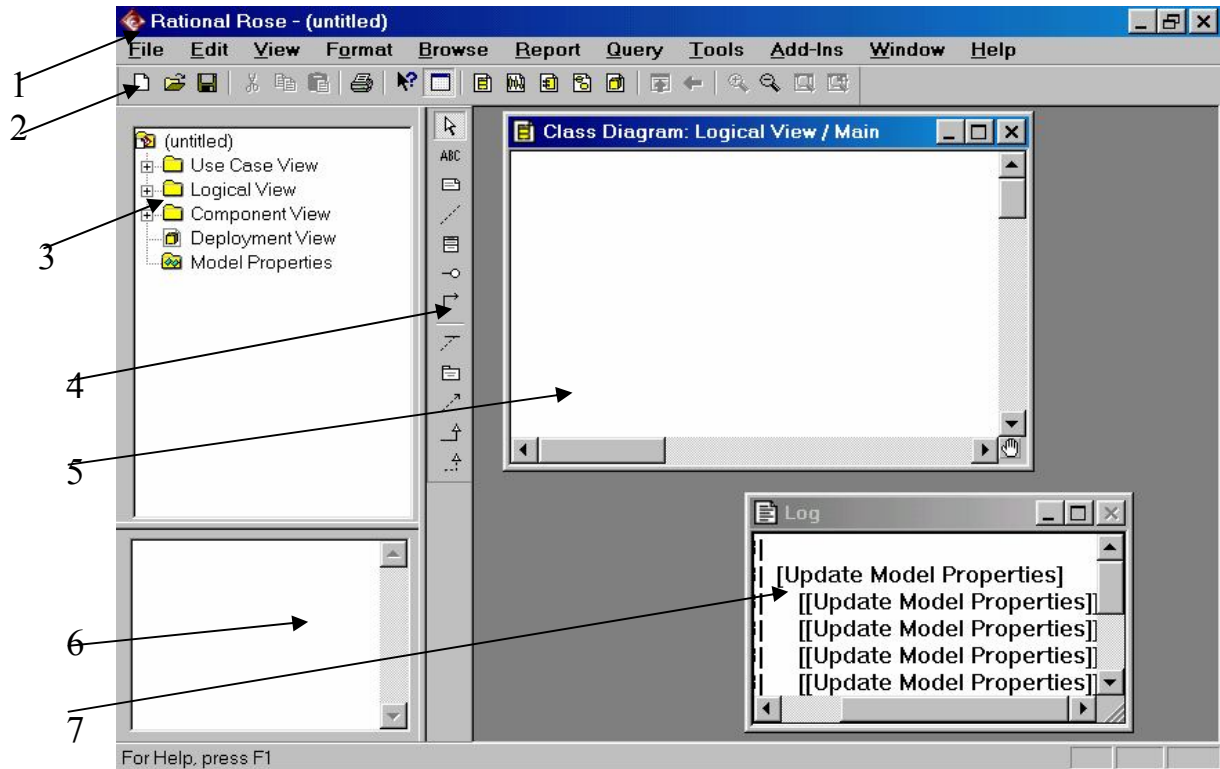


Рис. 1.1. Главное окно Rational Rose

Рабочий интерфейс Rational Rose состоит из следующих основных элементов:

1. Главное меню программы.
2. Стандартная панель инструментов.
3. Окно браузера.
4. Специальная панель инструментов.
5. Окно диаграммы.
6. Окно документации.
7. Окно журнала.

Рассмотрим кратко назначение и основные функции каждого из этих элементов.

Главное меню программы выполнено в общепринятом стандарте. Меню состоит из одиннадцати пунктов:

- **File** (файл) предназначен для сохранения, загрузки, обновления проекта, печати диаграмм;
- **Edit** (редактирование) предназначен для копирования и восстановления данных в буфер обмена Windows, а также для редактирования свойств и стилей объектов;
- **View** (вид) предназначен для настройки представления окон меню и строк инструментов;
- **Format** (форматирование) предназначен для настройки формата текущего значка, цветовой гаммы, линий и т.д.;

- **Browse** (просмотр) предназначен для навигации между диаграммами и спецификациями диаграмм, представленных в модели;
- **Report** (отчет) предназначен для получения различного вида справок и отчетов;
- **Query** (запрос) предоставляет возможности контролировать, какие элементы модели будут показаны на текущей диаграмме;
- **Tools** (инструменты) предоставляет доступ к различным дополнительным инструментам и подключаемым модулям;
- **Add-Ins** (добавить) предоставляет доступ к менеджеру подключаемых модулей;
- **Window** (окно) позволяет управлять окнами на рабочем столе;
- **Help** (помощь) позволяет получать справочную информацию.

*Стандартная панель инструментов* располагается ниже главного меню программы. Некоторые из инструментов недоступны (новый проект не имеет никаких элементов). Стандартная панель инструментов обеспечивает быстрый доступ к тем командам меню, которые выполняются разработчиками наиболее часто.

Пользователь может настроить внешний вид этой панели по своему усмотрению. Для этого необходимо выбрать пункт меню **Tools-»Options** (Инструменты-»Параметры) и открыть вкладку **Toolbars** (Панели инструментов). Этим способом можно показать или скрыть различные кнопки инструментов, а также изменить их размер.

Следует заметить, что внешний вид панели инструментов определяется не только выбором и не только видом разрабатываемой диаграммы, но и выбором графической нотации для изображения самих элементов этих диаграмм. В Rational Rose реализованы три таких нотации: UML, OMT и Booch. Одна и та же диаграмма может быть представлена различным образом, для этого достаточно выбрать желаемое представление через пункт меню **View** (Вид). При этом никаких дополнительных действий выполнять не требуется — диаграмма преобразуется в выбранную нотацию автоматически. В дальнейшем система Rational Rose рассматривается в контексте только языка UML. В связи с этим особенности двух других нотаций, которые отражают эволюционный аспект этой системы, рассматриваться не будут.

*Окно браузера* по умолчанию располагается в левой части основного окна под стандартной панелью инструментов. Обзор организует представления модели в виде иерархической структуры, которая упрощает навигацию и позволяет отыскать любой элемент модели в проекте. При этом любой элемент, который разработчик добавляет в модель, сразу отображается в окне браузера. Соответственно, выбрав элемент в окне браузера, можно его визуализировать в окне диаграммы или изменить его спецификацию. Обзор позволяет также организовывать элементы модели в пакеты и перемешать элементы между различными представлениями модели. При желании окно браузера можно расположить в другом месте рабочего интерфейса либо скрыть вовсе, используя для этого пункт меню **View-»Browser**. Можно также изменить размеры браузера, переместив мышью границу его внешней рамки.

*Специальная панель инструментов* располагается между окном браузера и окном диаграммы в средней части основного окна. По умолчанию показывается панель инструментов для построения диаграммы классов модели (при этом окно диаграммы классов должно быть активным).

Расположение специальной панели инструментов можно изменять, переместив рамку панели в нужное место. Можно настраивать и состав панели, добавляя или удаляя отдельные кнопки, соответствующие тем или иным инструментам. Назначения кнопок можно узнать из всплывающих подсказок.

*Окно диаграммы* является основной рабочей областью ее интерфейса, в которой визуализируются различные представления модели проекта. По умолчанию окно диаграммы располагается в правой части основного окна, однако его расположение и размеры также можно изменить. При разработке нового проекта, если не был использован мастер проектов, окно диаграммы представляет собой чистую область, не содержащую никаких элементов модели.

Название диаграммы, которая располагается в данном окне, указывается в строке заголовка программы или, если окно не развернуто во весь экран, в строке заголовка окна диаграммы. Одновременно в окне диаграммы могут присутствовать несколько диаграмм, однако активной может быть только одна из них. Переключение между диаграммами можно осуществить выбором нужного представления на стандартной панели инструментов либо через пункт меню **Window** (Окно). При активизации отдельного вида диаграммы изменяется внешний вид специальной панели инструментов, которая настраивается под конкретный вид диаграммы.

*Окно документации* по умолчанию может не присутствовать на экране. В этом случае оно может быть активизировано через пункт меню **View->Documentation** (Вид->Документация), после чего появится ниже браузера.

Окно документации, как следует из его названия, предназначено для документирования элементов представления модели. В него можно записывать самую различную информацию, и что важно – на русском языке. Эта информация в последующем преобразуется в комментарии и никак не влияет на логику выполнения программного кода.

В окне документации активизируется та информация, которая относится к отдельному выделенному элементу диаграммы. При этом выделить элемент можно либо в окне браузера, либо в окне диаграммы. При добавлении нового элемента на диаграмму (например, класса) автоматически генерируется документация к нему, которая является пустой (No documentation). В последующем разработчик самостоятельно вносит необходимую пояснительную информацию, которая запоминается и может быть изменена в ходе работы над проектом.

Так же, как и для других окон, можно изменять размеры и положение окна документации.

*Окно журнала* (Log) предназначено для автоматической записи различной служебной информации, образующейся в ходе работы с программой.

В журнале фиксируется время и характер выполняемых разработчиком действий, таких как обновление модели, настройка меню и панелей инструментов, а также сообщений об ошибках, возникающих при генерации программного кода.

Окно журнала всегда присутствует на экране в области окна диаграммы. Однако оно может быть закрыто другими окнами с диаграммами или быть свернутым. Активизировать окно журнала можно через меню **Window-»Log** (Окно-»Журнал). В этом случае оно изображается поверх других окон в правой области основного окна. Полностью удалить это окно нельзя, его можно только минимизировать.

В модели, создаваемой с помощью системы Rational Rose, поддерживаются четыре вида представления информационных объектов модели:

- представление вариантов использования (Use Case View);
- логическое представление (Logical View);
- представление компонентов (Component View);
- представление размещения (Deployment View).

*Представление вариантов использования* содержит всех действующих лиц, все варианты использования и их диаграммы для конкретной системы. Оно может также содержать диаграммы взаимодействия (последовательности и кооперации), диаграммы состояний и диаграммы деятельности.

*Логическое представление* служит для отображения информации о том, как система будет реализовывать поведение, описанное в вариантах использования. Оно дает подробную картину составных частей системы и описывает взаимодействие этих частей. Логическое представление включает конкретные требуемые классы, диаграммы классов, а так же может включать диаграммы состояний и диаграммы деятельности для отдельных классов. С их помощью конструируется детальный проект создаваемой системы.

*Представление компонентов* содержит диаграммы компонентов, которые включают компоненты, являющиеся физическими модулями кода, и пакеты, являющиеся группами связанных компонентов.

*Представление размещения* - это последнее представление в системе Rational Rose. Оно соответствует физическому размещению системы, которое может отличаться от ее логической архитектуры. В представление размещения входят диаграммы размещения, которые включают процессы и потоки, исполняемые в отведенной для них области памяти, а так же процессоры, способные обрабатывать данные, и устройства, не способные обрабатывать данные (например, терминалы ввода-вывода и принтеры).

### 1.3. Начало работы над проектом в среде Rational Rose

Общая последовательность работы над проектом состоит в следующем.

В первую очередь производится анализ списка операций, которые будет выполнять система, и определяется множество объектов системы, которые должны выполнять данные функции. Таким образом, определяются требования к системе и границы предметной области. С этой целью создается диаграмма вариантов использования (Use case).

Затем определяется список классов, которые должны присутствовать в системе, пока без конкретной детализации и подробного описания действий. Для этого используется диаграмма классов (Class diagram).

После определения в системе необходимых классов описывается взаимодействие классов при помощи диаграммы последовательности (Sequence diagram) и диаграммы кооперации (Collaboration diagram).

Далее определяется поведение конкретных классов при помощи диаграммы состояний (State diagram) и диаграммы деятельности (Activity diagram).

На основании производимых классами действий создается окончательная иерархия классов системы при помощи диаграммы классов, и определяются компоненты, в которые эти классы необходимо включить при помощи диаграммы компонентов (Component diagram). Размещение компонентов определяется с помощью диаграммы развертывания (Deployment diagram).

После проверки правильности модели и согласованности спецификаций ее элементов на основе полученной диаграммы классов система Rational Rose позволяет сгенерировать текст программного кода на одном из выбранных языков программирования. Как правило, этот текст дорабатывается в соответствующей среде программирования с целью получения исполняемых модулей программ, ориентированных на работу в определенной операционной среде и вычислительной платформе.

При создании моделей систем для различных предметных областей порядок работы может несколько отличаться от приведенного, поэтому при их разработке необходимо внести в него соответствующие изменения.

Разработчику необходимо учитывать, что проектирование системы — это итерационный процесс. На практике трудно за один шаг создать полный проект системы. Приходится многократно возвращаться к уже созданным диаграммам и вносить в них изменения.

Для нового проекта можно воспользоваться мастером типовых проектов (если он установлен в данной конфигурации). Мастер типовых проектов доступен из меню **File-»New** (Файл-»Создать). Если мастер недоступен, то на экране появляется рабочий интерфейс программы с чистым окном диаграммы.

**ПРИМЕЧАНИЕ.** При выполнении лабораторных работ необходимо создавать новый пустой проект, для чего в мастере выбора типового проекта нужно нажать кнопку Отмена (Cancel).

Если имеется готовый проект (файл с расширением .mdl), то его можно открыть для последующей модификации через меню **File-»Open**

(Файл-»Открыть). В этом случае программа загрузит существующий проект со всеми имеющимися в нем диаграммами, спецификациями и документацией.

По окончании 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.

Как и другие программы, Rational Rose позволяет настраивать глобальные параметры среды, такие как выбор шрифтов и цвета для представления различных элементов модели. Настройка шрифтов производится через меню **Tools-»Options** (Инструменты-»Параметры). Характерной особенностью среды является возможность работы с символами кириллицы. Однако следует учесть, что при спецификации элементов модели с последующей генерацией текста программного кода нужно сразу записывать имена и свойства элементов символами того языка, который поддерживается соответствующей системой программирования.

Общий процесс работы над проектом заключается в добавлении на диаграммы соответствующих графических элементов, установлении отношений между этими элементами, их спецификации и документировании. Процесс добавления графических элементов на диаграммы аналогичен реализованному в популярных средах визуального программирования. При этом следует предостеречь от неосторожного добавления элементов на диаграммы, поскольку каждый добавляемый элемент заносится в браузер. Последующее удаление элемента с диаграммы автоматически не удаляет его из браузера, и необходимо предпринять дополнительные меры для удаления ненужного элемента из модели проекта.

## 1.4. Разработка основных видов диаграмм в среде Rational Rose

### 1.4.1. Разработка диаграммы вариантов использования

Работа над проектом в среде Rational Rose начинается с общего анализа проблемы и построения диаграммы вариантов использования, которая отражает функциональное назначение проектируемой программной системы.

Для разработки диаграммы вариантов использования в среде Rational Rose необходимо активизировать соответствующую диаграмму в окне диаграммы. Это можно сделать различными способами:

- раскрыть представление вариантов использования в браузере (Use Case View) и дважды щелкнуть на пиктограмме **Main** (Главная);
- через пункт меню **Browse-»Use Case Diagram** (Обзор-»Диаграмма вариантов использования).

При этом появляется специальная панель инструментов, содержащая графические элементы, характерные для разработки диаграммы вариантов использования.

На этой панели инструментов присутствуют все необходимые для построения диаграммы варианты использования элементы. Назначение отдельных кнопок панели можно узнать из всплывающих подсказок. Для



добавления элемента нужно нажать кнопку с изображением соответствующего примитива, после чего щелкнуть мышью на свободном месте диаграммы. На диаграмме появится изображение выбранного элемента с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию. Имя элемента может быть изменено разработчиком либо сразу после размещения элемента на диаграмме, либо в ходе последующей работы над проектом. По щелчку правой кнопкой мыши на выбранном элементе вызывается контекстное меню элемента, среди команд которого имеется команда **Open Specification** (Открыть спецификацию). При выполнении этой команды открывается диалоговое окно со специальными вкладками, в поля которых заносится вся необходимая информация по данному элементу.

Добавление на диаграмму связей между элементами выполняется следующим образом. На специальной панели инструментов выбирается требуемый тип связи щелчком по кнопке с соответствующим изображением. Затем выделяется первый элемент связи (источник, от которого исходит связь) и, не отпуская нажатую левую кнопку мыши, перемещается ее указатель ко второму элементу связи (приемник, к которому направлена связь). После перемещения ко второму элементу кнопку мыши следует отпустить, а на диаграмму вариантов использования будет добавлена новая связь.

При необходимости можно указать характеристики связи между элементами. Для этого используется команда контекстного меню связи **Open Specification** (Открыть спецификацию). На вкладке **General** можно указать имя связи, ее стереотип и другие характеристики. Для определения степени связи следует выбрать требуемое значение в списке **Multiplicity** (Кратность) на вкладках **Role A Detail** или **Role B Detail**.

Диаграмма вариантов использования является высокоуровневым представлением модели, поэтому она не должна содержать слишком много вариантов использования и актеров. В последующем построенная диаграмма может быть изменена добавлением новых элементов или их удалением. Для удаления элемента из диаграммы необходимо выделить удаляемый элемент и нажать клавишу **Delete** или выполнить команду **Edit»Delete**. В этом случае элемент удаляется только из данной диаграммы, но остается в модели. Удалить элемент из модели можно с помощью команды **Edit»Delete from Model**.

При работе со связями на диаграмме вариантов использования следует помнить о назначении соответствующих связей. Если для двух элементов выбранный вид связи не является допустимым, то система сообщит об этом разработчику (см. рис. 1.2), и такая связь не будет добавлена на диаграмму.

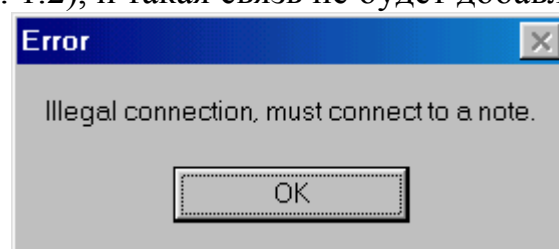


Рис. 1.2

Более подробная информация по построению диаграмм вариантов использования в среде Rational Rose приведена в [2].

#### 1.4.2. Документирование вариантов использования.

Описательную спецификацию варианта использования (поток событий) обычно хранят в некотором файле в качестве документации. Чтобы связать файл с вариантом использования необходимо выполнить следующие действия:

- щелкнуть правой кнопкой мыши на соответствующем варианте использования;
- в открывшемся меню выбрать пункт **Open Specification** (Открыть спецификацию);
- перейти на вкладку **Files** (Файлы);
- щелкнуть правой кнопкой мыши в белой области и в открывшемся меню выбрать пункт **Insert File** (Вставить файл);
- указать имя файла с расширением .doc и нажать на кнопку **Open** (Открыть), чтобы прикрепить файл к варианту использования.

#### 1.4.3. Разработка диаграммы классов

Диаграмма классов является основным логическим представлением модели и содержит самую подробную информацию о внутреннем устройстве объектно-ориентированной программной системы. Активизировать диаграмму классов в окне диаграммы можно несколькими способами:

- эта диаграмма появляется по умолчанию в окне диаграммы после создания нового проекта;
- раскрыть логическое представление в браузере (Logical View) и дважды щелкнуть на пиктограмме **Main** (Главная);
- щелкнуть на кнопке с изображением диаграммы классов на стандартной панели инструментов;
- через пункт меню **Browse-»Class Diagram** (Обзор-»Диаграмма классов).


Если требуется построить диаграмму классов для отдельного варианта использования, то это можно выполнить одним из следующих способов:


- раскрыть представление вариантов использования в браузере (Use Case View), выделить рассматриваемый вариант использования и выбрать пункт контекстного меню **New-»Class Diagram** (Новая-»Диаграмма классов). Ввести название диаграммы в браузере.


- выбрать пункт меню **Browse-»Class Diagram** (Обзор-»Диаграмма классов) и в появившемся окне **Select Class Diagram** (Выбор диаграммы классов) указать название варианта использования.


После открытия окна диаграммы классов на экране появляется специальная панель инструментов. Добавление и удаление элементов происходит аналогично выполнению тех же действий на диаграмме вариантов использования, однако у каждого класса имеется обширная спецификация, содержащая информацию об его атрибутах и операциях. При этом видимость атрибутов и операций изображается в форме специальных пиктограмм.

Используемые пиктограммы для видимости изображаются перед именем соответствующего атрибута или операции и имеют следующий смысл:

■ *общий, открытый* (Public) — устанавливается по умолчанию. В этом случае элемент доступен всем остальным классам модели. В нотации языка UML такому элементу соответствует знак .

■ *защищенный* (Protected). В этом случае элемент доступен из самого класса или из его потомков. В нотации языка UML такому элементу соответствует знак .

■ *закрытый* (Private). В этом случае элемент не доступен никакому другому классу, кроме того, в котором он определен. В нотации языка UML такому элементу соответствует знак .

■ *пакетный* (Implementation). Такой элемент является общим только в пределах своего пакета. В нотации языка UML такому элементу соответствует знак .

Для отдельных классов можно задать стереотипы через пункт контекстного меню **Open Specification** (Открыть спецификацию). На вкладке **General** предлагается выбор соответствующих значений из раскрывающегося списка.

Для более удобного просмотра операций и атрибутов классов можно изменить способ отображения классов на диаграмме. Для этого в контекстном меню класса выбрать **Options»Stereotype Display»Decoration**.

На вкладке **Attributes** (Атрибуты) для атрибутов выделенного класса можно задать тип данных и начальные значения атрибута, а также другие характеристики, через пункт контекстного меню **Specification** (Спецификация).

На вкладке **Operations** (Операции) для отдельных операций выбранного класса можно задать тип возвращаемого результата, добавить аргументы к операции, задать исключительные ситуации и целый ряд дополнительных свойств. Для этого следует выбрать пункт контекстного меню **Specification** (Спецификация).

Добавление к классу очередной операции или атрибута выполняется следующим образом:

- выбрать соответствующую вкладку диалогового окна **Class Specification** (Спецификация класса) и нажать клавишу **Insert**;
- выбрать соответствующую вкладку диалогового окна **Class Specification** (Спецификация класса) и с помощью контекстного меню выполнить команду **Insert**.

Добавление на диаграмму классов отношений (связей) между классами типа ассоциаций, зависимостей, агрегаций и обобщений выполняется следующим образом. На специальной панели инструментов выбирается требуемый тип связи щелчком по кнопке с соответствующим изображением. Если связь направленная, то на диаграмме классов надо выделить первый элемент связи (источник, от которого исходит связь) и, не отпуская нажатую левую кнопку мыши, переместить ее указатель ко второму элементу связи

(приемник, к которому направлена связь). После перемещения ко второму элементу кнопку мыши следует отпустить, а на диаграмму классов будет добавлена новая связь.

Если же связь ненаправленная (двунаправленная), то порядок выбора классов для этой связи произвольный. Для связей можно определить кратность каждого из концов связи, задать имя и стереотип, использовать ограничения и роли, а также некоторые другие свойства. Доступ к спецификации связи можно получить после выделения связи на диаграмме и выбора пункта контекстного меню **Open Specification** (Открыть спецификацию). Степень связи устанавливается как на диаграмме вариантов использования.

#### 1.4.4. Разработка диаграмм взаимодействия

Диаграммы взаимодействия могут быть разработаны как для проектируемой системы в целом, так и для конкретного варианта использования (основного или дополнительного).

Диаграмма взаимодействия может быть активизирована одним из следующих способов:

- щелкнуть на кнопке с изображением диаграммы взаимодействия на стандартной панели инструментов;
- через пункт меню **Browse-»Interaction Diagram** (Обзор-»Диаграмма взаимодействия).

В открывшемся окне **Select Interaction Diagram** (Выбрать диаграмму взаимодействия) выделить элемент, для которого необходимо создать диаграмму взаимодействия, например, некоторый вариант использования (см. рис. 1.3).

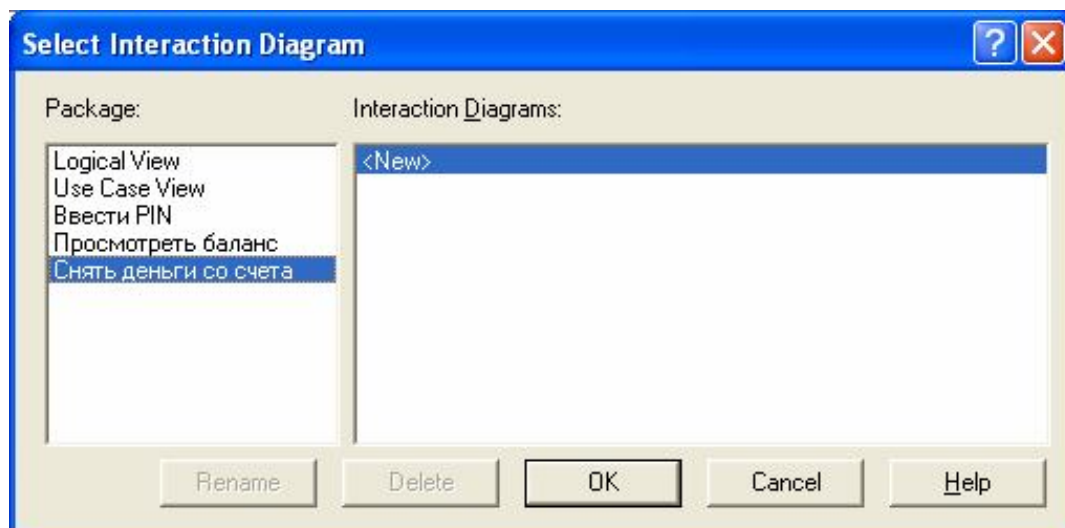


Рис. 1.3

Далее в окне **New Interaction Diagram** (см. рис. 1.4) предлагается выбрать один из двух типов диаграмм взаимодействия (**Diagram Types**): диаграмму последовательности (**Sequence**) или диаграмму кооперации (**Collaboration**).

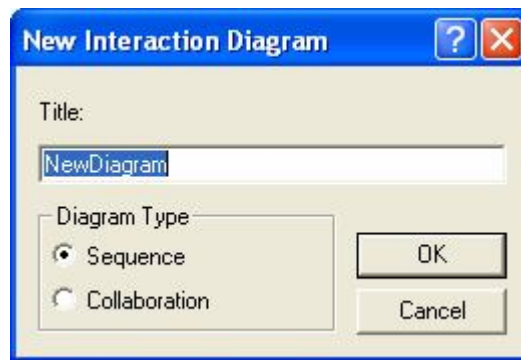


Рис. 1.4

После выполнения указанных действий в окне диаграммы появится чистое изображение для размещения элементов диаграммы, выбираемых с помощью специальной панели инструментов.

#### 1.4.4.1. Разработка диаграммы последовательности

Построение диаграммы последовательности сводится к добавлению или удалению отдельных объектов и сообщений, а также к определению их спецификации. Доступ к спецификации этих элементов возможен либо через контекстное меню, либо через пункт меню **Browse-»Specification** (Обзор-»Спецификация).

Диаграмма последовательности строится на основе разработанной ранее описательной спецификации, по которой можно определить взаимодействие классов, выделив последовательность обмена сообщениями между ними. Взаимодействие между классами в дальнейшем может быть реализовано, например, с помощью обмена сообщениями между классами или последовательного вызова методов классов. Во втором случае при генерации программного кода система Rational Rose может дополнить код программы вызовами методов классов, соответствующих ранее определенным сообщениям. Более сложные механизмы обмена сообщениями придется реализовывать самостоятельно.

Для того чтобы система преобразовывала сообщения в вызовы методов, необходимо определять такие сообщения как новые операции классов. В этом случае при добавлении нового сообщения на диаграмму последовательности (за исключением возврата результатов из методов) необходимо в его контекстном меню выбрать команду **new operation**, после чего в появившемся окне задать необходимые параметры сообщения (например, список формальных параметров метода, тип возвращаемого результата). При этом данный метод автоматически добавится к методам класса – приемника сообщения.

#### 1.4.5. Разработка диаграммы кооперации

Диаграмма кооперации является другим способом визуализации взаимодействия в модели и, как и диаграмма последовательности, оперирует объектами и сообщениями. Особенность работы в среде Rational Rose заключается в том, что этот тип диаграммы создается автоматически после построения диаграммы последовательности и нажатия клавиши <F5>. С

помощью этой же клавиши осуществляется переключение между диаграммами последовательности и кооперации.

После того как диаграмма кооперации активизирована, специальная панель инструментов приобретает соответствующий вид. На панели имеются кнопки с пиктограммами объектов и различных типов сообщений.

Работа с диаграммой кооперации состоит в добавлении или удалении объектов и сообщений, а также их специфицировании. При этом изменения, вносимые в диаграмму кооперации, автоматически вносятся и в диаграмму последовательности, что можно увидеть, активизировав последнюю нажатием клавиши <F5>.

#### 1.4.6. Разработка диаграммы состояний

Данный тип диаграмм в среде Rational Rose можно разрабатывать как для системы в целом, так и для конкретного варианта использования или отдельного класса.

Диаграмма состояний для системы в целом может быть активизирована одним из следующих способов:

- щелкнуть на кнопке с изображением диаграммы состояний на стандартной панели инструментов;

- через пункт меню **Browse-»State Machine Diagram** (Обзор-»Диаграмма состояний).

В открывшемся окне в поле **State Diagrams** выбрать **New**, а в следующем окне выбрать тип диаграммы **Statechart**.

Для того чтобы построить диаграмму состояний для отдельного элемента (варианта использования или класса) необходимо выделить элемент на соответствующей диаграмме или в браузере и выполнить пункт контекстного меню **New-»Statechart Diagram** (Новая-»Диаграмма состояний).

После выполнения указанных действий в окне диаграммы появится чистое изображение для размещения элементов этой диаграммы, выбираемых с помощью специальной панели инструментов. Процесс добавления и удаления состояний и переходов на диаграмму состояний аналогичен этим же действиям с элементами других диаграмм.

После добавления состояния или перехода на диаграмму состояний можно открыть спецификацию выбранных элементов и определить их специальные свойства, доступные на соответствующих вкладках. При необходимости можно визуализировать вложенность состояний и подключить историю отдельных состояний.

#### 1.4.7. Разработка диаграммы деятельности

Диаграмма деятельности, так же как и диаграмма состояний, может относиться ко всей системе, конкретному варианту использования или отдельному классу.

Диаграмма деятельности для системы в целом может быть активизирована одним из следующих способов:



■ щелкнуть на кнопке с изображением диаграммы состояний на стандартной панели инструментов;

■ через пункт меню **Browse»State Machine Diagram** (Обзор»Диаграмма состояний).

В открывшемся окне в поле **State Diagrams** выбрать **New**, а в следующем окне выбрать тип диаграммы **Activity**.

Для того чтобы построить диаграмму деятельности для отдельного элемента (варианта использования или класса) необходимо выделить элемент на соответствующей диаграмме или в браузере и выполнить пункт контекстного меню **New»Activity Diagram** (Новая»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.

Процесс редактирования диаграммы деятельности выполняется аналогично редактированию других диаграмм в системе Rational Rose.

Свойства элементов диаграммы деятельности можно изменить, открыв спецификацию соответствующих элементов.

Диаграммы деятельности позволяют определять ответственность отдельных классов за выполняемую деятельность. Для этого на диаграмме выделяют специальные зоны, называемые дорожками ответственности. Каждая из дорожек соответствует определенному классу, поэтому на ней располагается только та деятельность, которая выполняется данным классом.

Для того чтобы создать дорожки ответственности, необходимо выбрать на панели инструментов диаграммы деятельности элемент **Swimlane** (рис. 1.5), после чего на диаграмме появится дорожка ответственности.

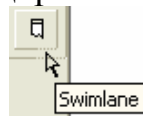


Рис. 1.5

Чтобы разместить элемент диаграммы на дорожке ответственности, необходимо перетащить его с помощью мыши в соответствующую область диаграммы.

Для удаления дорожки ответственности необходимо вызвать контекстное меню на заголовке дорожки и выполнить команду **Delete** (рис. 1.6).

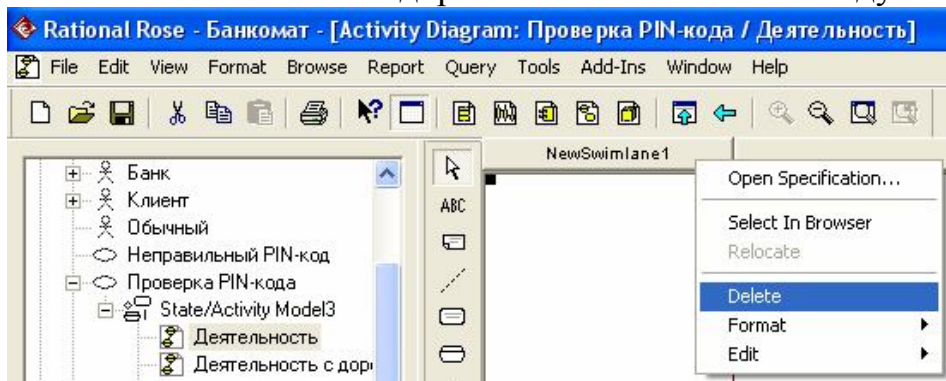


Рис. 1.6

#### 1.4.8. Разработка диаграммы компонентов

Диаграмма компонентов является частью физического представления модели проектируемой системы. На ней отображаются компоненты программного обеспечения и связи между ними. При этом на такой диаграмме могут быть выделены два типа компонентов: исполняемые компоненты и библиотеки кода.

Активизация диаграммы компонентов может быть выполнена одним из следующих способов:

- щелкнуть на кнопке с изображением диаграммы компонентов на стандартной панели инструментов;
- раскрыть компонентное представление в браузере (Component View) и дважды щелкнуть на пиктограмме **Main** (Главная);
- через пункт меню **Browse-»Component Diagram** (Обзор-»Диаграмма компонентов).

Добавление и удаление элементов происходит аналогично элементам других диаграмм. Для каждого компонента можно определить различные детали, такие как стереотип, язык программирования, декларации, классы. Работа с этими деталями компонентов осуществляется через спецификацию компонента, доступную после вызова контекстного меню.

Каждый класс модели (или ее подсистема) преобразуется в компонент исходного кода. Между отдельными компонентами устанавливаются зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы.

#### 1.4.9. Разработка диаграммы развертывания

Диаграмма развертывания является второй составной частью физического представления модели. 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 в браузере (Deployment View);
- через пункт меню **Browse-»Deployment Diagram** (Обзор-»Диаграмма развертывания).

Работа с диаграммой развертывания состоит в создании процессоров и устройств, их спецификации, установлении связей между ними, а также добавлении и спецификации процессов. Применительно к отдельным процессорам можно использовать стереотипы.

#### 1.4.10. Генерация программного кода

Одним из наиболее важных свойств среды Rational Rose является возможность генерации программного кода после построения модели системы. Возможность генерации текста программы на том или ином языке программирования зависит от установленной версии Rational Rose. Общая



последовательность действий, которые необходимо выполнить для этого, состоит в следующем:

1. Проверка модели независимо от выбора языка генерации кода.
2. Создание компонентов для реализации классов.
3. Отображение классов на компоненты.
4. Установка свойств генерации программного кода.
5. Выбор класса, компонента или пакета.
6. Генерация программного кода.

Особенности выполнения каждого из этапов могут изменяться в зависимости от выбранного языка программирования. В работе рассматривается универсальный вариант создания кода на языке C++, который независим от используемого компилятора.

Генерация кода в этом случае может быть выполнена двумя способами на основе диаграммы классов и на основе диаграммы компонентов.

В первом случае на диаграмме классов выбирается соответствующий класс и для него командой меню **Tools-»C++-»Code Generation** устанавливается возможность получения кода. В появившемся диалоговом окне нажимается кнопка **Assign** (назначить). Для просмотра результата генерации на выбранном классе открывается контекстное меню, в котором доступны команды **C++-»Browse Header** (просмотр заголовочного файла) и **C++-»Browse Body** (просмотр файла тела класса).

Во втором случае в контекстном меню выбранного компонента сначала выполняется команда **C++-»Code Generation**, а затем команды просмотра заголовочного файла и файла тела класса. При этом компоненты должны быть предварительно специфицированы.

## 2. Практическая часть.

**Постановка задачи.** Пусть требуется разработать программное обеспечение для моделирования работы банкомата в банковской сети. Банкомат принимает от клиента карту, взаимодействует с клиентом, соединяется с банком для аутентификации клиента, позволяет выполнять просмотр баланса, выдачу наличных, оплату сотовой связи и перевод денег с одного счета на другой. При запросе клиентом суммы больше присутствующей на счете для VIP-клиентов возможно получение кредита.

2.1. Запустим систему Rational Rose, выполнив **Пуск-»Программы-» Rational Rose Enterprise Edition -» Rational Rose Enterprise Edition**.

2.2. При появлении окна **Create New Model** (Создать новую модель) нажмем **Cancel** (Отмена), тем самым отказываясь от выбора типового проекта.

Обратите внимание, что для упрощения освоения системы Rational Rose для наименований элементов во всех далее разрабатываемых диаграммах используются русские буквы. При генерации кода программы такие наименования отображаться не будут. Поэтому при выполнении лабораторной работы предлагается использовать один из двух следующих вариантов:

- изначально использовать только наименования на английском языке с учетом ограничений, накладываемых на идентификаторы в языках программирования;

- использовать наименования на русском языке, но перед генерацией кода переименовать элементы.

2.3. Построим *диаграмму вариантов использования* для системы «Банкомат» с учетом следующих условий:

- действующие лица (актеры) – «Клиент», «Банк»;

- действия, выполняемые моделируемой системой (варианты использования): «Снять деньги со счета», «Просмотреть баланс», «Оплатить сотовую связь», «Перевести деньги на другой счет».

2.3.1. Между актерами и вариантами использования установим необходимые связи. В результате должна получиться диаграмма, представленная на рис. 2.1.

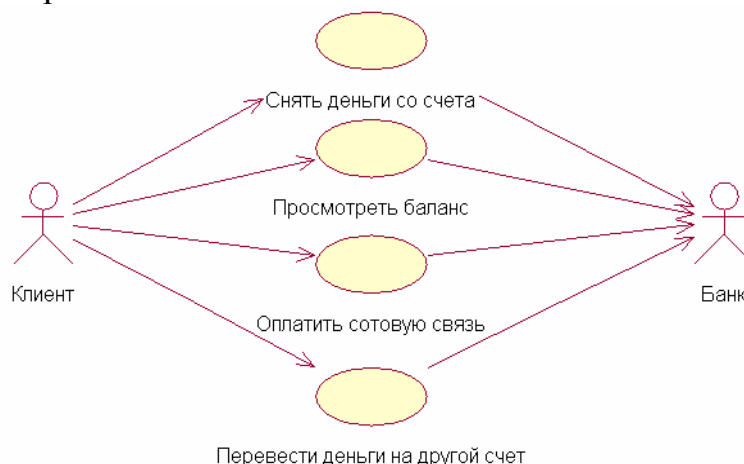


Рис. 2.1. Диаграмма вариантов использования

2.3.2. Разработаем описательную спецификацию для варианта использования «Снять деньги со счета».

### Описательная спецификация для ВИ «Снять деньги со счета».

Актеры: Клиент, Банк.

Цель: Получение требуемой суммы наличными.

Краткое описание: Клиент запрашивает требуемую сумму. Банкомат обеспечивает доступ к счету клиента. Банкомат выдает клиенту наличные.

Предусловие: Банкомат свободен.

Действие актеров	Отклик системы
1. Клиент вставляет карточку в банкомат.	
	2. Банкомат предлагает ввести PIN-код.
3. Клиент вводит PIN-код.	
	4. Банкомат делает запрос в банк с целью проверки PIN-кода.
5. Банк проверяет PIN-код. Если PIN-код неправильный, то выполняется Исключение 1.	
	6. Банкомат выводит список допустимых операций: – снять деньги со счета; – просмотреть баланс.
7. Клиент выбирает операцию «Снять деньги со счета».	
	8. Банкомат предлагает ввести сумму.
9. Клиент вводит сумму.	
	10. Банкомат делает запрос в банк с целью проверки платежеспособности клиента.
11. Банк выясняет состояние счета клиента. Если средств на счете недостаточно, то выполняется Исключение 2.	
	12. Банкомат делает запрос в банк на снятие нужной суммы со счета клиента.
13. Банк изменяет состояние счета клиента.	
	14. Банкомат выдает клиенту наличные.
	15. Банкомат возвращает клиенту карточку. ВИ завершается.

Исключение 1. Неправильный PIN-код

Действие актеров	Отклик системы
	1. Банкомат информирует клиента, что PIN-код неправильный.
	2. Банкомат возвращает клиенту карточку. ВИ завершается.

Исключение 2. Недостаточно денег на счете

Действие актеров	Отклик системы
	1. Банкомат информирует клиента, что денег на его счете недостаточно.
	2. Банкомат делает запрос в банк о возможности выдачи кредита.
3. Банк проверяет возможность выдачи кредита. Если клиент является VIP-клиентом, то выполняется переход к п. 12. основного потока событий. Если клиент обычный, то выполняется Исключение 3.	

## Исключение 3. Невозможность выдачи кредита

Действие актеров	Отклик системы
	1. Банкомат информирует клиента, что получение кредита невозможно.
	2. Банкомат возвращает клиенту карточку. ВИ завершается.

Сохраним описательную спецификацию в файле и закрепим за вариантом использования «Снять деньги со счета».

2.3.3. На основе разработанных описательных спецификаций можно выделить дополнительные варианты использования:

- общий вариант использования «Проверка PIN-кода»;
- варианты использования «Неправильный PIN-код», «Недостаточно денег на счете» и «Невозможность выдачи кредита», расширяющие функциональность основных вариантов использования.

2.3.4. На основе разработанной описательной спецификации можно выделить два дополнительных типа актера «Клиент»:

- «Обычный клиент»;
- «VIP-клиент».

2.3.5. Уточним диаграмму вариантов использования, добавив на нее дополнительные варианты использования и дополнительных актеров. В результате получим диаграмму, приведенную на рис. 2.2.

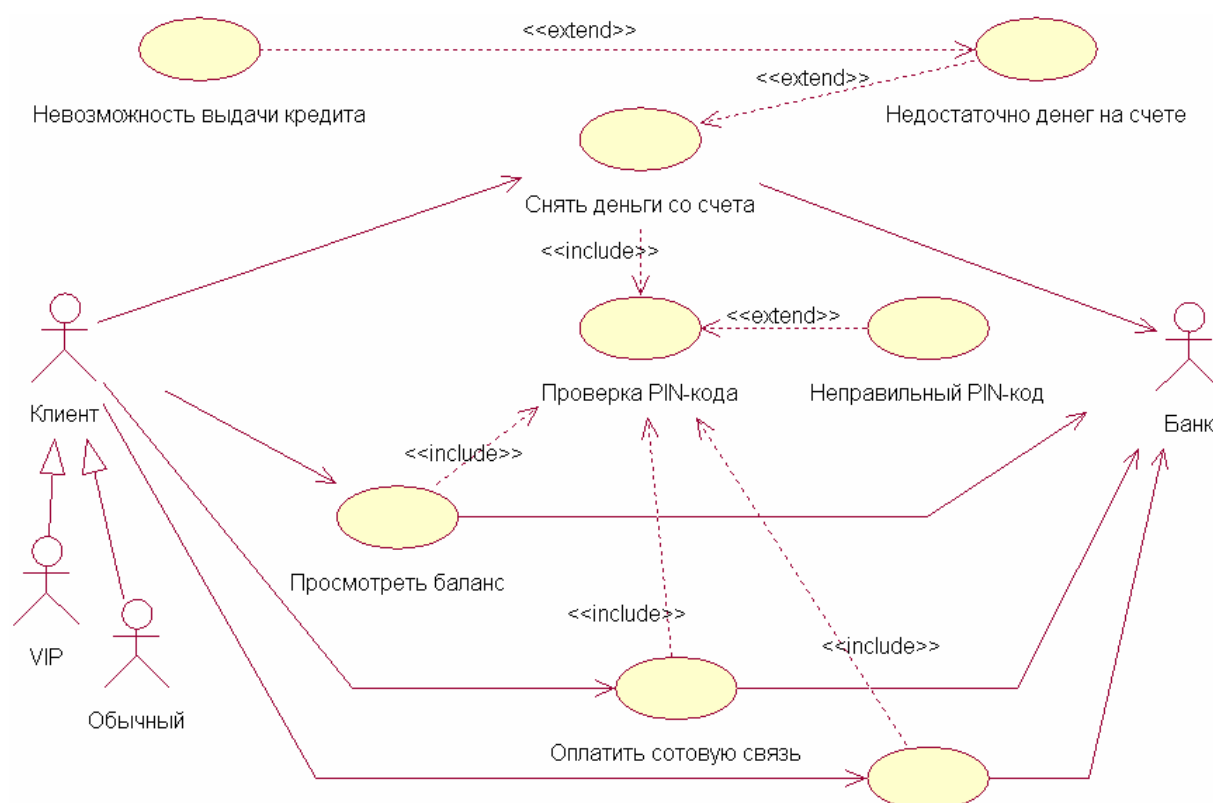


Рис. 2.2. Уточненная диаграмма вариантов использования

2.3.6. Для вариантов использования «Просмотреть баланс», «Оплатить сотовую связь», «Перевести деньги на другой счет» описательные спецификации разработать самостоятельно.

2.3.7. При необходимости после разработки описательных спецификаций уточнить диаграмму вариантов использования, добавив на нее дополнительные варианты использования и дополнительных актеров.

2.4. Построим *диаграмму классов* для системы «Банкомат».

2.4.1. На диаграмме отобразим следующие классы:

1. «Контроллер банкомата».
2. «УЧК» (Устройство чтения карт).
3. «Терминал».
4. «Кассовый аппарат».
5. «Контроллер банка».

2.4.2. Для определения возможных направлений передачи сообщений установим связи типа однонаправленной ассоциации (символ  $\rightarrow$ ) между классами по следующей схеме:

- $1 \Rightarrow 2$ ;
- $1 \Rightarrow 3$ ;
- $1 \Rightarrow 4$ ;
- $1 \Rightarrow 5$ .

На связях зададим кратности отдельных классов:

- на связях  $1 \Rightarrow 2$ ,  $1 \Rightarrow 3$ ,  $1 \Rightarrow 4$  кратности обоих классов равны 1, так как одному контроллеру банкомата будет соответствовать по одному устройству: УЧК, терминал, кассовый аппарат;
- на связи  $1 \Rightarrow 5$  кратность класса «Контроллер банкомата» равна 0..n, так как контроллер банка может быть связан с любым количеством банкоматов, а кратность класса «Контроллер банка» равна 1, так как контроллер банкомата всегда связан только с одним контроллером банка.

В результате должна получиться диаграмма, представленная на рис. 2.3.

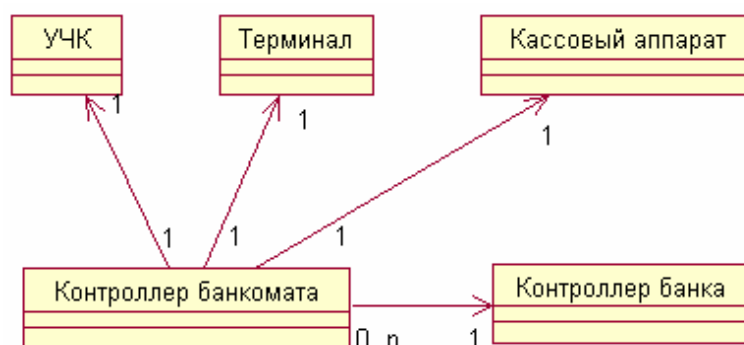


Рис. 2.3. Диаграмма классов

2.4.3. Для классов задать следующие стереотипы:

1. Для класса «Контроллер банкомата» стереотип «Control».
2. Для класса «УЧК» стереотип «Boundary».
3. Для класса «Терминал» стереотип «Boundary».
4. Для класса «Кассовый аппарат» стереотип «Boundary».
5. Для класса «Контроллер банка» стереотип «Interface».

В результате должна получиться диаграмма, представленная на рис. 2.4.

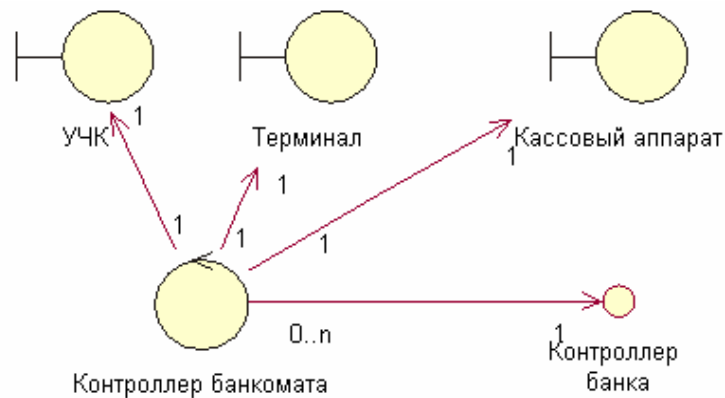


Рис. 2.4. Диаграмма классов со стереотипами для классов

Для более удобного просмотра операций и атрибутов классов следует изменить способ отображения классов на диаграмме. После выполнения данной команды для всех классов диаграмма примет вид, приведенный на рис. 2.5.

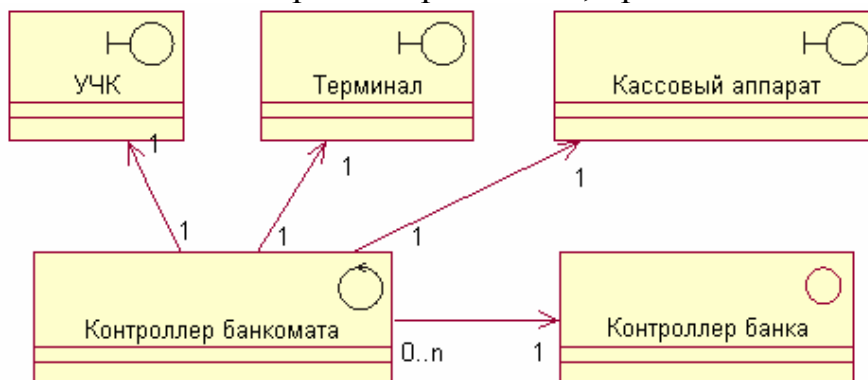


Рис. 2.5. Диаграмма классов

2.5. Построим *диаграмму последовательности* для основного потока варианта использования «Снять деньги со счета».

2.5.1. На диаграмме отобразим необходимые классы:

1. «Клиент».
2. «Контроллер банкомата».
3. «УЧК».
4. «Терминал».
5. «Кассовый аппарат».
6. «Контроллер банка».

Следует обратить внимание, что для размещения классов на диаграмме последовательности их можно перетащить из окна браузера.

2.5.2. На основе разработанных описательных спецификаций можно определить взаимодействие классов, выделив последовательность обмена сообщениями между ними. Разработаем диаграмму последовательности для основного варианта использования «Снять деньги со счета».

Добавим на диаграмму последовательностей следующие сообщения:

1. Сообщение «Карта вставлена», направленное от класса «Клиент» к классу «Контроллер банкомата». Тип сообщения: асинхронное. Параметры: нет. Возвращаемый результат: нет.
2. Сообщение «Читать карту», направленное от класса «Контроллер банкомата» к классу «УЧК». Тип сообщения: вызов процедуры. Параметры: нет. Возвращаемый результат: номер карты типа integer.

3. Сообщение «Возврат номера карты», направленное от класса «УЧК» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

4. Сообщение «Запросить PIN», направленное от класса «Контроллер банкомата» к классу «Терминал». Тип сообщения: вызов процедуры. Параметры: нет. Возвращаемый результат: PIN-код типа integer.

5. Сообщение «Ввод PIN», направленное от класса «Клиент» к классу «Терминал». Тип сообщения: асинхронное. Параметры: PIN-код типа integer. Возвращаемый результат: нет.

6. Сообщение «Возврат PIN», направленное от класса «Терминал» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

7. Сообщение «Проверить PIN», направленное от класса «Контроллер банкомата» к классу «Контроллер банка». Тип сообщения: вызов процедуры. Параметры: Номер карты типа integer, PIN-код типа integer. Возвращаемый результат: Результат проверки типа boolean.

8. Сообщение «Возврат результата проверки», направленное от класса «Контроллер банка» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

9. Сообщение «Запросить операцию», направленное от класса «Контроллер банкомата» к классу «Терминал». Тип сообщения: вызов процедуры. Параметры: нет. Возвращаемый результат: Код операции типа integer.

10. Сообщение «Выбор операции», направленное от класса «Клиент» к классу «Терминал». Тип сообщения: асинхронное. Параметры: Код операции типа integer. Возвращаемый результат: нет.

11. Сообщение «Возврат кода операции», направленное от класса «Терминал» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

12. Сообщение «Запросить сумму», направленное от класса «Контроллер банкомата» к классу «Терминал». Тип сообщения: вызов процедуры. Параметры: нет. Возвращаемый результат: Сумма типа integer.

13. Сообщение «Ввод суммы», направленное от класса «Клиент» к классу «Терминал». Тип сообщения: асинхронное. Параметры: Сумма типа integer. Возвращаемый результат: нет.

14. Сообщение «Возврат суммы», направленное от класса «Терминал» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

15. Сообщение «Проверить баланс», направленное от класса «Контроллер банкомата» к классу «Контроллер банка». Тип сообщения: вызов процедуры. Параметры: Номер карты типа integer, Сумма типа integer. Возвращаемый результат: Результат проверки типа boolean.

16. Сообщение «Возврат результата проверки», направленное от класса «Контроллер банка» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

17. Сообщение «Снять средства», направленное от класса «Контроллер банкомата» к классу «Контроллер банка». Тип сообщения: вызов процедуры. Параметры: Номер карты типа integer, Сумма типа integer. Возвращаемый результат: нет.

18. Сообщение «Возврат из процедуры», направленное от класса «Контроллер банка» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

19. Сообщение «Выдать наличные», направленное от класса «Контроллер банкомата» к классу «Кассовый аппарат». Тип сообщения: вызов процедуры. Параметры: Сумма типа integer. Возвращаемый результат: нет.

20. Сообщение «Возврат из процедуры», направленное от класса «Кассовый аппарат» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

21. Сообщение «Выдать карту», направленное от класса «Контроллер банкомата» к классу «УЧК». Тип сообщения: вызов процедуры. Параметры: нет. Возвращаемый результат: нет.

22. Сообщение «Возврат из процедуры», направленное от класса «УЧК» к классу «Контроллер банкомата». Тип сообщения: возврат из вызова процедуры.

В результате должна получиться диаграмма, представленная на рис. 2.6.

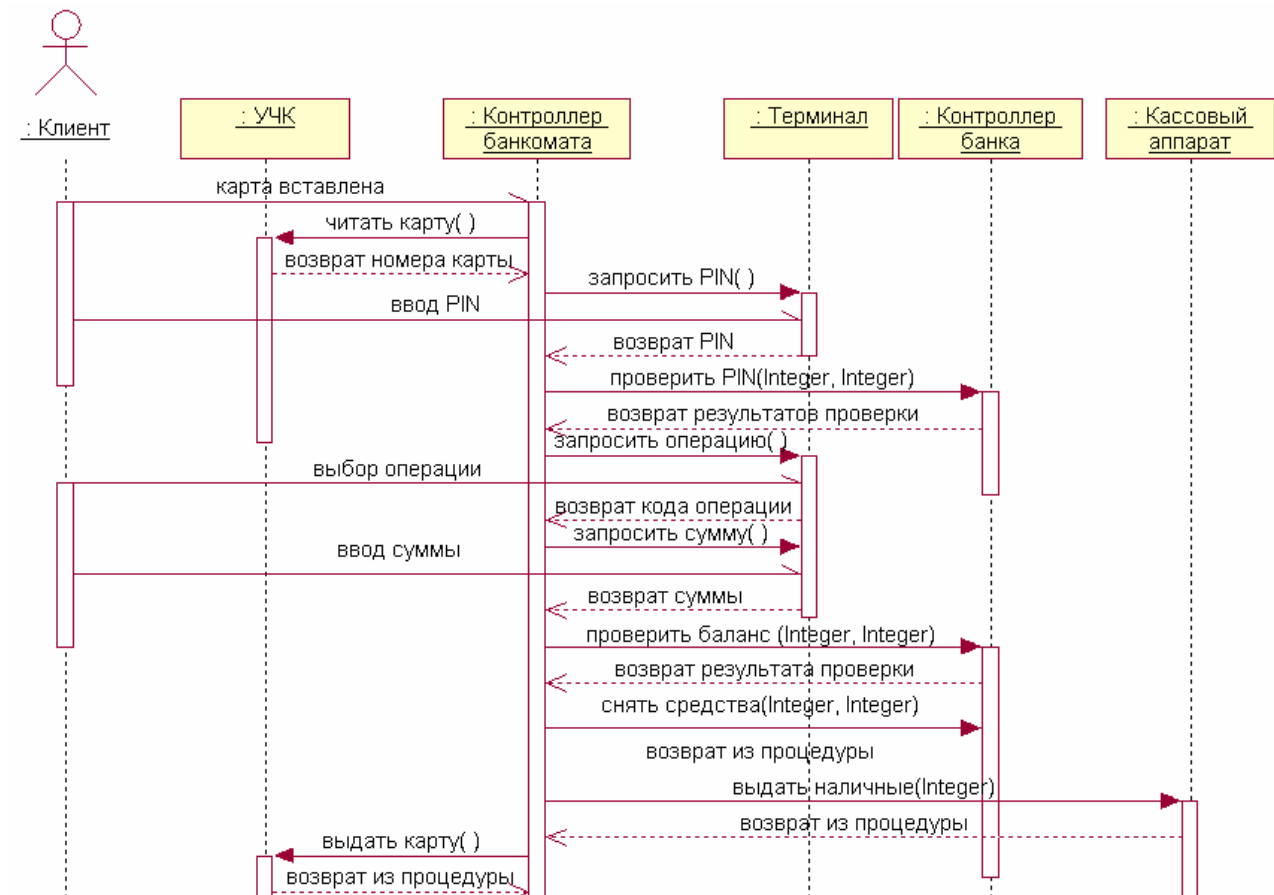


Рис. 2.6. Диаграмма последовательности



2.5.3. Диаграммы последовательности для основных вариантов использования: «Просмотреть баланс», «Оплатить сотовую связь», «Перевести деньги на другой счет» следует разработать самостоятельно.

Для дополнительных вариантов использования, таких как «Неправильный PIN-код», «Недостаточно денег на счете» и «Невозможность выдачи кредита» и других диаграмм, выделенных при моделировании, также следует разработать отдельные диаграммы последовательности.

2.6. По разработанным диаграммам последовательностей система Rational Rose автоматически построит диаграммы кооперации. Переход между диаграммами осуществляется по нажатию клавиши F5. Полученная диаграмма кооперации для варианта использования «Снять деньги со счета» представлена на рис. 2.7.

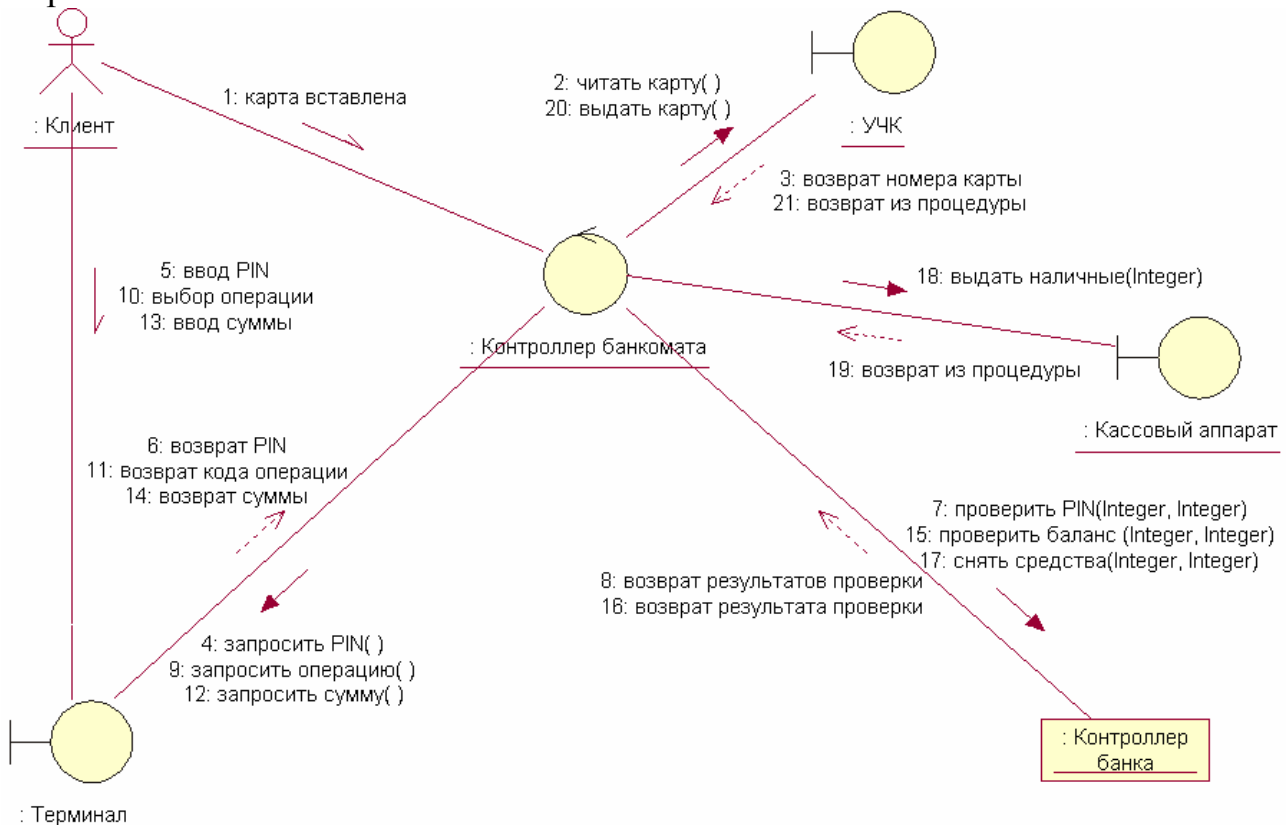


Рис. 2.7. Диаграмма кооперации

2.7. Просмотрим диаграмму классов. Обратите внимание, что описание сообщений, определенных как операции классов при разработке диаграммы последовательности, автоматически присоединяется к соответствующим классам.

2.7.1. Добавим классу «Контроллер банкомата» следующие атрибуты:

1. Номер карты типа Integer;
2. PIN-код типа Integer;
3. Сумма типа Integer.

Полученная диаграмма классов приведена на рис. 2.8.

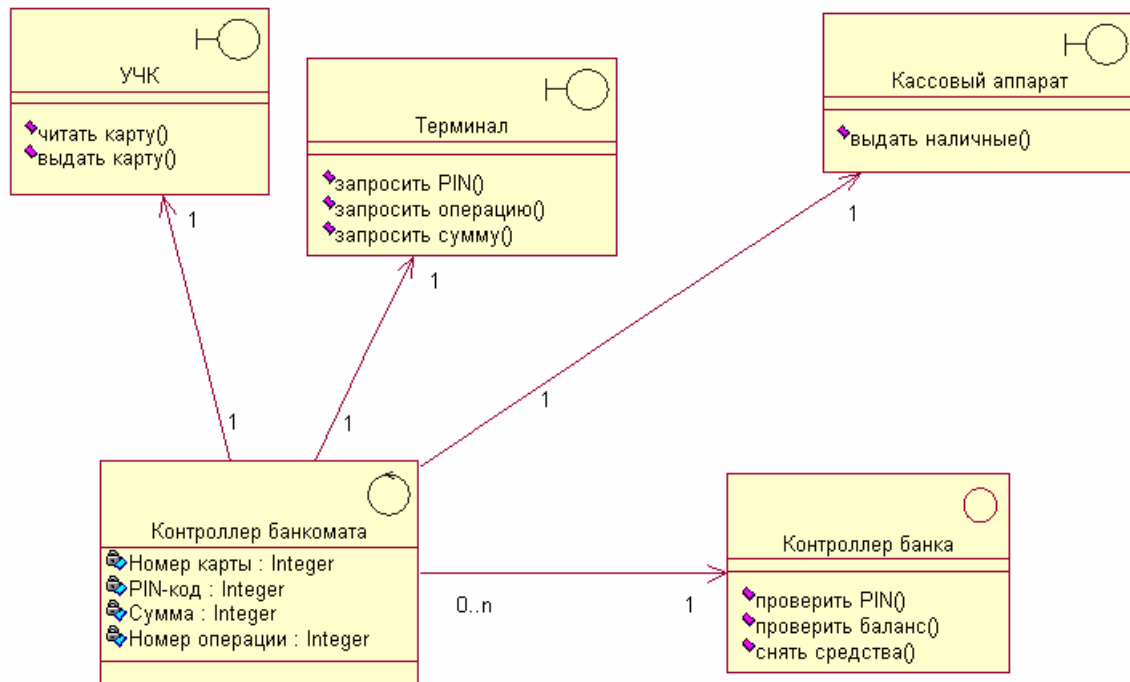


Рис. 2.8. Диаграмма классов

2.8. Построим *диаграмму состояний* для системы «Банкомат» в целом.

2.8.1. Можно выделить следующие состояния системы:

1. Банкомат включен (начальное состояние).
2. Ожидание карты.
3. Ожидание ввода PIN.
4. Ожидание выбора операции.
5. Выполнение операции «Снять деньги со счета».
6. Выполнение операции «Просмотр баланса».
7. Выполнение операции «Оплатить сотовую связь»
8. Выполнение операции «Перевести деньги на другой счет»
9. Банкомат выключен (конечное состояние).

Добавим выделенные состояния на диаграмму.

2.8.2. Уточним состояния, добавив к ним список внутренних действий:

1. Состояние «Ожидание карты» – входное действие «Вывести приглашение».
2. Состояние «Ожидание ввода PIN» – входное действие «Вывести запрос PIN», выходное действие «Проверить PIN».
3. Состояние «Ожидание выбора операции» – входное действие «Вывести меню».
4. Состояние «Выполнение операции «Снять деньги» – do-деятельность «Снятие денег».
5. Состояние «Выполнение операции «Просмотр баланса» – do-деятельность «Просмотр баланса».
6. Состояние «Выполнение операции «Оплатить сотовую связь» – do-деятельность «Оплата сотовой связи».
7. Состояние «Выполнение операции «Перевести деньги на другой счет» do-деятельность «Перевод денег на другой счет».

2.8.3. Уточним диаграмму состояний, добавив переходы между состояниями, определим тип и спецификацию каждого перехода:

1. Переход от начального состояния к состоянию «Ожидание карты».

Тип: нетриггерный.

Имя события: Банкомат включен.

2. Переход от состояния «Ожидание карты» к состоянию «Ожидание ввода PIN».

Тип: нетриггерный.

Имя события: Карта вставлена.

Выражение действия: Читать карту.

3. Переход от состояния «Ожидание ввода PIN» к состоянию «Ожидание карты».

Тип: нетриггерный.

Имя события: PIN введен.

Сторожевое условие: PIN неправильный.

Выражение действия: Выдать карту. Вывести сообщение

4. Переход от состояния «Ожидание ввода PIN» к состоянию «Ожидание выбора операции».

Тип: нетриггерный.

Имя события: PIN введен.

Сторожевое условие: PIN правильный.

5. Переход от состояния «Ожидание выбора операции» к состоянию «Выполнение операции «Снять деньги».

Тип: нетриггерный.

Имя события: Операция выбрана.

Сторожевое условие: Операция = «Снять деньги».

6. Переход от состояния «Ожидание выбора операции» к состоянию «Выполнение операции «Просмотр баланса».

Тип: нетриггерный.

Имя события: Операция выбрана.

Сторожевое условие: Операция = «Просмотр баланса».

7. Переход от состояния «Ожидание выбора операции» к состоянию «Выполнение операции «Оплатить сотовую связь».

Тип: нетриггерный.

Имя события: Операция выбрана.

Сторожевое условие: Операция = «Оплатить сотовую связь».

8. Переход от состояния «Ожидание выбора операции» к состоянию «Выполнение операции «Перевести деньги на другой счет».

Тип: нетриггерный.

Имя события: Операция выбрана.

Сторожевое условие: Операция = «Перевести деньги на другой счет».

9. Переход от состояния «Выполнение операции «Снять деньги» к состоянию «Ожидание карты».

Тип: триггерный.

Выражение действия: Выдать карту.

10. Переход от состояния «Выполнение операции «Просмотр баланса» к состоянию «Ожидание карты».

Тип: триггерный.

Выражение действия: Выдать карту.

11. Переход от состояния «Выполнение операции «Оплатить сотовую связь» к состоянию «Ожидание карты».

Тип: триггерный.

Выражение действия: Выдать карту.

12. Переход от состояния «Выполнение операции «Перевести деньги на другой счет» к состоянию «Ожидание карты».

Тип: триггерный.

Выражение действия: Выдать карту.

13. Переход от состояния «Ожидание карты» к конечному состоянию.

Тип: нетриггерный.

Имя события: Банкомат выключен.

Полученная диаграмма состояний приведена на рис. 2.9.

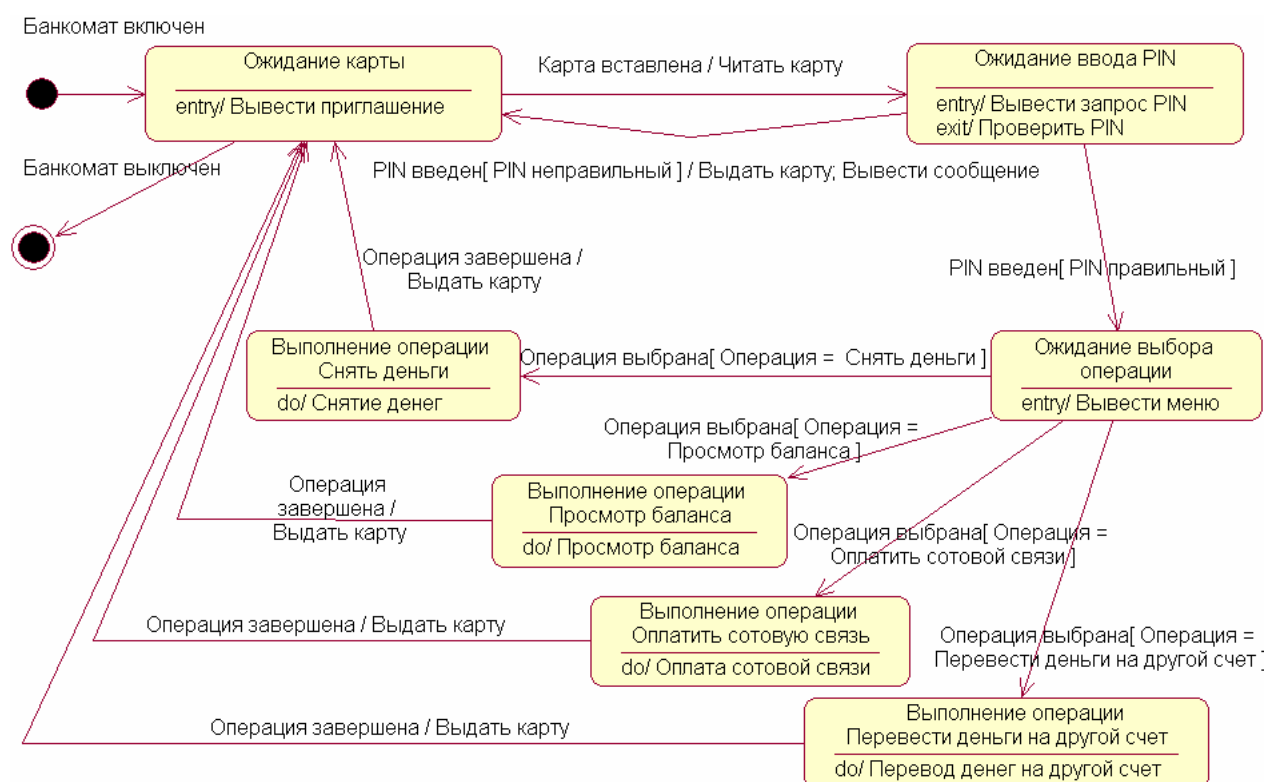


Рис. 2.9. Диаграмма состояний

2.9. Для детализации особенностей реализации выполняемых системой операций строят *диаграмму деятельности*.

2.9.1. Пусть при уточнении варианта использования «Проверка PIN-кода» стало известно, что должны выполняться два требования:

1. Время ожидания ввода PIN-кода ограничено – не более 5-ти минут.
2. Число попыток ввода PIN-кода ограничено – не более 3-х попыток.

Построим *диаграмму деятельности* для варианта использования «Проверка PIN-кода» с учетом вышеуказанных требований. Полученная диаграмма деятельности приведена на рис. 2.10.

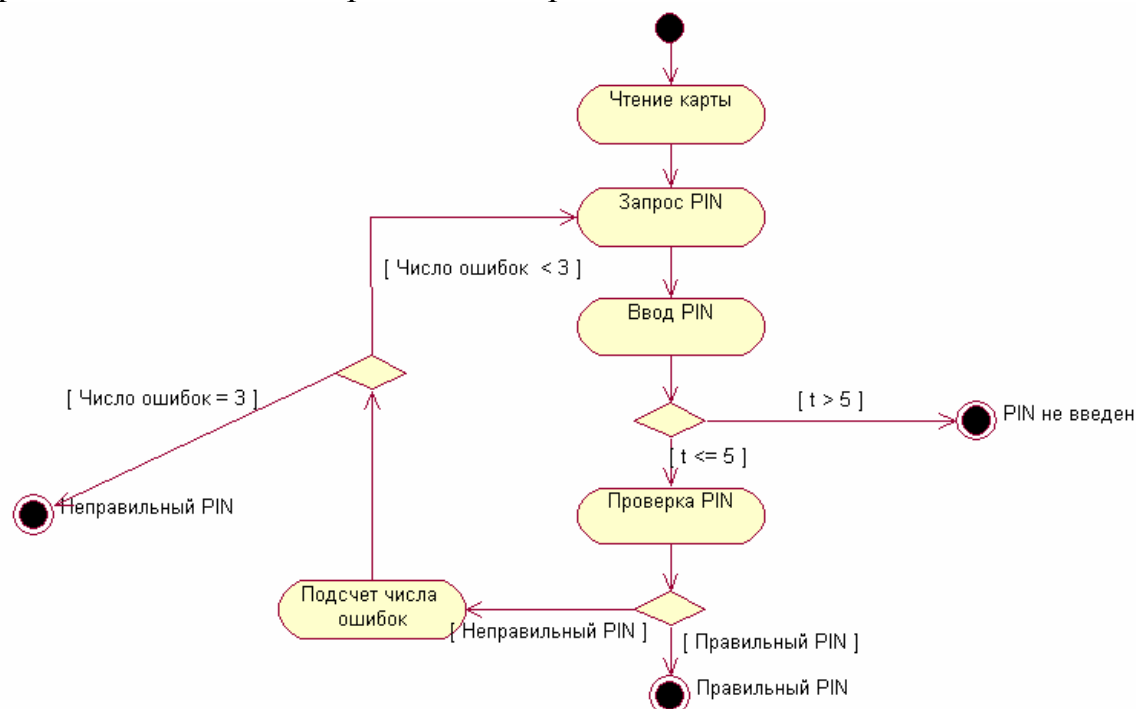


Рис. 2.10. Диаграмма деятельности

2.9.2. Уточним диаграмму деятельности, выделив ответственность классов за выполняемую деятельность.

Деятельность по проверке PIN-кода выполняют следующие классы:

1. УЧК.
2. Контроллер банкомата.
3. Терминал.
4. Контроллер банка.

Для каждого из перечисленных классов создадим отдельную дорожку ответственности, которую назовем именем класса, и поместим на нее соответствующую деятельность.

Полученная диаграмма приведена на рис. 2.11.

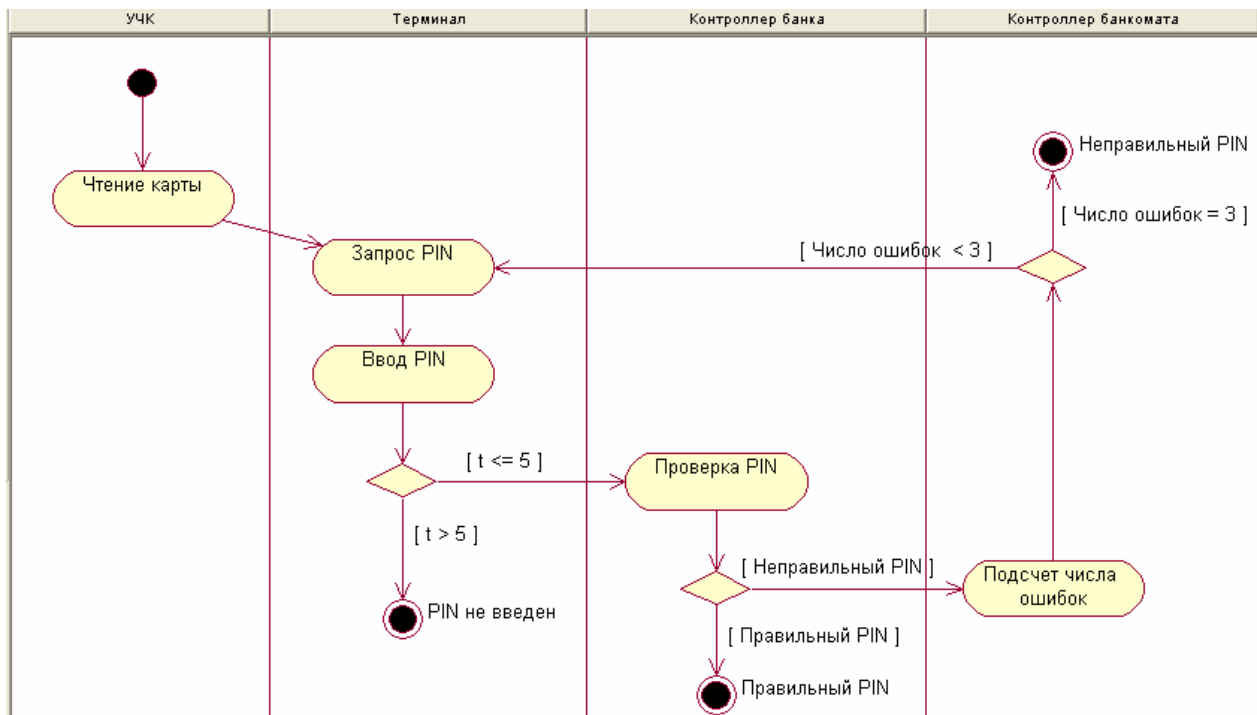


Рис. 2.11. Диаграмма деятельности с дорожками ответственности

2.10. Для проектирования физических особенностей представления системы построим *диаграмму компонентов*.

2.10.1. Будем считать, что для реализации используется язык C++.

Пусть все устройства банкомата реализованы в отдельной библиотеке Device.dll, а программный код для реализации классов данной библиотеки находится в файлах Device.h и Device.cpp. Предположим, что система снабжена справкой – файл Help.hlp. Основная программа, реализующая контроллер банкомата, называется Main.exe. Текст данной программы находится в файлах Main.h и Main.cpp. Взаимодействие банкомата с контроллером банка будет осуществляться через компонент MainBank.exe, который использует базу данных с информацией о клиентах банка - Client.mdf. Отметим, что компонент MainBank.exe не является предметом разработки в нашей системе.

Полученная диаграмма приведена на рис. 2.12.

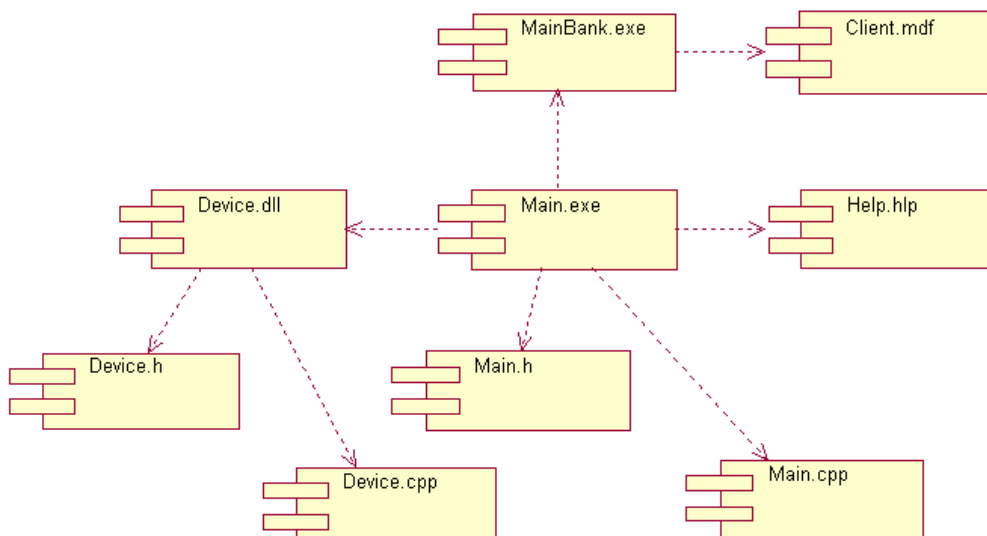


Рис. 2.12. Диаграмма компонентов

2.11. Для системы «Банкомат» построим *диаграмму развертывания*, которая будет показывать физические взаимосвязи между программными и аппаратными компонентами разрабатываемой системы.

2.11.1. Пусть банк имеет три банкомата, размещенные по следующим адресам: Гагарина 59/1, Есенина 71 и Урицкого 15. Соединение банка с банкоматами осуществляется по сети.

Тогда диаграмма развертывания будет выглядеть следующим образом (рис. 2.13):

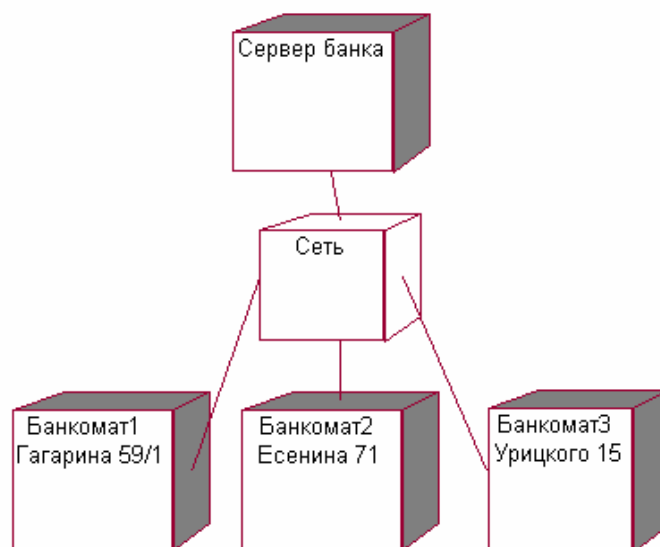


Рис. 2.13. Диаграмма развертывания

2.12. По разработанной ранее диаграмме классов сгенерируем фрагменты кода для классов Терминал и Контроллер банкомата. В качестве языка программирования выберем язык C++.

По умолчанию на диске, где установлен Rational Rose, в папке Program Files\Rational\Rose\C++\source\ будут сформированы файлы с расширением .h и .cpp, содержащие объявление классов и заготовки для описания методов классов. После этого их можно просмотреть, открыв соответствующий файл или выполнив команду контекстного меню C++\Browse Header или C++\Browse Body.

## Вариант 1. Цифровой диктофон

Требуется разработать средствами Rational Rose модель программного обеспечения, управляющего работой цифрового диктофона.

Цифровой диктофон - это бытовое электронное устройство, предназначенное для записи и воспроизведения речи. Звуковые сообщения записываются через встроенный микрофон и сохраняются в памяти устройства. Сообщения воспроизводятся через встроенный громкоговоритель. Работа устройства осуществляется под управлением центрального процессора.

Примерный внешний вид устройства изображен на рисунке 1.

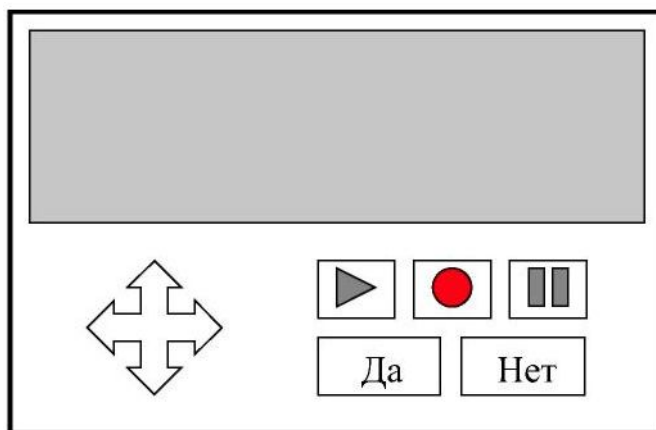


Рис. 1. Внешний вид диктофона

Диктофон хранит до 10 звуковых сообщений. Длина каждого сообщения ограничена размером свободной памяти. Диктофон осуществляет прямой (по номеру сообщения) доступ к любому сообщению из памяти. Пользователь имеет возможность воспроизводить сообщения, хранящиеся в памяти диктофона, стирать их, записывать новые. Исполнителем должна быть разработана схема базы данных для хранения сообщений в памяти диктофона.

Интерфейс с пользователем осуществляется при помощи экранного меню и управляющих кнопок на корпусе диктофона. При помощи кнопок-стрелок осуществляется навигация по пунктам меню. Кнопки «Да», «Нет» служат для подтверждения или отмены пользователем выбора той или иной опции меню (структуру меню исполнитель должен разработать самостоятельно). Имеются также кнопки «Воспроизведение», «Пауза» и «Запись» для работы со звуковыми сообщениями.



Во время записи сообщения на экране отображается время, в течение которого ведется запись, при воспроизведении - длительность воспроизведенной части сообщения.

Если диктофон не используется, через 30 секунд он автоматически переходит в режим сбережения энергии. В этом режиме никакие операции над звуковыми сообщениями не возможны. Энергия расходуется только на сохранение памяти диктофона в неизменном состоянии. Переход из режима сбережения энергии в обычный режим осуществляется при нажатии пользователем любой кнопки.

В диктофоне имеется датчик уровня заряда батарей. При падении уровня заряда ниже установленного предела диктофон автоматически переходит в режим сбережения энергии (независимо от того используется он в данный момент или нет). Переход в обычный режим становится возможным только после восстановления нормального уровня заряда батарей.

## Вариант 2. Торговый автомат

Требуется разработать средствами Rational Rose модель программного обеспечения встроенного процессора универсального торгового автомата.

Внешний вид автомата изображен на рисунке 2. В автомате имеется пять лотков для хранения и выдачи товаров. Загрузка товаров на лотки осуществляется обслуживающим персоналом. Автомат следит за наличием товара. Если какой-либо товар распродан, автомат отправляет сообщение об этом на станцию обслуживания и информирует покупателей (зажигается красная лампочка рядом с лотком данного товара).

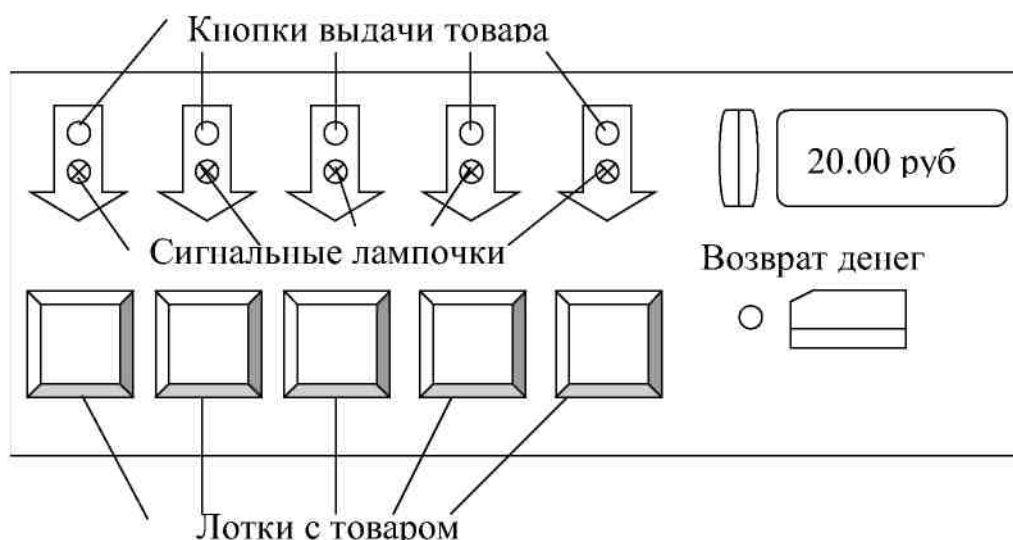


Рис. 2. Лицевая панель торгового автомата

Автомат принимает к оплате бумажные купюры и монеты. Специальный индикатор высвечивает текущую сумму денег, принятых автоматом к оплате. После ввода денег клиент нажимает на кнопку выдачи товара. Выдача товара производится только в том случае, если введенная сумма денег соответствует цене товара. Товар выдается поштучно. При нажатии на кнопку «Возврат» клиенту возвращаются все принятые от него к оплате деньги. Возврат денег не производится после выдачи товара. Автомат должен корректно работать при одновременном нажатии на кнопки выдачи товара и возврата денег.

В специальном отделении автомата, закрываемом замком, есть «секретная кнопка», которая используется обслуживающим персоналом для выемки выручки. При нажатии на эту кнопку открывается доступ к ящику с деньгами.

Автомат получает со станции обслуживания данные о товарах и хранит их в своей памяти. Данные включают в себя цену, наименование товара, номер лотка, на котором находится товар и количество товара на лотке. Вариант задания включает в себя разработку схемы базы данных о товарах.

### **Вариант 3. Табло на станции метро**

Требуется разработать средствами Rational Rose модель программного обеспечения табло для информационной службы метрополитена.

Табло расположены на каждой станции метро. Они работают под управлением единого пункта управления (ПУ) информационной службы метро. Табло отображает текущее время (часы, минуты, секунды) и время, прошедшее с момента отправления последнего поезда (минуты, секунды). Момент прибытия и отправления поезда определяется при помощи датчиков, устанавливаемых на путях. Все табло метро синхронизованы, текущее время отсчитывается и устанавливается из центральной службы времени, находящейся на ПУ.

На табло высвечивается конечная станция назначения прибывающего поезда. Эти данные содержатся в расписании движения поездов, которое хранится в памяти табло и периодически обновляется с ПУ.

В «бегущей строке» табло отображается рекламная информация. Память табло хранит до 10 рекламных сообщений. Сообщения отображаются друг за другом с небольшими паузами, циклически. Содержание рекламных сообщений поступает с ПУ.

Дополнительная функция табло - по запросу с ПУ оно пересылает данные о нарушениях расписания (преждевременных отправлениях поездов или опозданиях).

В ходе выполнения задания должна быть создана схема базы данных для хранения рекламных сообщений, расписания и сведений о нарушении расписаний.

Пояснение: в задании требуется разработать модель ПО только для табло, но не для пункта управления информационной службы.

#### **Вариант 4. Система автоматизации для пункта проката видеокассет**

Требуется разработать средствами Rational Rose модель программной системы автоматизации работы пункта проката видеокассет (далее в тексте - системы).

Пункт проката содержит каталог кассет, имеющихся в наличии в данный момент времени. Система поддерживает работу каталога, позволяя служащим проката добавлять новые наименования кассет, удалять старые и редактировать данные о кассетах.

Клиент, обратившийся в пункт, выбирает кассету по каталогу, вносит залог и забирает ее на определенный срок. Срок проката, измеряемый в сутках, оговаривается при выдаче кассеты. Стоимость проката вычисляется системой исходя из тарифа за сутки и срока проката. Клиент возвращает кассету и оплачивает прокат. Если кассета не повреждена, клиенту возвращается залог. Служащий пункта проката регистрирует сдачу кассеты клиенту и ее возврат в системе. Если клиент повредил кассету, то кассета удаляется из каталога, а залог остается в кассе проката. При необходимости служащий может запросить у системы следующие данные:

- имеется ли в наличии кассета с данным названием;
- когда будет возвращена какая-либо кассета из тех, что сданы в прокат;
- является ли данный клиент постоянным клиентом пункта проката (пользовался ли прокатом 5 или более раз).

Постоянным клиентам предоставляются скидки, а также от них принимаются заявки на пополнение ассортимента кассет. Заявки регистрируются в системе. По ним готовится итоговый отчет, руководствуясь которым, служащие пункта проката обновляют ассортимент кассет.

Необходимо разработать схему базы данных для хранения каталога, учетных записей о прокате кассет и заявок на пополнение ассортимента.

## Вариант 5. Мини-АТС

Требуется разработать средствами Rational Rose модель программного обеспечения встроенного микропроцессора учрежденческой мини-АТС (автоматической телефонной станции).

Мини-АТС осуществляет связь между служащими учреждения. Каждый абонент подключен к ней линией связи. Мини-АТС соединяет линии абонентов (осуществляет коммутацию линий). Абоненты имеют номера, состоящие из трех цифр. Специальный номер 9 зарезервирован для внешней связи.

Телефонное соединение абонентов производится следующим образом. Абонент поднимает трубку телефона, и мини-АТС получает сигнал «Трубка». В ответ мини-АТС посылает сигнал «Тон». Приняв этот сигнал, абонент набирает телефонный номер (посылает три сигнала «Цифра»). Мини-АТС проверяет готовность вызываемого абонента. Если абонент не готов (его линия занята), мини-АТС посылает вызывающему абоненту сигнал «Занято». Если абонент готов, мини-АТС посылает обоим абонентам сигнал «Вызов». При этом телефон вызываемого абонента начинает звонить, а вызывающий абонент слышит в трубке длинные гудки. Вызываемый абонент снимает трубку, и мини-АТС получает от него сигнал «Трубка», после чего осуществляет коммутацию линии. Абоненты обмениваются сигналами «Данные», которые мини-АТС должна передавать от одного абонента к другому. Когда один из абонентов опускает трубку, мини-АТС получает сигнал «Конец» и посылает другому абоненту сигнал «Тон».

В любой момент абонент может положить трубку, при этом мини-АТС получает сигнал «Конец». После получения этого сигнала сеанс обслуживания абонента завершается.

Если абонент желает соединиться с абонентом за пределами учреждения, то он набирает номер «9». Мини-АТС посылает по линии, соединяющей с внешней (городской) АТС, сигнал «Трубка» и в дальнейшем служит посредником между телефоном абонента и внешней АТС. Она принимает и передает сигналы и данные между ними, не внося никаких изменений. Единственное исключение касается завершения сеанса. Получив от городской АТС сигнал «Конец», мини-АТС посылает абоненту сигнал «Тон», и ждет

сигнала «Конец» для завершения обслуживания абонента. Если же вызывавший абонент первым вешает трубку, то мини-АТС получает сигнал «Конец», передает его городской АТС и завершает сеанс.

Мини-АТС может получить сигнал «Вызов» от городской АТС. Это происходит, когда нет соединений с внешними абонентами. Сигнал «Вызов» от городской АТС передается абоненту с кодом «000». Только этот абонент может отвечать на внешние звонки.

## Вариант 6. Телефон

Требуется разработать средствами Rational Rose модель программного обеспечения встроенного микропроцессора для аппарата учреденческой телефонной сети.

Аппарат подключен к линии связи, ведущей к мини-АТС. В его задачу входит прием и передача сигналов (в том числе и голосовых данных) мини-АТС. Аппарат имеет кнопочную панель управления, экран для отображения набираемых номеров, звонок и трубку, в которую встроены микрофон и громкоговоритель.

В начальном состоянии трубка телефона повешена, телефон не реагирует на нажатия кнопок. Телефон реагирует только на сигнал «Вызов» от мини-АТС, при этом включается звонок.

При снятии трубки на АТС подается сигнал «Трубка». При получении ответного сигнала «Тон» от АТС телефон воспроизводит звуковой тон «Готов» (длинный непрекращающийся гудок) в трубку. При получении сигнала «Занято», в трубке воспроизводится тон «Занято» (частые короткие гудки).

Пользователь, слыша в трубке тон «Готов», набирает трехзначный номер. Номер может быть набран при помощи кнопок с цифрами или нажатием на специальную кнопку « # ». При нажатии на кнопку с цифрой соответствующий ей сигнал «Цифра» передается АТС. Нажатия на кнопки с цифрами после третьего игнорируются. Во время набора номера введенные цифры отображаются на экране. Последний полностью набранный номер запоминается в памяти аппарата для того, чтобы можно было его воспроизвести при нажатии на кнопку « # ». При нажатии на эту кнопку номер из памяти аппарата высвечивается на экране, и АТС передается последовательность из трех сигналов «Цифра». В ответ на набранный номер от АТС приходит либо сигнал «Занято», либо сигнал «Вызов». При получении сигнала «Вызов» телефон воспроизводит в трубку длинные гудки до того момента, когда АТС осуществит коммутацию и передаст сигнал «Данные».

Телефон воспроизводит данные, передаваемые с сигналом, в трубку. Ответ пользователя воспринимается микрофоном трубки, преобразуется в сигнал



«Данные» и передается АТС. Обмен данными прерывается, если повешена трубка одного из телефонов, участвующих в обмене. О том, что трубку повесил вызываемый абонент, сообщает сигнал «Занято», посылаемый АТС. После того, как трубка аппарата была повешена, телефон посылает АТС сигнал «Конец», и телефон переходит в начальное состояние.

### **Вариант 7. Стиральная машина**

Требуется разработать средствами Rational Rose модель программного обеспечения встроенного микропроцессора стиральной машины.

Машина предназначена для автоматической стирки белья. Машина включает в себя следующие устройства: бак для белья, клапаны для забора и слива воды, мотор, устройство подогрева воды, термометр, таймер, дверца для доступа в бак, несколько емкостей для различных моющих средств, панель управления с кнопками и индикатором. В памяти машины хранятся 5 программ стирки, заданные изготовителем. Пользователи не могут вносить в них изменения. Каждая программа определяет температуру воды, длительность стирки, используемые моющие средства (номер емкости и время подачи), скорость вращения бака во время стирки и отжима. Вариант задания предусматривает разработку схемы базы данных для хранения программ стирки в памяти машины.

Для использования машины необходимо открыть дверцу, поместить белье в бак, поместить моющие средства в емкости, закрыть дверцу, выбрать программу стирки и нажать на кнопку «Пуск». Перед тем как приступить к стирке машина открывает клапан для забора воды, набирает необходимое количество воды, после чего закрывает клапан. Далее, машина действует по выбранной пользователем программе:

- 1) Подогревает, если необходимо, воду до нужной температуры.
- 2) Включает таймер и запускает вращение бака для стирки.
- 3) По таймеру подает в бак моющие средства, предусмотренные программой.
- 4) По окончании стирки сливает воду и запускает отжим.

Во время работы машины на индикаторе высвечивается время, прошедшее с момента запуска (минуты и секунды), текущий режим работы (стирка или отжим), номер текущей программы стирки. В целях безопасности дверца бака блокируется до окончания стирки. Машина не воспринимает нажатий на кнопки, за исключением одной -пользователь имеет возможность в любой момент нажать на кнопку «Останов», чтобы принудительно остановить стирку и слить воду.

## Вариант 8. Таксофон

Требуется разработать средствами Rational Rose модель встроенной системы управления работой таксофона городской телефонной сети.

Таксофон предназначен для оказания платных услуг телефонной связи. Он подключен к линии связи. В нем имеется кнопочная панель, дисплей, трубка со встроенным микрофоном и громкоговорителем, приемник карт - устройство для считывания телефонных карт, используемых для оплаты разговора.

В начальном состоянии трубка таксофона повешена, дисплей потушен, таксофон не реагирует на нажатия кнопок и какие-либо сигналы из линии. При снятии трубки таксофон выдает на дисплей сообщение «Вставьте карту» и ожидает, когда пользователь вставит карту в приемник. Дальнейшее функционирование таксофона осуществляется только при вставленной карте. Если карту вынимают, таксофон возвращается к началу и выдает сообщение о необходимости вставить карту. При попадании карты в приемник производится считывание информации с карты. Если кредит исчерпан или карта не пригодна (не удастся узнать кредит), то таксофон выдает соответствующее сообщение на дисплей таксофона. Если карта может быть использована для оплаты, то на дисплей выдается количество «единиц» на карте, и на телефонную станцию (АТС) подается сигнал «Трубка». При получении ответного сигнала «Тон» из линии таксофон воспроизводит звуковой тон «Готов» (длинный непрерывающийся гудок) в трубку. При получении сигнала «Занято», в трубке воспроизводится тон «Занято» (короткие гудки).

После получения от АТС сигнала «Тон» от пользователя принимаются семизначный номер вызываемого абонента, остальные нажатия на кнопки игнорируются. Когда пользователь нажимает на кнопку с цифрой соответствующий ей сигнал «Цифра» передается АТС. Во время набора номера введенные цифры отображаются на дисплее. В ответ на набранный номер от АТС приходит либо сигнал «Занято», либо сигнал «Вызов». При получении сигнала «Вызов» таксофон воспроизводит в трубку длинные гудки до того момента, когда АТС осуществит коммутацию и передаст сигнал «Данные». Таксофон воспроизводит данные, передаваемые с сигналом, в трубку. При

получении данных из трубки, аппарат преобразует их в сигнал «Данные» и передает их АТС. Во время разговора на дисплее ведется отсчет времени и уменьшается кредит на телефонной карте - каждые 15 секунд вычитается четверть «единицы». Обмен данными прерывается, в следующих случаях:

- исчерпан кредит;
- карта вынута из приемника;
- от АТС пришел сигнал «Занято»;
- повешена трубка таксофона.

Если трубка была повешена, аппарат посылает в линию сигнал «Конец» и выдает на дисплей сообщение «Выньте карту». После извлечения карты из приемника таксофон переходит в начальное состояние.

## **Вариант 9. Холодильник**

Требуется разработать средствами Rational Rose модель программного обеспечения встроенного процессора холодильника. Холодильник состоит из нескольких холодильных камер для хранения продуктов. В каждой холодильной камере имеется регулятор температуры, мотор, термометр, индикатор, таймер, датчик открытия двери камеры и устройство для подачи звуковых сигналов.

При помощи терморегулятора устанавливается максимально допустимая температура в данной камере. Мотор предназначен для поддержания низкой температуры. Термометр постоянно измеряет температуру внутри камеры, а индикатор температуры, расположенный на дверце, постоянно высвечивает ее значение. При повышении температуры выше предела, определяемого текущим положением регулятора, включается мотор. При снижении температуры ниже некоторого другого значения, связанного с первым, мотор отключается.

Доступ в камеру осуществляется через дверцу. Если дверь холодильной камеры открыта в течение слишком долгого времени, подается звуковой сигнал. Звуковой сигнал также подается в любых нештатных ситуациях (например, при поломке мотора).

Холодильник ведет электронный журнал, в котором отмечаются все происходящие события:

- изменение положения терморегулятора камеры;
- включение и отключение мотора;
- доступ в камеру;
- внештатные ситуации.

Вариантом задания предусмотрена разработка схемы базы данных для хранения журнала событий холодильника. Содержимое журнала может быть передано в компьютер, подсоединенный к специальному гнезду на корпусе холодильника.

### **Вариант 10. Кодовый замок**

Требуется разработать средствами Rational Rose модель программного обеспечения встроеного микропроцессора для кодового замка, регулирующего доступ в помещение.

Кодовый замок состоит из панели с кнопками (цифры «0»...«9», кнопка «Вызов», кнопка «Контроль»), цифрового дисплея, электромеханического замка, звонка. Панель с кнопками устанавливается с наружной стороны двери, замок устанавливается с внутренней стороны двери, звонок устанавливается внутри охраняемого помещения.

В обычном состоянии замок закрыт. Доступ в помещение осуществляется после набора кода доступа, состоящего из четырех цифр. Во время набора кода введенные цифры отображаются на дисплея. Если код набран правильно, то замок открывается на некоторое время, после чего дверь снова закрывается. Содержимое дисплея очищается.

Кнопка «Вызов» используется для подачи звукового сигнала внутри помещения. Кнопка «Контроль» используется для смены кодов. Смена кода доступа осуществляется следующим образом. При открытой двери нужно набрать код контроля, состоящий из четырех цифр, и новый код доступа. Для смены кода контроля нужно при открытой двери и нажатой кнопке «Вызов» набрать код контроля, после чего - новый код контроля.

### **Вариант 11. Турникет метро**

Требуется разработать средствами Rational Rose модель программного обеспечения встроенного процессора турникета для метрополитена.

При помощи турникета контролируется проход пассажиров в метро и взимается входная плата. Турникет имеет приемник карт, устройство для перекрывания доступа, таймер, три оптических датчика для определения прохода пассажира, устройство подачи звуковых сигналов, индикаторы «Проход» и «Стоп».

В начальном состоянии турникета зажжен индикатор «Стоп», индикатор «Проход» потушен. Если один из датчиков посылает сигнал, то проход через турникет сразу же перекрывается, и подается предупредительный звуковой сигнал. Для прохода пассажир должен поместить карту в приемник карт. Турникет считывает с нее данные: срок годности карты и количество «единиц» на ней. Если данные не удастся считать, или карта просрочена, или заблокирована, то карта возвращается пассажиру, и турникет остается в исходном состоянии. В другом случае с карты списывается одна «единица», карта возвращается из приемника, индикатор «Стоп» гаснет, зажигается индикатор «Проход», и пассажир может пройти через турникет. Получив от одного из датчиков сигнал, турникет ожидает время, отведенное на проход пассажира (5 секунд), после чего он возвращается в начальное состояние.

Наличие трех датчиков в турникете гарантирует, что при проходе пассажира хотя бы один из них подаст сигнал (датчики невозможно перешагнуть, перепрыгнуть и т.д.). Во время прохода пассажира возможна ситуация, когда все три датчика посылают сигналы. В этом случае принимается только первый сигнал и от момента его приема отсчитывается положенное время. Остальные сигналы игнорируются.

Турникет заносит в свою память время всех оплаченных проходов. В конце рабочего дня он передает всю информацию, накопленную за день, в АСУ метрополитена.

В ходе выполнения этого варианта задания должна быть разработана схема базы данных о проходах через турникет.

## **Вариант 12. Система учета товаров**

Требуется разработать средствами Rational Rose модель системы поддержки заказа и учета товаров в бакалейной лавке.

В бакалейной лавке для каждого товара фиксируется место хранения (определенная полка), количество товара и его поставщик. Система поддержки заказа и учета товаров должна обеспечивать добавление информации о новом товаре, изменение или удаление информации об имеющемся товаре, хранение (добавление, изменение и удаление) информации о поставщиках, включающей в себя название фирмы, ее адрес и телефон. При помощи системы составляются заказы поставщикам. Каждый заказ может содержать несколько позиций, в каждой позиции указываются наименование товара и его количество в заказе. Система учета по требованию пользователя формирует и выдает на печать следующую справочную информацию:

- список всех товаров;
- список товаров, имеющих в наличии;
- список товаров, количество которых необходимо пополнить;
- список товаров, поставляемых данным поставщиком.

В ходе выполнения этого варианта задания должна быть разработана схема базы данных, хранящей информацию о товарах, заказах и поставщиках.



### **Вариант 13. Библиотечная система**

Требуется разработать средствами Rational Rose модель системы автоматизирующей деятельность библиотеки.

Система поддержки управления библиотекой должна обеспечивать операции (добавление, удаление и изменение) над данными о читателях. В регистрационном списке читателей хранятся следующие сведения: фамилия, имя и отчество читателя; номер его читательского билета и дата выдачи билета. Наряду с регистрационным списком системой должен поддерживаться каталог библиотеки, где хранится информация о книгах: название, список авторов, библиотечный шифр, год и место издания, название издательства, общее количество экземпляров книги в библиотеке и количество экземпляров, доступных в текущий момент. Система обеспечивает добавление, удаление и изменение данных каталога, а также поиск книг в каталоге на основании введенного шифра или названия книги. В системе осуществляется регистрация взятых и возвращенных читателем книг. Про каждую выданную книгу хранится запись о том, кому и когда была выдана книга, и когда она будет возвращена. При возврате книги в записи делается соответствующая пометка, а сама запись не удаляется из системы. Система должна выдавать следующую справочную информацию:

- какие книги были выданы за данный промежуток времени;
- какие книги были возвращены за данный промежуток времени;
- какие книги находятся у данного читателя;
- имеется ли в наличии некоторая книга.

Вариант задания предусматривает разработку схемы базы данных, хранящей список читателей, каталог книг и записи о выдаче книг.

## Вариант 14. Интернет-магазин

Требуется разработать средствами Rational Rose модель программного обеспечения Интернет-магазина.

Интернет-магазин позволяет делать покупки с доставкой на дом. Клиенты магазина при помощи программы-браузера имеют доступ к каталогу продаваемых товаров, поддержку которого осуществляет Интернет-магазин. В каталоге товары распределены по разделам. О каждом товаре доступна полная информация (название, вес, цена, изображение, дата изготовления и срок годности). Для удобства клиентов предусмотрена система поиска товаров в каталоге. Заполнение каталога информацией происходит автоматически в начале рабочего дня, информация берется из системы автоматизации торговли.

При отборе клиентами товаров поддерживается виртуальная «торговая корзина». Любое наименование товара может быть добавлено в «корзину» или изъято в любой момент по желанию покупателя с последующим пересчетом общей стоимости покупки. Текущее содержимое «корзины» постоянно показывается клиенту.

По окончании выбора товаров производится оформление заказа и регистрация покупателя. Клиент указывает в регистрационной форме свою фамилию, имя и отчество, адрес доставки заказа и телефон, по которому с ним можно связаться для подтверждения сделанного заказа. Заказы передаются для обработки в систему автоматизации торговли. Проверка наличия товаров на складе и их резервирование Интернет-магазином не производятся. Дополнительно требуется разработать схему базы данных, хранящей заказы.

При выполнении этого варианта задания рекомендуем ознакомиться с работой [Коналлен-2001]. Следует определиться, по какому архитектурному шаблону будет строиться Web-приложение («тонкий клиент» или «толстый клиент»). В соответствии с выбранным шаблоном следует построить модели клиентской части магазина и серверной части, промоделировать связи между частями приложения. Для Web-приложений типичными являются следующие классы:

- клиентская Web-страница;

- серверная Web-страница (например, CGI-скрипт);
- HTML-форма;
- объект JavaScript.

Дополнительные *связи* между классами Web-приложений:

- link - ссылка с одной страницы на другую;
- build - связь между CGI-скриптом и клиентской страницей, генерируемой при его выполнении;
- submit - связь между формой и серверной Web-страницей, принимающей данные из формы.

Типичные *компоненты*:

- Web-страница (HTML-файл),
- Active Server Page (ASP),
- Java Server Page (JSP),
- сервлет,
- библиотека скриптов (например, подключаемый файл с Javascript-функциями).

## **Вариант 15. WWW-конференция**

Требуется разработать средствами Rational Rose модель программного обеспечения WWW-конференции.

WWW-конференция представляет собой хранилище сообщений в сети Интернет, доступ к которому осуществляется при помощи браузера. Для каждого сообщения конференции хранятся значения следующих полей: номер сообщения, автор, тема, текст сообщения, дата добавления сообщения, ссылка на родительское сообщение. Начальной страницей конференции является иерархический список сообщений. Верхний уровень иерархии составляют сообщения, открывающие новые темы, а подуровни составляют сообщения, полученные в ответ на сообщения верхнего уровня. Сообщение-ответ всегда имеет ссылку на исходное сообщение. В списке отображаются только темы сообщений, их авторы и даты добавления. Просматривая список, пользователь выбирает сообщение и по гиперссылке открывает страницу с текстом сообщения. Помимо текста на этой странице отображается список (иерархический) сообщений являющихся ответами, ответами на ответы и т.д. Для удобства пользователей необходимо предусмотреть поиск сообщений по автору или по ключевым словам в теме или тексте сообщения.

Сообщения добавляются в конференцию зарегистрированными пользователями, которые при отправке сообщения должны указать своё имя и пароль. Регистрирует новых пользователей модератор конференции - её ведущий. При регистрации пользователь заполняет специальную форму, содержимое которой затем пересылается модератору и запоминается в базе пользователей. Модератор решает, регистрировать пользователя или нет, и отправляет свой ответ.

При добавлении сообщений пользователь имеет возможность начать новую тему или ответить на ранее добавленные сообщения. После добавления сообщения оно доступно для чтения всем пользователям (даже незарегистрированным), и список сообщений обновляется.

Модератор имеет право по тем или иным причинам удалять сообщения любых авторов. Он также может наказывать пользователей, нарушающих

правила поведения в конференции, лишая на некоторое время пользователя возможности добавлять и редактировать сообщения.

Вариант задания включает в себя разработку схемы базы данных для хранения сообщений конференции и информации об её участниках.

Выполняющим это задание полезно ознакомиться с заключительным замечанием к варианту «Интернет-магазин». Наиболее подходящей архитектурой для WWW-конференции является «тонкий клиент», поскольку клиентская часть практически не содержит «бизнес-логики». Единственным её элементом, который может выполняться на стороне клиента, является проверка правильного заполнения полей формы, перед отправкой её содержимого на сервер.

## Вариант 16. Банкомат

Требуется разработать средствами Rational Rose модель программного обеспечения банкомата. Банкомат - это автомат для выдачи наличных денег по кредитным пластиковым карточкам. В его состав входят следующие устройства: дисплей, панель управления с кнопками, приемник кредитных карт, хранилище денег и лоток для их выдачи, хранилище конфискованных кредитных карт, принтер для печати справок.

Банкомат подключен к линии связи для обмена данных с банковским компьютером, хранящим сведения о счетах клиентов.

Обслуживание клиента начинается с момента помещения пластиковой карточки в банкомат. После распознавания типа пластиковой карточки, банкомат выдает на дисплей приглашение ввести персональный код. Персональный код представляет собой четырехзначное число. Затем банкомат проверяет правильность введенного кода. Если код указан неверно, пользователю предоставляются еще две попытки для ввода правильного кода. В случае повторных неудач карта перемещается в хранилище карт, и сеанс обслуживания заканчивается. После ввода правильного кода банкомат предлагает пользователю выбрать операцию. Клиент может либо снять наличные со счета, либо узнать остаток на его счету.

При снятии наличных со счета банкомат предлагает указать сумму (10, 50, 100, 200, 500, 1000 рублей). После выбора клиентом суммы банкомат запрашивает, нужно ли печатать справку по операции. Затем банкомат посылает запрос на снятие выбранной суммы центральному компьютеру банка. В случае получения разрешения на операцию, банкомат проверяет, имеется ли требуемая сумма в его хранилище денег. Если он может выдать деньги, то на дисплей выводится сообщение «Выньте карту». После удаления карточки из приемника, банкомат выдает указанную сумму в лоток выдачи. Банкомат печатает справку по произведенной операции, если она была затребована клиентом.

Если клиент хочет узнать остаток на счету, то банкомат посылает запрос центральному компьютеру банка и выводит сумму на дисплей. По требованию клиента печатается и выдается соответствующая справка.

В специальном отделении банкомата, закрываемом замком, есть «секретная кнопка», которая используется обслуживающим персоналом для загрузки денег. При нажатии на эту кнопку открывается доступ к хранилищу денег и конфискованным кредитным картам.

### **Вариант 17. Каталог ресурсов Интернет**

Требуется разработать средствами Rational Rose модель программного обеспечения каталога ресурсов сети Интернет.

В каталоге хранится следующая информация о ресурсах: название ресурса, уникальный локатор ресурса (URL), раздел каталога, в котором содержится ресурс, список ключевых слов, краткое описание, дата последнего обновления, контактная информация.

Доступ пользователей к каталогу осуществляется при помощи браузера. Пользователи каталога могут добавлять новые ресурсы, информация о которых не была внесена ранее. Ресурсы в каталоге классифицируются по разделам. Полный список ресурсов каждого раздела должен быть доступен пользователям. Пользователям каталога должны быть предоставлены возможности по поиску ресурсов. Поиск осуществляется по ключевым словам. Если пользователь не доволен результатами поиска, он может уточнить запрос (осуществить поиск среди результатов предыдущего поиска). Должна быть возможность выдавать результаты поиска в разной форме (вывод всей информации о ресурсах или частичной). Пользователь может отсортировать список ресурсов по релевантности (соответствию ключевым словам из запроса) или по дате обновления.

Поскольку содержание ресурсов Интернет со временем изменяется необходимо следить за датой последнего обновления, периодически опрашивая Web-сайты, URL которых хранятся в каталоге.

Вариант задания включает в себя разработку схемы базы данных для хранения сообщений конференции и информации об её участниках.

Выполняющим это задание полезно ознакомиться с заключительным замечанием к варианту «Интернет-магазин». Как и в варианте «WWW-конференция» самой подходящей архитектурой для каталога является «тонкий клиент», поскольку клиентская часть практически не включает в себя функций «бизнес-логики» кроме проверки содержимого форм перед пересылкой на сервер.



### **Вариант 18. Будильник**

Требуется разработать средствами Rational Rose модель программного обеспечения встроеного микропроцессора для будильника.

На экране будильника постоянно отображается текущее время (часы и минуты, например: 12 : 00), двоеточие между числом часов и числом минут загорается и гаснет с интервалом в полсекунды.

Управление будильником осуществляется следующими кнопками:

- кнопкой режима установки времени,
- кнопкой режима установки времени срабатывания,
- двумя отдельными кнопками для установки часов и минут,
- кнопкой сброса сигнала «СБРОС».

На будильнике имеется переключатель режима работы со следующими положениями: «ВЫКЛ», «ВКЛ», «РАДИО» и «ТАЙМЕР».

Для установки текущего времени нужно нажать на кнопку режима установки и, при нажатой кнопке, нажимать на кнопки установки часов и минут. При каждом нажатии на кнопки, устанавливаемое значение увеличивается на одну единицу (один час или одну минуту соответственно). При достижении максимального значения производится сброс. Для установки времени срабатывания будильника нужно нажать на кнопку режима установки времени срабатывания и, держа кнопку нажатой, нажимать на кнопки установки часов и минут. Когда переключатель режима работы находится в положении «ВКЛ», при достижении времени срабатывания происходит подача звукового сигнала в течение одной минуты. Сигнал можно прервать, нажав на кнопку «СБРОС». При этом сигнал должен быть возобновлен через пять минут. При установке переключателя в положение «ВЫКЛ» звуковой сигнал не подается.

Когда переключатель находится в положении «РАДИО» работает радиоприемник. При переводе переключателя в положение «ТАЙМЕР» включается радиоприемник на тридцать минут, а затем часы переходят в состояние будильника (аналогично положению «ВКЛ»). При нажатии на кнопку режима установки времени, будильник должен отображать время срабатывания.

### **Вариант 19. Генеалогическое дерево**

Требуется разработать средствами Rational Rose модель системы для поддержки генеалогических деревьев.

Система хранит сведения о персонах (Ф.И.О., пол, дата рождения, дата смерти, биография) и о родственных связях между ними. Связи бывают только трех видов: «мужья-жены», «дети-родители» и «братья-сестры». Система обеспечивает возможность добавления данных о новых персонах и родственных связях, изменение введенных данных и удаление ненужных данных. Система следит за непротиворечивостью вводимых данных. Например, недопустимо, чтобы человек был собственным предком или потомком.

Разработанная модель должна содержать схему базы данных для хранения генеалогических деревьев.

Пользователи системы могут осуществлять поиск полезной информации по дереву:

- находить для указанного члена семьи его детей;
- находить для указанного члена семьи его родителей;
- находить для указанной персоны братьев и сестер, если таковые есть;
- получать список всех предков персоны;
- получать список всех потомков персоны;
- получать список всех родственников персоны;
- прослеживать цепочку родственных связей от одной персоны до другой (например, если Петр является шурином Ивана, то на запрос о родственных связях между Петром и Иваном выдается такой результат: «Петр - брат Ольги, Ольга - жена Ивана»).

## Вариант 20. Телевизор

Требуется разработать средствами Rational Rose модель встроенной системы управления работой телевизора.

В телевизоре имеются следующие устройства: приемник телевизионного сигнала, устройство отображения картинки, память каналов, память настроек, управляющие кнопки, пульт дистанционного управления (ДУ). Управление телевизором осуществляется при помощи кнопок на корпусе (их четыре: «ВКЛ/ВЫКЛ», «-», «+», кнопка начальной установки) и пульта ДУ. Кнопка «ВКЛ/ВЫКЛ» позволяет включать и выключать телевизор. После включения телевизора на экран отображается передача, идущая по каналу №1, при этом используются параметры изображения и значение громкости, сохраненные в памяти настроек.

Память каналов телевизора хранит до 60 каналов. Каналы нумеруются, начиная с нуля. Последовательное переключение каналов осуществляется при помощи кнопок «-» и «+». Нажатие на «+» переключает телевизор на канал с номером, на единицу большим (с 59-го канала телевизор переключается на 0-ой). Нажатие на «-» переключает телевизор на канал с номером, на единицу меньшим (с 0-го канала телевизор переключается на 59-ый).

При нажатии на кнопку начальной установки очищается память каналов телевизора, после чего осуществляется поиск передач и сохранение их частот в памяти каналов. Поиск начинается с нижней границы рабочего диапазона телевизора. На экран телевизора выводится «синий экран». Рабочая частота постепенно увеличивается до тех пор, пока приемник не обнаружит телевизионный сигнал. Найденная передача выводится на экран в течение 10 секунд. Также отображается номер, под которым найденный канал будет сохранен в памяти (начиная с 1).