МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

СЕВЕРО-КАВКАЗСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ

А. Х. Башиева

ПРОЕКТЫЙ ПРАКТИКУМ

Учебно-методическое пособие для обучающихся по направлению подготовки 09.03.03 Прикладная информатика

Черкесск, 2024

УДК 004.9
ББК 16
Б 33

Рассмотрено на заседании кафедры Прикладной информатики
Протокол № 1 от 31.08.2023 г.
Рекомендовано к изданию редакционно-издательским советом СКГА
Протокол № 26 от 29.09.2024 г.

Рецензенты: Бостанова Л. К. – к.п.н., доцент кафедры ПИ

Б33 **Башиева, А. Х.** Проектный практикум: учебно-методическое пособие по работе для студентов обучающихся по направлению подготовки 09.03.03. Прикладная информатика / А. Х. Башиева. – Черкесск: БИЦ СКГА, 2024. – 100с.

Учебно-методическое пособие составлено на основе примерной рабочей программы по курсу Проектный практикум в соответствии с требованиями государственного общеобразовательного стандарта.

УДК 004.9
ББК 16

СОДЕРЖАНИЕ

Общая характеристика CASE-средства IBM Rational Rose 2003

Среди всех фирм-производителей CASE-средств именно компания IBM Rational Software Corp. (до августа 2003 года – Rational Software Corp.) одна из первых осознала стратегическую перспективность развития объектно-ориентированных технологий анализа и проектирования программных систем. Эта компания выступила инициатором унификации языка визуального моделирования в рамках консорциума OMG, что, в конечном итоге, привело к появлению первых версий языка UML. И эта же компания первой разработала инструментальное объектно-ориентированное CASE-средство, в котором был реализован язык UML как базовая нотация визуального моделирования.

CASE-средство IBM Rational Rose со времени своего появления претерпело серьезную эволюцию, и в настоящее время представляет собой современный интегрированный инструментарий для проектирования архитектуры, анализа, моделирования и разработки программных систем. Именно в IBM Rational Rose язык UML стал базовой технологией визуализации и разработки программных систем, что определило популярность и стратегическую перспективность этого инструментария.

В рамках общего продукта IBM Rational Rose существуют различные варианты этого средства, отличающиеся между собой диапазоном предоставляемых возможностей. Базовым средством в настоящее время является IBM Rational Rose Enterprise Edition, которое обладает наиболее полными возможностями. Возможности версии IBM Rational Rose 2003 (release 2003.06.00) аккумулируют практически все современные достижения в области информационных технологий. Наиболее характерные функциональные особенности этой программы заключаются в следующем:

- интеграция с MS Visual Studio 6, которая включает поддержку на уровне прямой и обратной генерации кодов и диаграмм Visual Basic и Visual C++ с использованием ATL (Microsoft Active Template Library), Web-Классов, DHTML и протоколов доступа к различным базам данных;
- непосредственная работа (инжиниринг и реинжиниринг) с исполняемыми модулями и библиотеками форматов EXE, DLL, TLB, OCX.
- поддержка технологий MTS (Microsoft Transaction Server) и ADO (ActiveX Data Objects) на уровне шаблонов и исходного кода, а также элементов технологии Microsoft – COM+ (DCOM);
- полная поддержка компонентов CORBA и J2EE, включая реализацию технологии компонентной разработки приложений CBD (Component-Based Development), языка определения интерфейса IDL (Interface Definition Language) и языка определения данных DDL (Data Definition Language);
- полная поддержка среды разработки Java-приложений, включая прямую и обратную генерацию классов Java формата JAR, а также работу с файлами формата CAB и ZIP.

Особенности рабочего интерфейса программы IBM Rational Rose

В CASE-средстве IBM Rational Rose 2003 реализованы общепринятые стандарты на *рабочий интерфейс программы*, аналогично известным средам визуального программирования. После установки IBM Rational Rose 2003 на компьютер пользователя, что практически не вызывает трудностей у разработчиков, запуск этого средства в среде MS Windows 2000/XP приводит к появлению на экране соответствующего *рабочего интерфейса* (рис. 1).

Рабочий интерфейс программы IBM Rational Rose состоит из множества основных элементов:

- *главное меню*;
- *стандартная панель инструментов*;
- специальная панель инструментов;
- окно браузера проекта;
- рабочая область изображения диаграммы или окно диаграммы;
- окно документации;
- окно журнала.

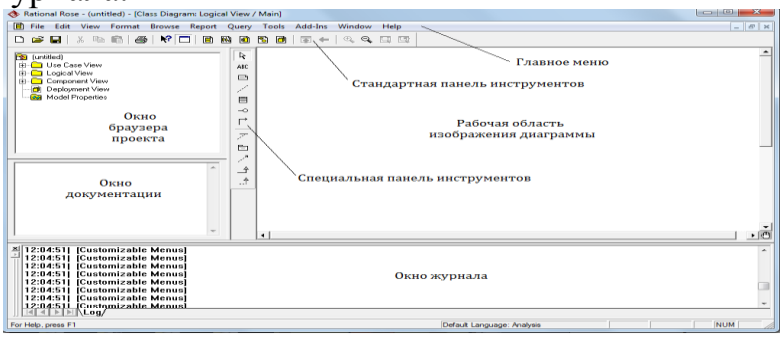


Рисунок 1 – Общий вид рабочего интерфейса CASE-средства IBM Rational Rose

Главное меню и стандартная панель инструментов

Стандартная панель инструментов.

Некоторые из инструментов недоступны для нового проекта, который не имеет никаких элементов.

Пользователь может настроить внешний вид этой панели по своему усмотрению. Для этого необходимо выполнить: **Tools→Options** (Инструменты→Параметры), открыть вкладку **Toolbars** (Панели инструментов)→**Standard** *(Стандартная)*.

Назначение операций главного меню File и Edit

Главное меню, позволяет пользователю загружать и сохранять информацию во внешних файлах, изменять внешний вид элементов графического интерфейса, вызывать справочную информацию, вызывать другие диалоговые окна для работы с программой IBM Rational Rose 2003 и т.д.

Таблица 1 – Операции пункта главного меню File (Файл)

Название операции меню	Наличие кнопки на стандартной панели	Назначение операции главного меню
New		Создает новую модель IBM Rational Rose 2003. При этом новая модель по умолчанию имеет имя untitled
Open		Вызывает стандартное диалоговое окно открытия внешнего файла с диска. Открыть можно либо файл модели (файл с расширением «mdl»), либо файл подмодели (файл с расширением «ptl»)
Save		Позволяет сохранить разрабатываемую модель во внешнем файле на диске
Save As		Позволяет сохранить разрабатываемую модель под другим именем во внешнем файле на диске. При этом вызывается стандартное диалоговое окно сохранения файла на диске с предложением задать имя соответствующего файла модели или подмодели.
Save Log As		Позволяет сохранить содержание журнала во внешнем файле на диске с именем error.log. При этом вызывается стандартное диалоговое окно сохранения файла на диске с предложением изменить предлагаемое по умолчанию имя соответствующего файла.
AutoSave Log		Позволяет автоматически сохранять содержание журнала во внешнем файле на диске с именем error.log. При первом выполнении этого пункта меню также вызывается стандартное диалоговое окно сохранения файла на диске с предложением изменить предлагаемое по умолчанию имя соответствующего файла.
Clear Log		Очищает содержание журнала
Load Model Workspace		Позволяет загрузить рабочую область из внешнего файла на диске. Вызывает стандартное диалоговое окно открытия внешнего файла с диска, при этом открыть можно файл с расширением «wsp»
Save Model Workspace		Позволяет сохранить рабочую область модели во внешнем файле на диске. При выполнении этого пункта меню вызывается стандартное диалоговое окно сохранения файла с расширением «wsp»
Save Model Workspace As		Позволяет сохранить рабочую область модели во внешнем файле на диске. Вызывается стандартное диалоговое окно сохранения файла с предложением изменить предлагаемое по умолчанию имя соответствующего файла
Units		Позволяет загрузить категорию элементов модели из внешнего файла на диске. Вызывает стандартное диалоговое окно открытия внешнего файла с диска, при этом открыть можно файл с расширением «cat»
Import		Позволяет импортировать информацию из внешних файлов различных форматов, включая файлы моделей, подмоделей, категорий и подсистем
Export Model		Позволяет экспортировать информацию о модели во внешний файл. Вид этого пункта меню зависит от выделенного элемента

		модели
Update		Позволяет вставить информацию обратного проектирования из внешнего файла с расширением «red» в разрабатываемую модель
Print		Позволяет распечатать на принтере отдельные диаграммы и спецификации различных элементов разрабатываемой модели. В этом случае вызывается диалоговое окно выбора диаграмм и спецификаций для печати на подключенном к данному компьютеру принтеру
Print Setup		Вызывается стандартное диалоговое окно макета страницы для настройки свойств печати
Edit Path Map		Вызывает окно задания путей доступа к файлам системы IBM Rational Rose 2003. Как правило, значения путей, установленные по умолчанию, следует изменять только в случае крайней необходимости
<Имена файлов>		Секция с именами последних файлов, с которыми осуществлялась работа в IBM Rational Rose 2003
Exit		Прекращает работу и закрывает IBM Rational Rose 2003

Таблица 2 – Операции пункта главного меню Edit (Редактирование)

Название операции меню	**Наличие кнопки на стандартной панели**	**Назначение операции главного меню**
Undo		Отменяет выполнение последнего действия по удалению или перемещению элементов модели
Redo		Восстанавливает изображение диаграммы после отмены выполнения последней операции перемещения
Cut		Вырезает выделенный элемент разрабатываемой модели и помещает его в буфер обмена
Copy		Копирует выделенный элемент разрабатываемой модели и помещает его в буфер обмена
Paste		Вставляет элемент разрабатываемой модели или его копию из буфера обмена в текущую активную диаграмму
Delete		Удаляет выделенные элементы из текущей диаграммы, но не из разрабатываемой модели
Select All		Выделяет все элементы на текущей диаграмме разрабатываемой модели
Delete from Model		Удаляет все выделенные элементы из разрабатываемой модели
Relocate		Позволяет перемещать или отменять перемещение классов, ассоциаций или компонентов из одного пакета в другой
Find		Вызывает диалоговое меню поиска элемента в разрабатываемой модели по его имени
Reassign		Позволяет заменить выделенный элемент разрабатываемой модели другим элементом модели
Compartment		Позволяет отображать дополнительную информацию об объектах, классах, актерах или пакетах
Change Info		Позволяет изменить тип выделенного элемента на текущей диаграмме на другой тип элемента

Операции *главного меню* **View** (Вид) позволяют отображать на экране различные элементы *рабочего интерфейса* и изменять графическое представление диаграмм.

Таблица 3 – Операции пункта главного меню View (Вид)

Название операции меню	**Наличие кнопки на *стандартной панели***	**Назначение операции *главного меню***
Toolbars		Позволяет настроить внешний вид рабочего интерфейса системы IBM Rational Rose 2003 и содержит дополнительные подпункты: **Standard** – делает видимой/невидимой *стандартную панель инструментов* (рис. 1.3) **Toolbox** – делает видимой/невидимой *стандартную панель инструментов* текущей активной диаграммы **Configure** – вызывает диалоговое окно настройки параметров модели, открытое на вкладке настройки панелей инструментов
Status Bar		Делает видимой/невидимой строку состояния
Documentation		Делает видимым/невидимым окно документации
Browser		Делает видимым/невидимым браузер проекта
Log		Делает видимым/невидимым окно журнала
Editor		Делает видимым/невидимым встроенный текстовый редактор
Time Stamp		Включает/выключает режим отображения времени в записях журнала
Zoom to Selection		Изменяет масштаб изображения выделенных элементов модели , так чтобы они разместились в одном окне
Zoom In		Увеличивает масштаб изображения
Zoom Out		Уменьшает масштаб изображения
Fit in Window		Изменяет (уменьшает) масштаб изображения всех элементов текущей диаграммы, так чтобы все они разместились в одном окне
Undo Fit in Window		Отменяет изменение масштаба изображения размещения элементов в одном окне
Page Breaks		Разбивает текущую диаграмму на страницы для последующей печати
Refresh		Перерисовывает текущую диаграмму
As Booch		Изображает элементы модели в соответствии с нотацией Г. Буча
As OMT		Изображает элементы модели в соответствии с нотацией OMT
As Unified		Изображает элементы модели в соответствии с нотацией языка UML

Операции *главного меню* **Format** (Формат) позволяют выполнять действия по изменению внешнего вида элементов модели на различных диаграммах.

Таблица 4 – Операции пункта главного меню Format (Формат)

Название операции меню	**Назначение операции *главного меню***
Font Size	Изменяет масштаб используемого шрифта
Font	Вызывает диалоговое окно выбора шрифта
Line Color	Вызывает диалоговое окно выбора цвета линий
Fill Color	Вызывает диалоговое окно выбора цвета для изображения графических элементов диаграмм
Use Fill Color	Включает/выключает режим отображения цвета для изображения графических элементов диаграмм
Automatic Resize	Включает/выключает режим автоматического изменения размеров графических элементов диаграмм для отображения текстовой информации об их свойствах
Stereotype	Позволяет выбрать способ изображения стереотипов выделенных элементов диаграммы и содержит дополнительные подпункты: **None** – стереотип не показывается; **Label** – стереотип отображается в форме текста; **Decoration** – стереотип отображается в форме небольшой пиктограммы в правом верхнем углу графического элемента; **Icon** – элемент диаграммы отображается в форме специального графического стереотипа, если данный стереотип предусмотрен в программе.
Stereotype Label	Включает/выключает режим отображения текстовых стереотипов для взаимосвязей (ассоциаций, зависимостей и пр.) диаграммы
Show Visibility	Включает/выключает режим отображения кванторов видимости атрибутов и операций выделенных классов
Show Compartment Stereotypes	Включает/выключает режим отображения текстовых стереотипов атрибутов и операций выделенных классов
Show Operation Signature	Включает/выключает режим отображения сигнатуры операций выделенных классов
Show All Attributes	Делает видимыми/невидимыми атрибуты выделенных классов
Show All Operations	Делает видимыми/невидимыми операции выделенных классов
Suppress Attributes	Делает видимой/невидимой секцию атрибутов выделенных классов. Скрывает секцию атрибутов даже в том случае, когда выбрана опция **Show All Attributes**
Suppress Operations	Делает видимой/невидимой секцию операций выделенных классов. Скрывает секцию операций даже в том случае, когда выбрана опция **Show All Operations**
Line Style	Позволяет выбрать способ графического изображения линий взаимосвязей и содержит дополнительные подпункты: **Rectilinear** – линия изображается в форме вертикальных и горизонтальных отрезков; **Oblique** – линия изображается в форме наклонных отрезков; **Toggle** – промежуточный вариант изображения линии

Layout Diagram	Позволяет автоматически разместить графические элементы в окне диаграммы с минимальным количеством пересечений и наложений соединительных линий
Autosize All	Позволяет автоматически изменить размеры графических элементов текущей диаграммы таким образом, чтобы текстовая информация помещалась внутри изображений соответствующих элементов
Layout Selected Shapes	Позволяет автоматически разместить выделенные графические элементы в окне диаграммы с минимальным количеством пересечений и наложений соединительных линий

Операции *главного меню* **Browse** (Обзор) позволяют отображать рабочие окна с различными каноническими диаграммами разрабатываемой модели и вызывать диалоговые окна редактирования свойств отдельных элементов модели.

Таблица 5 – Операции пункта главного меню Browse (Обзор)

Название операции меню	**Наличие кнопки на *стандартной панели***	**Назначение операции *главного меню***
Use Case Diagram		Вызывает диалоговое окно с предложением выбрать для отображения в рабочем окне одну из существующих диаграмм вариантов использования модели или приступить к разработке новой диаграммы
Class Diagram		Вызывает диалоговое окно с предложением выбрать для отображения в рабочем окне одну из существующих диаграмм классов модели или приступить к разработке новой диаграммы
Component Diagram		Вызывает диалоговое окно с предложением выбрать для отображения в рабочем окне одну из существующих диаграмм компонентов модели или приступить к разработке новой диаграммы
Deployment Diagram		Позволяет отобразить в рабочем окне диаграмму развертывания разрабатываемой модели
Interaction Diagram		Вызывает диалоговое окно с предложением выбрать для отображения в рабочем окне одну из существующих диаграмм кооперации или последовательности, а также приступить к разработке новой диаграммы взаимодействия
State Machine Diagram		Вызывает диалоговое окно с предложением выбрать для отображения в рабочем окне одну из существующих диаграмм состояний модели или приступить к разработке новой диаграммы
Expand		Отображает в рабочем окне первую из диаграмм выделенного пакета модели
Parent		Отображает в рабочем окне родителя выделенной диаграммы модели
Specification		Вызывает диалоговое окно свойств выделенного элемента

		модели
Top Level		Отображает в рабочем окне диаграмму самого верхнего уровня для текущей диаграммы модели
Referenced Item		Отображает в рабочем окне диаграмму классов, содержащую класс для выделенного объекта модели
Previous Diagram		Отображает в рабочем окне предыдущую диаграмму модели
Create Message Trace Diagram		Позволяет создать диаграмму трассировки сообщений

Окно браузера проекта

Браузер проекта организует представления модели в виде иерархической структуры, которая упрощает навигацию и позволяет отыскать любой элемент модели в проекте. При этом самая верхняя строка браузера проекта содержит имя разрабатываемого проекта. Любой элемент, который разработчик добавляет в модель, сразу отображается в *окне браузера*. Соответственно, выбрав элемент в *окне браузера*, мы можем его визуализировать в *окне диаграммы* или изменить его спецификацию.

Иерархическое представление структуры каждого разрабатываемого проекта организовано в форме 4–х представлений:

- **Use Case View** – представление вариантов использования, в котором содержатся диаграммы вариантов использования и их реализации в виде вариантов взаимодействия;
- **Logical View** – логическое представление, в котором содержатся диаграммы классов, диаграммы состояний и диаграммы деятельности;
- **Component View** – представление компонентов, в котором содержатся диаграммы компонентов разрабатываемой модели;
- **Deployment View** – представление развертывания, в котором содержится единственная диаграмма развертывания разрабатываемой модели.

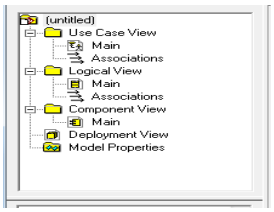


Рисунок 2 – Внешний вид браузера проекта с иерархическим представление его структуры

При создании нового проекта указанная иерархическая структура формируется программой автоматически.

Специальная панель инструментов и окно диаграммы

Внешний вид *специальной панели инструментов* зависит не только от выбора типа разрабатываемой диаграммы, но от выбора графической

нотации для изображения самих элементов этих диаграмм. В IBM Rational Rose 2003 реализованы три таких нотации: UML, OMT и Booch. При использовании отдельной нотации одна и та же диаграмма может быть представлена различным образом, для этого достаточно выбрать желаемое представление через соответствующую операцию главного меню **View** (Вид). При этом никаких дополнительных действий выполнять не требуется – диаграмма преобразуется в выбранную нотацию автоматически.

Окно диаграммы является основной графической областью программы IBM Rational Rose 2003, в которой визуализируются различные представления модели проекта. При разработке нового проекта, если не был использован мастер проектов, ***окно диаграммы*** представляет собой чистую область, не содержащую никаких элементов модели. По мере разработки отдельных диаграмм в *окне диаграммы* будут располагаться соответствующие графические элементы модели.

При активизации отдельного вида диаграммы изменяется внешний вид *специальной панели инструментов*, которая настраивается под конкретный вид диаграммы.

Окно документации и окно журнала

Окно документации по умолчанию должно присутствовать на экране после загрузки программы. Если по какой–то причине оно отсутствует, то его можно отобразить через пункт меню **View→Documentation** (Вид→ Документация), после чего *окно документации* появится ниже *окна браузера проекта* (рис. 3). ***Окно документации***, как следует из его названия, предназначено для документирования элементов разрабатываемой модели. В него можно записывать различную текстовую информацию, и что важно – на русском языке. Эта информация при генерации программного кода преобразуется в комментарии и никак не влияет на логику выполнения программного кода.

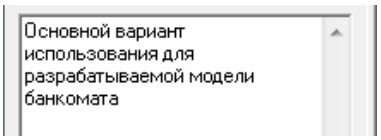


Рисунок 3 – Внешний вид окна документации с информацией о варианте использования

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

Окно журнала (**Log**) предназначено для автоматической записи различной служебной информации 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* изображается поверх других окон в нижней области рабочего интерфейса программы (рис. 4).

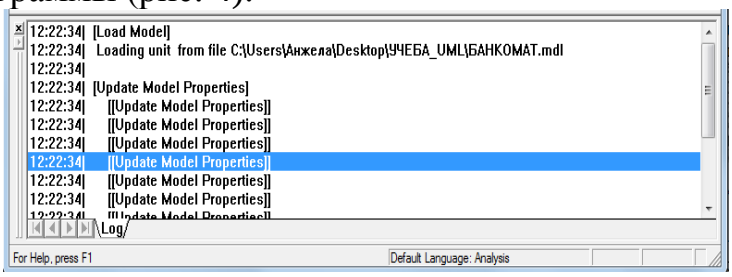


Рисунок 4 – Внешний вид окна журнала с информацией о выполненных операциях с моделью

Если *окно журнала* отсутствует на экране, то отобразить его можно с помощью операции главного меню **View→Log** (Вид→Журнал), для чего следует выставить отметку в соответствующей строке вложенного меню для данной операции.

Назначение операций главного меню **Report, Query и Tools**

Операции главного меню **Report** (Отчет) позволяют отображать различную информацию об элементах разрабатываемой модели и вызывать диалоговое окно выбора шаблона для генерации отчета о модели.

Таблица 6 – Операции пункта главного меню Report (Отчет)

Название операции меню	**Назначение операции главного меню**
Show Usage	Отображает в диалоговом окне информацию об использовании выделенного элемента модели на различных диаграммах
Show Instances	Отображает в диалоговом окне информацию об использовании объектов выделенного класса модели на различных диаграммах
Show Access Violations	Отображает в диалоговом окне информацию о ссылках классов одного пакета на классы другого пакета при отсутствии соответствующей зависимости доступа или импорта между этими пакетами в модели
SoDA Report	Позволяет сгенерировать отчет о разрабатываемой модели в формате MS Word с использованием специального средства IBM Rational SoDA
Show Participants in UC	Отображает в диалоговом окне информацию о классах, компонентах и операциях, которые участвуют в реализации выделенного варианта использования модели на различных диаграммах

Таблица 7 – Операции пункта главного меню Query (Запрос)

Название операции меню	**Назначение операции главного меню**
Add Classes	Вызывает диалоговое окно с предложением добавить на текущую диаграмму классы, которые имеются в модели на различных диаграммах
Add Use Cases	Вызывает диалоговое окно с предложением добавить на текущую диаграмму варианты использования, которые имеются в модели на различных уровнях
Expand Selected Elements	Вызывает диалоговое окно с предложением добавить на текущую диаграмму элементы модели, которые связаны с выделенным элементом на других диаграммах
Hide Selected Elements	Вызывает диалоговое окно с предложением удалить с текущей диаграммы элементы модели, которые связаны с выделенным элементом
Filter Relationships	Вызывает диалоговое окно, позволяющее включить/выключить режим отображения различных отношений на текущей диаграмме

Состав операций пункта главного меню **Tools** (Инструменты) зависит от установленных в программе IBM Rational Rose 2003 конкретных расширений.

Таблица 8 – Операции пункта главного меню Tools (Инструменты

Название операции меню	**Назначение операции главного меню**
Create	Создает новый элемент модели из предлагаемого списка, для последующего размещения его на текущей или другой диаграмме, дублируя нажатие соответствующей кнопки на *специальной панели инструментов*
Check Model	Проверяет разрабатываемую модель на наличие ошибок, информация о которых отображается в *окне журнала*
Model Properties	Позволяет выполнить настройку свойств языка реализации для выделенного элемента модели и содержит дополнительные подпункты: **Edit** – редактирование набора свойств; **View** – просмотр набора свойств; **Replace** – замена существующего набора свойств на новый набор свойств, загружаемый из внешнего файла с расширением "prp" или "pty"; **Export** – сохранение существующего набора свойств во внешнем файле с расширением "prp" или "pty"; **Add** – добавление к существующему набору свойств нового набора свойств, загружаемого из внешнего файла с расширением "prp" или "pty"; **Update** – обновление существующего набора свойств после его редактирования или дополнения
Options	Вызывает диалоговое окно настройки параметров модели, открытое на вкладке **General**
Open Script	Вызывает стандартное диалоговое окно для открытия внешнего файла, содержащего текст скрипта (файл с расширением "ebs") для его

	редактирования в окне встроенного редактора скриптов
New Script	Открывает дополнительное окно встроенного редактора скриптов для создания, отладки, выполнения и сохранения нового скрипта во внешнем файле с расширением "ebs"
ANSI C++	Позволяет выполнить настройку свойств языка программирования ANSI C++, выбранного в качестве языка реализации отдельных элементов модели
CORBA	Позволяет выполнить настройку свойств и спецификацию модели для генерации объектов CORBA для реализации отдельных элементов модели
Java/J2EE	Позволяет выполнить настройку свойств языка программирования Java/J2EE, выбранного в качестве языка реализации отдельных элементов модели
Oracle8	Позволяет выполнить настройку свойств и спецификацию модели для генерации схем СУБД Oracle8 для отдельных элементов модели
Quality Architect	Позволяет выполнить настройку свойств и тестирование модели с помощью специального средства IBM Rational Quality Architect
Rational Requisite Pro	Позволяет выполнить настройку свойств модели для установления связей со специальным средством спецификации и управления требованиями
Model Integrator	Открывает окно специального средства интеграции моделей
Web Publisher	Позволяет выполнить настройку свойств модели для ее публикации в гипертекстовом формате
TOPLink	Вызывает мастер преобразования таблиц модели данных в классы языка программирования Java, выбранного в качестве языка реализации отдельных элементов модели
COM	Позволяет выполнить настройку свойств и спецификацию модели для генерации объектов COM с целью реализации отдельных элементов модели
Visual C++	Позволяет выполнить настройку свойств и спецификацию модели для генерации программного кода MS Visual C++, выбранного в качестве языка реализации отдельных элементов модели
Version Control	Позволяет выполнить настройку свойств модели для установления со специальным средством управления и контроля версий модели
Visual Basic	Позволяет выполнить настройку свойств и спецификацию модели для генерации программного кода MS Visual Basic, выбранного в качестве языка реализации отдельных элементов модели
XML_DTD	Позволяет выполнить настройку свойств и спецификацию модели для ее публикации в формате расширяемого языка разметки XML
Class Wizard	Вызывает мастер создания нового класса и его размещения на выбранной диаграмме модели

Назначение операций главного меню Add-Ins, Window и Help

Пункт главного меню **Add-Ins** (Расширения) вызывает специальное диалоговое окно менеджера расширений для добавления их в операции пункта меню **Tools** (рис. 5). При этом доступными являются те расширения, которые были установлены при инсталляции программы IBM Rational Rose 2003. Поскольку перечень расширений и конкретный вид диалогового окна зависит от вида лицензии и конфигурации поставки IBM Rational Rose 2003.

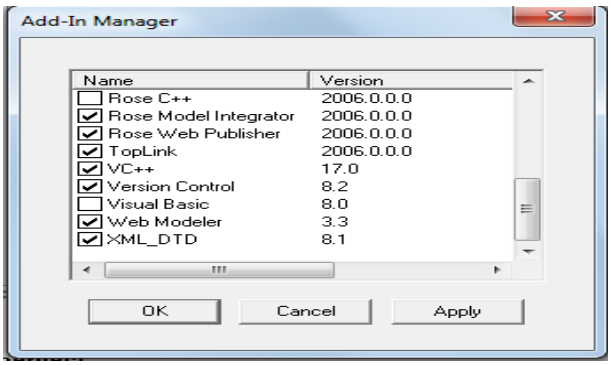


Рисунок 5 – Внешний вид диалогового окна менеджера расширений

Операции главного меню **Window** (Окно) позволяют активизировать *окно нужной диаграммы* разрабатываемой модели из числа открытых и отображать все открытые *окна диаграмм* в различном графическом представлении.

Таблица 9 – Операции пункта главного меню Window (Окно)

Название операции меню	**Назначение операции главного меню**
Cascade	Размещает *окна всех открытых диаграмм* модели каскадно
Tile	Отображает в *окне диаграмм* все открытые диаграммы модели
Arrange Icons	Упорядочивает расположение всех открытых диаграмм
	Секция, содержащая имена всех открытых диаграмм модели для переключения между ними. Если открывается новая диаграмма, то в этой секции появляется новая строка с именем этой диаграммы и ее типом, выбрав которую, можно сразу перейти в нужное окно

Операции главного меню **Help** (Справка) позволяют получить справочную информацию о программе IBM Rational Rose 2003 и об особенностях разработки графических моделей в соответствующих нотациях.

Таблица 10 – Операции пункта главного меню Help (Справка)

Название операции меню	**Назначение операции главного меню**
Contents and Index	Вызывает программу просмотра справочной системы, открытой на вкладке Содержание
Search for Help on	Вызывает программу просмотра справочной системы, открытой на вкладке Указатель
Using Help	Вызывает программу отображения информации об использовании справочной системы
Extended Help	Вызывает специальную программу расширенной справочной системы
Contacting Technical Support	Вызывает установленный в операционной системе по умолчанию браузер Интернет и делает попытку соединиться с web–сайтом технической поддержки компании IBM Rational при наличии доступа в

	Интернет
Rational on the Web	Вызывает установленный в операционной системе по умолчанию браузер Интернет и делает попытку соединиться с web-сайтом компании IBM Rational при наличии доступа в Интернет. Выбор отдельной операции этого пункта меню определяет загрузку той или иной web-страницы компании, предназначенной для выполнения специальных действий по дополнительной поддержке средства IBM Rational Rose или загрузке имеющихся обновлений
Rational Developer Network	Вызывает установленный в операционной системе по умолчанию браузер Интернет и делает попытку соединиться с web-сайтом разработчиков компании IBM Rational при наличии доступа в Интернет
About Rational Rose	Отображает информацию о текущей рабочей версии IBM Rational Rose

Задание 1. Разработка диаграммы вариантов использования и редактирование свойств ее элементов

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

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

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

Как и другие программы, IBM Rational Rose позволяет настраивать глобальные параметры среды, такие как выбор шрифтов и цвета для представления различных элементов модели. Настройка шрифтов, цвета

линий и графических элементов производится через операцию главного меню: **Tools→Options** (Инструменты→Параметры). Характерной особенностью среды является возможность работы с символами кириллицы. Однако следует заметить, что при спецификации элементов модели с последующей генерацией текста программного кода следует записывать имена и свойства классов, *ассоциаций*, атрибутов, операций и компонентов символами того языка, который поддерживается соответствующим языком программирования.

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

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

При этом появляется новое окно с чистым рабочим листом диаграммы вариантов использования и специальная панель инструментов, содержащая кнопки с изображением графических элементов, необходимых для разработки диаграммы вариантов использования.

Таблица 11 – Назначение кнопок специальной панели инструментов для диаграммы вариантов использования

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

Добавить кнопки с пиктограммами других графических элементов, например, таких как *бизнес-вариант использования* (business use case), *бизнес-актер* (business actor), сотрудник (business worker), или удалить

ненужные кнопки можно с помощью настройки специальной панели инструментов: **Tools→Options** (Инструменты→Параметры), раскрыв вкладку **Toolbars** (Панели инструментов) и нажав соответствующую кнопку (например, **Use Case diagram**) в группе опций **Customize Toolbars** (Настройка панелей инструментов). Это окно настройки также можно открыть с помощью операции контекстного меню **Customize** (Настройка) при позиционировании курсора на специальной панели инструментов.

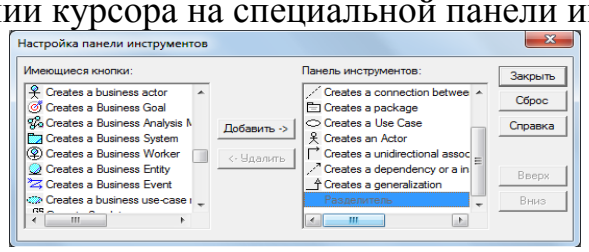


Рисунок 6 – Диалоговое окно настройки специальной панели инструментов для диаграммы вариантов использования

Добавление актера на диаграмму вариантов использования и редактирование его свойств

Для добавления актера на диаграмму *варианта использования* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы актера на специальной панели инструментов и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение актера с маркерами изменения его геометрических размеров и предложенным программой именем по умолчанию NewClass, которое мы меняем на Клиент Банкомата (рис. 7).

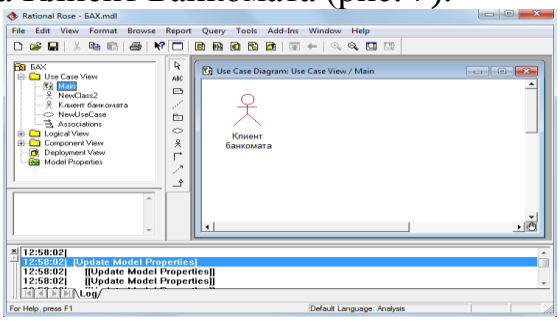


Рисунок 7 – Диаграмма вариантов использования после добавления на нее актера

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

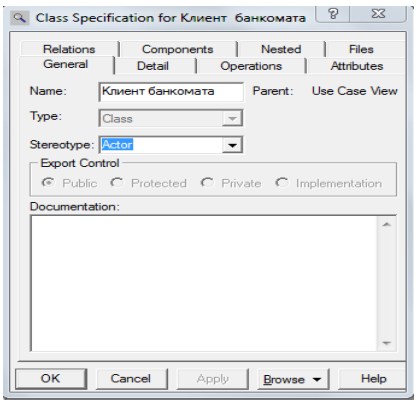


Рисунок 8 – Диалоговое окно спецификации свойств актера Клиент Банкомата

Хотя в среде IBM Rational Rose актер является классом, для него некорректно специфицировать атрибуты и операции, поскольку актер является внешней по отношению к разрабатываемой системе сущностью.

Для актера Клиент Банкомата можно уточнить его назначение в модели. С этой целью следует изменить его *стереотип* и добавить текст документации. Для изменения *стереотипа* во вложенном списке **Stereotype** нужно выбрать строку **Business Actor** (*бизнес–актер*). Для добавления текста документации в секцию **Documentation** следует ввести текст: «Любое физическое лицо, пользующееся услугами банкомата» и нажать кнопку **Apply** (Применить) или **OK**.

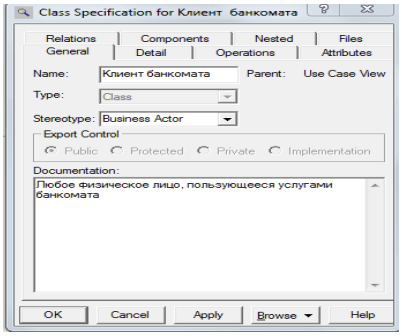


Рисунок 9 – Диалоговое окно спецификации свойств после изменения стереотипа и добавления текста документации для актера Клиент Банкомата

Добавление и редактирование варианта использования

Для добавления *варианта использования* на диаграмму нужно с помощью левой кнопки мыши нажать кнопку с изображением *варианта использования* на специальной панели инструментов и щелкнуть левой кнопкой мыши на свободном месте диаграммы. На диаграмме появится изображение *варианта использования*, дадим имя Снятие наличных по кредитной карточке (рис.10).

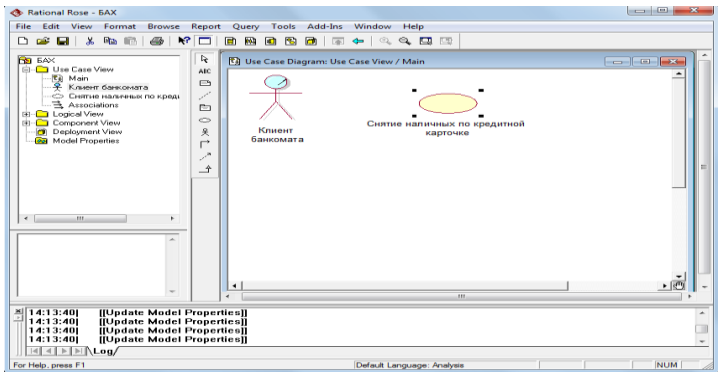


Рисунок 10 – Диаграмма вариантов использования после добавления на нее варианта использования

Зададим стереотип **Business Use Case**. Для добавления текста документации в секцию **Documentation** следует ввести текст: «Основной вариант использования для разрабатываемой модели банкомата» и нажать кнопку **Apply** (Применить) или **OK**.

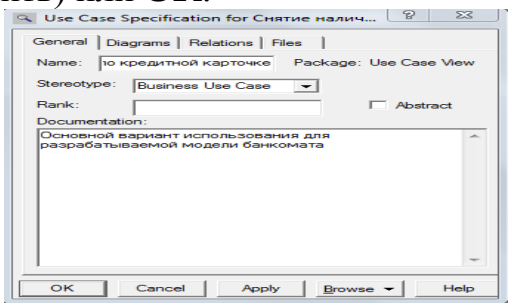


Рисунок 11 – Диалоговое окно спецификации свойств варианта использования Снятие наличных по кредитной карточке

Добавление ассоциации

Для добавления *ассоциации* между актером и вариантом использования на диаграмму нужно с помощью левой кнопки мыши нажать на специальной панели инструментов кнопку с изображением пиктограммы направленной *ассоциации*, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении актера на диаграмме и отпустить ее на изображении *варианта использования*. В результате этих действий на диаграмме появится изображение *ассоциации*, соединяющей актера с вариантом использования (рис. 12).

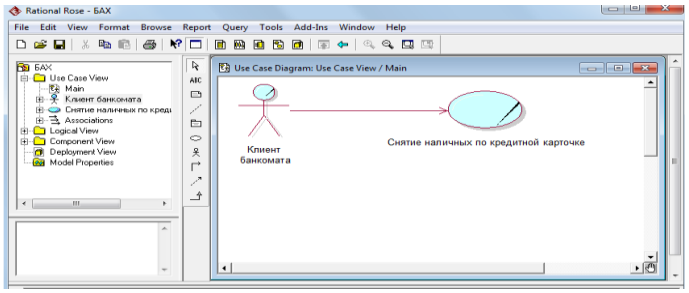


Рисунок 12 – Диаграмма вариантов использования после добавления на нее направленной ассоциации

При необходимости можно сделать направленную *ассоциацию* ненаправленной, для чего следует воспользоваться диалоговым окном свойств *ассоциации*. Открыть это окно можно, например, двойным щелчком на изображении линии *ассоциации* на диаграмме, после чего убрать отметку

строки выбора **Navigable** (Навигация) на вкладке **Role A Detail** (Детальные свойства концевой точки *ассоциации* A).

Добавление отношения зависимости и редактирование его свойств

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

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

После задания для данного отношения *зависимости стереотипа* <<include>> текст этого *стереотипа* в угловых скобках появится рядом с изображением пунктирной линии зависимости, связывающей соответствующие варианты использования (рис. 13).

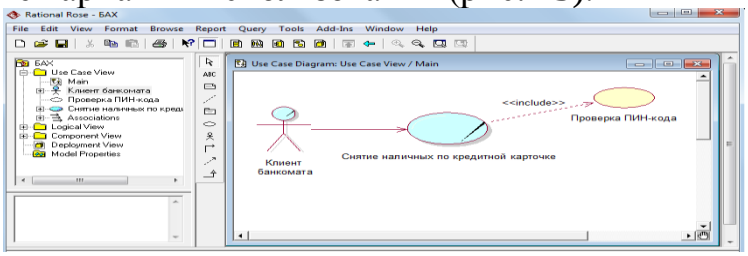


Рисунок 13 – Диаграмма вариантов использования после добавления на нее отношения зависимости

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

Окончательное построение диаграммы вариантов использования

К отдельному *варианту использования* можно добавить текстовый файл с описанием сценария его выполнения. Для этого необходимо выделить этот *вариант использования* в браузере проекта и выполнить операцию контекстное меню: **New→File** (Новый→Файл). В результате этого будет вызвано стандартное окно открытия файла, в котором необходимо задать имя предварительно созданного с помощью офисной программы MS Word добавляемого файла. После нажатия кнопки **Открыть** пиктограмма

добавленного файла появится в браузере проекта ниже соответствующего *варианта использования*. В последующем можно вернуться к редактированию этого файла сценария, выполнив двойной щелчок на этой пиктограмме. При этом файл сценария будет открыт в соответствующем приложении – в текстовом процессоре MS Word.

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

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

При этом отношение зависимости со стереотипом <<extend>> на данной диаграмме означает следующее. Вариант использования Блокирование кредитной карточки будет выполняться только в том случае, если в результате проверки ПИН-кода будет установлено, что соответствующая кредитная карточка утрачена ее владельцем или признана недействительной. Построенная таким образом диаграмма вариантов использования будет иметь следующий вид (рис. 14).

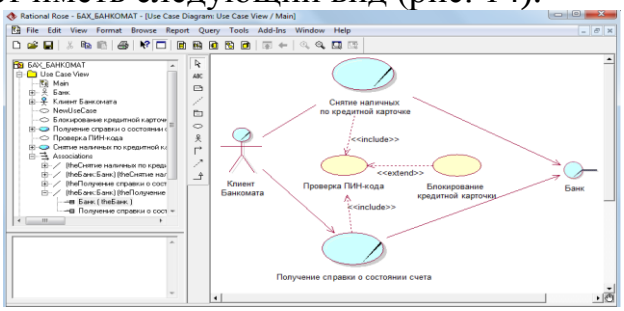


Рисунок 14 – Окончательный вид диаграммы вариантов использования для разрабатываемой модели банкоматов

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

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

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

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

Задание 2. Разработка диаграммы классов и редактирование их свойств

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

- окно диаграммы *классов* появляется по умолчанию в рабочем окне диаграммы после создания нового проекта;
- щелкнуть на кнопке с изображением диаграммы *классов* на стандартной панели инструментов;
- раскрыть логическое представление (Logical View) в браузере проекта и дважды щелкнуть на пиктограмме **Main** (Главная);
- выполнить операцию главного меню: **Browse→Class Diagram** (Обзор→Диаграмма *классов*).

При этом появляется новое окно с чистым рабочим листом диаграммы *классов* и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы *классов* (табл. 12). Назначение отдельных кнопок панели можно узнать также из всплывающих подсказок.

Таблица 12 – Назначение кнопок специальной панели инструментов для диаграммы классов

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

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

Добавление класса на диаграмму классов и редактирование его свойств

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

Продолжая разработку модели банкомата в качестве сквозного примера проекта, построим для этой модели следующую каноническую диаграмму – диаграмму *классов*. С этой целью следует изменить предложенное по умолчанию имя диаграммы Main на Диаграмма *классов* ATM, а имя добавленного на диаграмму *класса* – на Транзакция Банкомата (рис. 15).

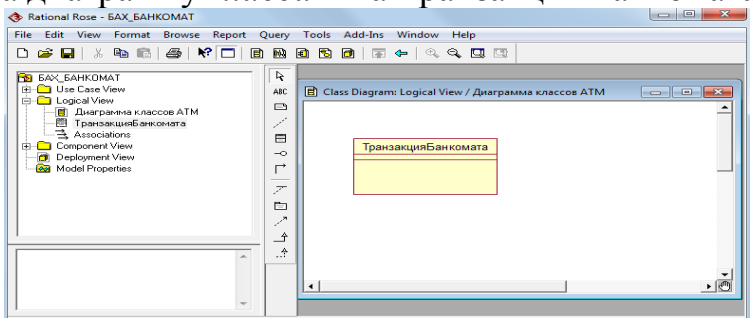


Рисунок 15 – Диаграмма классов модели банкомата после добавления на нее класса Транзакция Банкомата

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

Для *класса* ТранзакцияБанкомата можно уточнить его назначение в модели с помощью указания стереотипа и пояснительного текста в форме документации. С этой целью двойным щелчком левой кнопкой мыши на изображении этого *класса* на диаграмме или в браузере проекта следует открыть диалоговое окно спецификации свойств этого *класса* (рис. 16) и на вкладке **General** (Общие) выбрать из вложенного списка **Stereotype** стереотип **entity** (сущность).

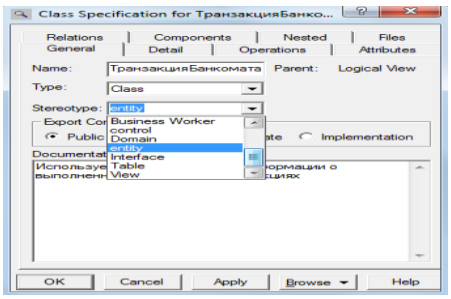


Рисунок 16 – Диалоговое окно спецификации свойств класса Транзакция Банкомата при выборе из вложенного списка стереотипа entity

Выбор данного стереотипа означает, что соответствующий *класс* предназначен для хранения информации, которая должна сохраняться в системе после уничтожения объектов данного *класса*. Далее в секцию документации данного *класса* можно ввести поясняющий текст: "Используется для сохранения информации о выполненных банкоматом транзакциях" и нажать кнопку **Apply** или **OK**, чтобы сохранить результаты редактирования свойств выбранного *класса*. После назначения стереотипа классу Транзакция банкомата текст данного стереотипа в угловых скобках появится выше имени данного *класса* (рис. 17).

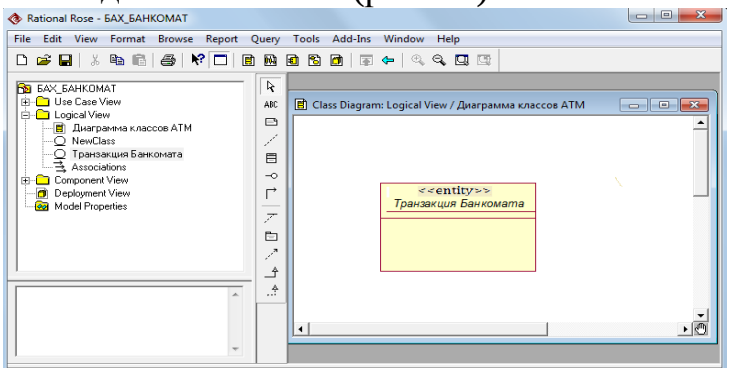


Рисунок 17 – Диаграмма классов модели банкомата после выбора стереотипа для класса Транзакция Банкомата

Для отдельного *класса* можно уточнить также и другие его свойства, доступные для редактирования на вкладке **Detail** (Подробно) окна спецификации свойств этого *класса*. Например, на этой вкладке с помощью вложенного списка **Multiplicity** (Кратность) можно задать количество объектов или экземпляров данного *класса*, для чего следует выбрать строку с буквой n. Данное значение означает, что у *класса* Транзакция банкомата может быть любое конечное число экземпляров (рис. 18). Поле ввода с именем **Space** (Пространство) служит для указания объема абсолютной или относительной памяти, которая требуется, по оценке разработчика, для реализации каждого объекта данного *класса*. Применительно к рассматриваемой модели это поле можно оставить пустым.

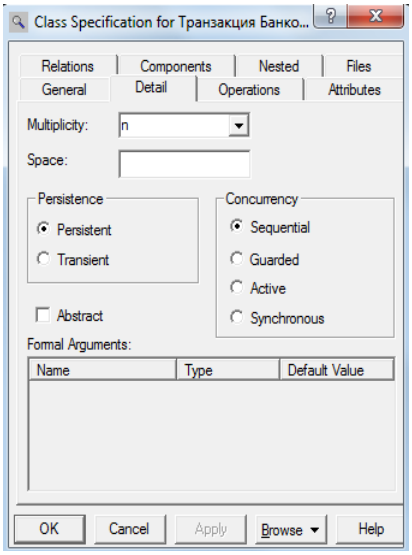


Рисунок 18 – Диалоговое окно спецификации свойств класса Транзакция Банкомата, открытое на вкладке Detail (Подробно)

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

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

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

различных потоках управления, поскольку объекты данного *класса* могут самостоятельно разрешать возможные конфликты.

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

Следует заметить, что для предотвращения потери информации о разрабатываемой модели и результатов редактирования свойств ее графических элементов необходимо периодически сохранять модель во внешнем файле. Для этого следует выполнить операцию главного меню: **File→Save** (Файл→Сохранить) или нажать комбинацию клавиш: **Ctrl+S.**

Стереотипы классов и их графическое представление

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

Изменить изображение стереотипа для отдельного *класса* можно, например, с помощью одной из вложенных операций контекстного меню: **Options→Stereotype Display** (Параметры→Изображение стереотипа). В качестве примера можно представить изображение *класса* Транзакция Банкомата в форме специальной графической пиктограммы стереотипа. С этой целью следует выполнить операцию контекстного меню: **Options→ Stereotype Display→Icon** (Параметры→Изображение стереотипа→ Пиктограмма). Соответствующее графическое изображение стереотипа <<entity>> для *класса* Транзакция Банкомата в форме пиктограммы будет иметь следующий вид (рис. 2.5, а).

Для сравнения можно выбрать изображение *класса* Транзакция Банкомата в форме декоративного графического стереотипа. С этой целью выполним операцию контекстного меню: **Options→Stereotype Display→ Decoration** (Параметры→Изображение стереотипа→Декорация). Соответствующее графическое изображение стереотипа <<entity>> для *класса* Транзакция Банкомата в форме декорации будет иметь следующий вид (рис. 19,20).

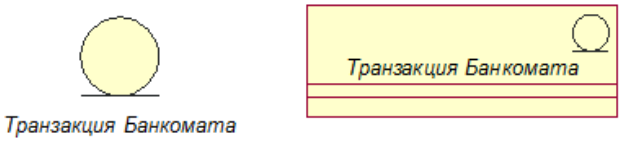


Рисунок 19 – Графические способы изображения стереотипа <<entity>> для класса Транзакция Банкомата

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

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

Добавим на диаграмму второй *класс* с именем Контроллер Банкомата, для которого в окне спецификации свойств выберем стереотип **control** (*управляющий класс*), а в качестве документации введем текст: "Реализует логику функционирования банкомата". При этом атрибуты и операции у данного *класса* будут отсутствовать.

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

Далее следует добавить *класс* с именем IКонтроллер Банка, для которого выбрать стереотип **Interface** (*Интерфейс*), означающий, что банкомат пользуется услугами Банка при обработке своих транзакций. Заметим, что первой буквой в имени этого *класса* является английское "I", которое служит в языке UML для указания *интерфейса*. Соответствующий фрагмент диаграммы *классов* после добавления на нее *классов* Устройство чтения карточки и IКонтроллер Банка будет иметь следующий вид (рис. 2.6).

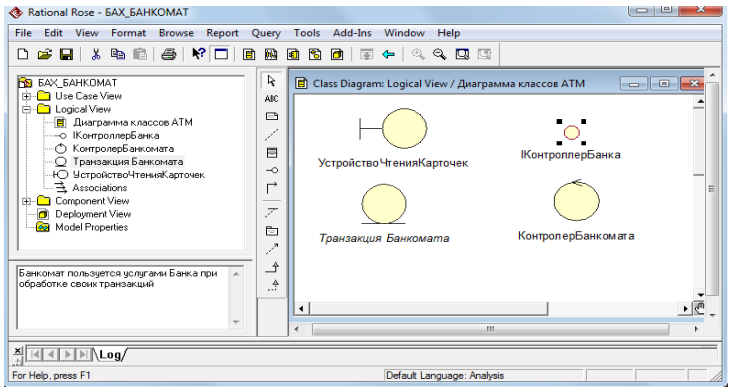


Рисунок 20 – Фрагмент диаграммы классов модели банкомата после добавления на нее классов Устройство чтения карточки и Контролер Банкомата

Добавление и редактирование атрибутов классов

Из всех графических элементов среды IBM Rational Rose 2003 класс обладает максимальным набором свойств, главными из которых являются его *атрибуты* и *операции*. Поскольку именно диаграмма классов используется в среде IBM Rational Rose 2003 для генерации программного кода, подробно рассмотрим соответствующие свойства *атрибутов* и *операций*.

Добавить *атрибут* к созданному ранее классу можно одним из следующих способов:

- С помощью *операции* контекстного меню **New Attribute** (Новый *атрибут*) для класса, выделенного на диаграмме классов. В этом случае активизируется курсор ввода текста в области графического изображения класса на диаграмме.
- С помощью *операции* контекстного меню: **New→Attribute** (Новый→ *Атрибут*) для класса, выделенного в браузере проекта. В этом случае активизируется курсор ввода текста в области иерархического представления класса в браузере проекта под именем соответствующего класса.
- С помощью *операции* контекстного меню **Insert** (Вставить), вызванного при позиционировании курсора в области открытой вкладки *атрибутов* в диалоговом окне свойств **Class Specification** соответствующего класса.

После добавления *атрибута* к классу по умолчанию ему присваивается имя name и некоторый *квантор видимости* (рис. 21).

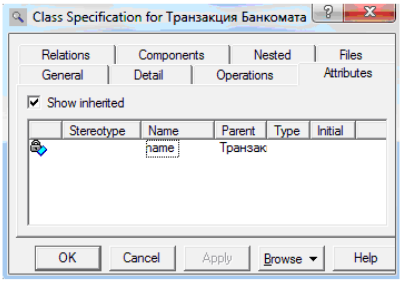


Рисунок 21 – Диалоговое окно спецификации свойств класса после добавления нового атрибута

Для рассматриваемой модели банкомата имя добавленного *атрибута* следует изменить на идентификатор карточки. Напомним, что имена *атрибутов* и *операций* классов должны начинаться со строчной буквы. Видимость *атрибутов* на диаграмме классов изображается в форме специальных пиктограмм или украшений. Используемые пиктограммы видимости изображаются перед именем соответствующего *атрибута* и имеют следующий смысл (табл. 13).

Таблица 13 – Пиктограммы видимости атрибутов классов

Графическое изображение	**Текстовый аналог**	**Назначение пиктограммы**
	Public	Общедоступный или открытый. В нотации языка UML такому *атрибуту* соответствует знак «+»
	Protected	Защищенный. В нотации языка UML такому *атрибуту* соответствует знак «#»
	Private	Закрытый. В нотации языка UML такому *атрибуту* соответствует знак «-»
	Implementation	Реализация. В нотации языка UML такому *атрибуту* соответствует знак «~»

Для редактирования свойств *атрибутов* предназначено специальное диалоговое окно спецификации *атрибута* **Class Attribute Specification**, которое открывается двойным щелчком мыши на строке выбранного *атрибута* в окне спецификации свойств класса. В окне свойств отдельного *атрибута* класса можно задать *тип* данных *атрибута* и его начальное *значение*, а также назначить *атрибуту* стереотип из раскрывающегося списка или изменить его *квантор видимости*.

Для *атрибута* идентификатор карточки в качестве *типа его допустимых значений* из вложенного списка **Type** следует выбрать тип **Integer** (целочисленный), а для задания *квантора видимости* следует выбрать в группе **Export Control** (Управление экспортом) *квантор* **Public**. Поскольку начальное *значение* для данного *атрибута* не определено, соответствующее поле ввода следует оставить пустым. В секцию документации данного *атрибута* класса можно ввести поясняющий текст: «Устройство чтения карточки считывает значение этого *атрибута* с кредитной карточки клиента» и нажать кнопку **Apply** или **OK**, чтобы сохранить результаты редактирования этих свойств *атрибута*. Соответствующее окно спецификации свойств *атрибута* идентификатор карточки после редактирования его общих свойств будет иметь следующий вид (рис. 22).

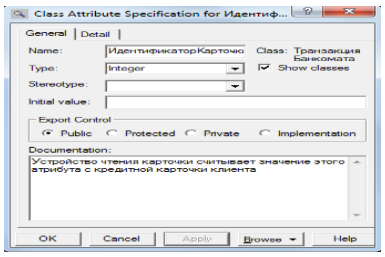


Рисунок .22 – Диалоговое окно спецификации свойств атрибута идентификатор карточки после его редактирования

Для отдельного *атрибута* можно также определить дополнительные свойства, доступные для редактирования на вкладке **Detail** (Подробно) диалогового окна спецификации свойств выбранного *атрибута* (рис. 23).

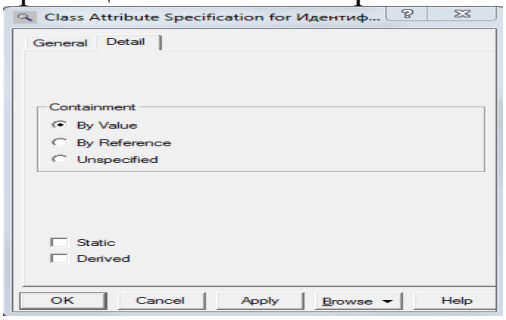


Рисунок 23 – Диалоговое окно спецификации свойств атрибута идентификатор карточки, открытое на вкладке Detail (Подробно)

На вкладке **Detail** в группе выбора **Containment** (Локализация) можно специфицировать условия хранения *атрибута* у объектов выбранного класса. Для выбора могут быть использованы следующие свойства:

- **By value** (По значению) – свойство по умолчанию, которое означает, что значения *атрибута* хранятся в пределах адресного пространства, выделенного для объекта данного класса. Например, если имеется атрибут типа **String**, то значение этой строки содержится в пределах определения класса.
- **By reference** (По ссылке) – означает, что значение *атрибута* хранится вне адресного пространства, выделенного для объекта данного класса, но у объектов класса имеется указатель на этот атрибут.
- **Unspecified** (Не определен) – означает, что метод локализации данного *атрибута* не определен. В этом случае при генерации программного кода для данного *атрибута* по умолчанию выбирается значение **By value**.

Далее можно определить атрибут как статичный, выставив отметку в строке выбора **Static**. Статичный атрибут по определению имеет одно и тоже значение для всех объектов рассматриваемого класса. Наконец, на вкладке **Detail** можно определить атрибут как производный, выставив отметку в строке выбора **Derived**. Значение производного *атрибута* по определению может быть вычислено на основании значений других *атрибутов* этого или другого класса.

Добавление и редактирование операций классов

Функционирование банкомата основано на выполнении отдельными его устройствами тех или иных действий. В модели структуры банкомата все действия представляются с помощью *операций* классов. Таким образом, следующий этап разработки диаграммы классов связан со спецификацией *операций* классов.

Добавить *операцию* к созданному ранее классу можно одним из следующих способов:

- С помощью *операции* контекстного меню **New Operation** (Новая *операция*) для класса, выделенного на диаграмме классов. В этом случае активизируется курсор ввода в области графического изображения класса на диаграмме.
- С помощью *операции* контекстного меню: **New→Operation** (Новая→ *Операция*) для класса, выделенного в браузере проекта. В этом случае активизируется курсор ввода в области иерархического представления класса в браузере под именем соответствующего класса.
- С помощью *операции* контекстного меню **Insert** (Вставить), вызванного при позиционировании курсора в области открытой вкладки *операций* в диалоговом окне свойств **Class Specification** соответствующего класса.

После добавления *операции* к классу по умолчанию ей присваивается имя **opname** и некоторый *квантор видимости*. Видимость *операций* на диаграмме классов также изображается в форме специальных пиктограмм или украшений. Используемые пиктограммы видимости изображаются перед именем соответствующей *операции* и имеют определенный смысл.

Таблица 14 – Пиктограммы видимости операций классов

Графическое изображение	**Текстовый аналог**	**Назначение пиктограммы**
	Public	Общедоступный или открытый. В нотации языка UML такому *атрибуту* соответствует знак «+»
	Protected	Защищенный. В нотации языка UML такому *атрибуту* соответствует знак «#»
	Private	Закрытый. В нотации языка UML такому *атрибуту* соответствует знак «-»
	Implementation	Реализация. В нотации языка UML такому *атрибуту* соответствует знак «~»

В контексте рассматриваемой модели банкомата в качестве имени первой *операции* для класса Транзакция Банкомата следует задать: создать новую транзакцию. При этом скобки при задании имени *операции* не записываются, поскольку программа IBM Rational Rose 2003 добавляет их автоматически. Однако, следуя правилам именования *операций* в языке UML, в тексте имена *операций* будут указываться со скобками.

Каждая из *операций* классов имеет собственное диалоговое окно спецификации свойств **Operation Specification**, которое может быть открыто по двойному щелчку на имени *операции* на соответствующей вкладке спецификации класса или на имени этой *операции* в браузере проекта. Для *операции* создать новую транзакцию() в качестве *квантора видимости* следует выбрать из вложенного списка *квантор* **public**. В секцию документации данной *операции* класса можно ввести поясняющий текст: «Вызывается после того, как кредитная карточка вставлена в Устройство чтения карточки» и нажать кнопку **Apply** или **OK**, чтобы сохранить результаты редактирования свойств этой *операции*. Соответствующее окно спецификации свойств *операции* создать новую транзакцию() после редактирования ее свойств будет иметь следующий вид (рис. 24).

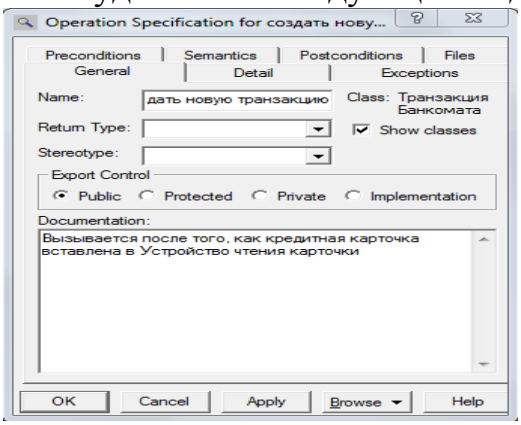


Рисунок 24 – Диалоговое окно спецификации свойств операции создать новую транзакцию()

Для *операций* классов кроме *квантора видимости* можно также задать: *аргументы* и их тип, *тип возвращаемого результата*, стереотип *операции*, а также определить протокол и размер, задать исключительные ситуации, специфицировать предусловия и постусловия и целый ряд других свойств. Для отдельной *операции* эти дополнительные свойства доступны для редактирования на вкладке **Detail** (Подробно) диалогового окна спецификации свойств выбранной *операции* (рис.25).

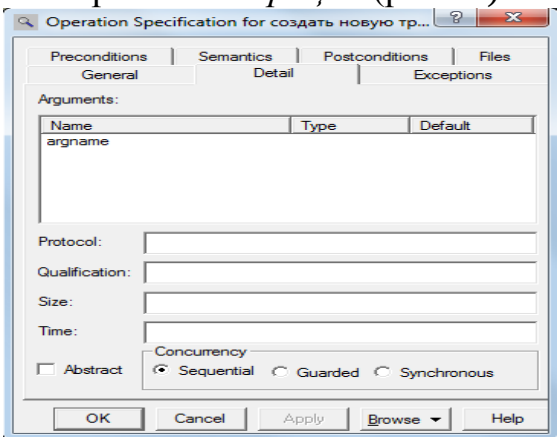


Рисунок 25 – Диалоговое окно спецификации свойств операции создать новую транзакцию(), открытое на вкладке Detail (Подробно)

На вкладке **Detail** в многостраничном поле **Arguments** (Аргументы) можно определить *аргументы редактируемой операции*. Для этого следует выполнить *операцию* контекстного меню **Insert** (Вставить). После этого в этом поле появится аргумент данной *операции* с именем по умолчанию **argname**. Для редактирования свойств аргумента предназначено специальное окно свойств аргумента.

На вкладке **Detail** в поле **Protocol** (Протокол) можно специфицировать порядок выполнения *операций* класса, например, указать, что одна *операция* не может быть вызвана раньше другой. Соответствующий текст в данное поле вводится с клавиатуры и попадает в генерируемый код в форме комментария. В поле **Qualification** (Квалификация) можно уточнить детали реализации *операции*, связанные с конкретным языком программирования. Соответствующий текст также вводится в данное поле с клавиатуры и попадает в генерируемый код в форме комментария.

Далее на этой же вкладке в полях **Size** (Размер) и **Time** (Время) можно специфицировать предполагаемый объем памяти и время, необходимое для выполнения *операции*. Соответствующая информация попадает в генерируемый код в форме комментария.

В группе выбора **Concurrency** (Параллельность) можно специфицировать условия на возможность параллельного выполнения данной *операции*. Для выбора могут быть использованы следующие свойства:

- **Sequential** (Последовательная) – свойство по умолчанию, которое означает, что данная *операция* класса может быть выполнена только при наличии одного потока управления, т.е. соответствующая *операция* класса должна выполняться последовательно. При наличии нескольких потоков управления выполнение данной *операции* класса не гарантируется.
- **Guarded** (Безопасная) – означает, что при наличии нескольких потоков управления выполнение данной *операции* класса гарантируется только в том случае, когда обеспечено взаимодействие объектов друг с другом в различных потоках.
- **Synchronous** (Синхронная) – означает, что выполнение данной *операции* класса гарантируется при наличии нескольких потоков управления. При этом нет необходимости во взаимодействии объектов в различных потоках управления, поскольку данная *операция* класса будет выполняться в отдельном потоке управления вплоть до своего завершения.

Применительно к рассматриваемой модели для *операции* создать новую транзакцию() следует выбрать свойство **Sequential**, а поля всех других свойств оставить пустыми.

Спецификация атрибутов и операций для класса Транзакция Банкомата

Чтобы закончить спецификацию класса Транзакция Банкомата аналогичным способом следует добавить еще 3 *атрибута* и 2 *операции* со следующими свойствами:

- значение ПИН-кода карточки с *квантором видимости* **public**. В качестве типа этого *атрибута* следует выбрать тип **Integer** (целочисленный), а в секцию документации *атрибута* ввести поясняющий текст: «Устройство чтения карточки считывает значение этого *атрибута* с кредитной карточки клиента».
- введенный ПИН-код с *квантором видимости* **public**. В качестве типа этого *атрибута* следует выбрать тип **Integer** (целочисленный), а в секцию документации *атрибута* ввести поясняющий текст: «Значение этого *атрибута* вводится клиентом с клавиатуры банкомата».
- введенная сумма наличных с *квантором видимости* **public**. В качестве типа этого *атрибута* следует выбрать тип **Currency** (Денежный), а в секцию документации *атрибута* ввести поясняющий текст: «Значение этого *атрибута* вводится клиентом с клавиатуры банкомата».
- проверить правильность ПИН-кода() с *квантором видимости* **public**. В качестве *типа возвращаемого результата* для этой *операции* следует выбрать тип **Boolean** (логический), а в секцию ее документации ввести поясняющий текст: «Вызывается после того, как клиент ввел значение ПИН-кода с клавиатуры банкомата».
- завершить транзакцию() с *квантором видимости* **public**. В секцию ее документации ввести поясняющий текст: «Вызывается после завершения всех действий банкомата по обслуживанию клиента».

Выполнить эти действия предлагается читателям самостоятельно. Соответствующий фрагмент диаграммы классов после добавления и спецификации *атрибутов* и *операций* для класса Транзакция Банкомата будет иметь следующий вид (рис. 26).

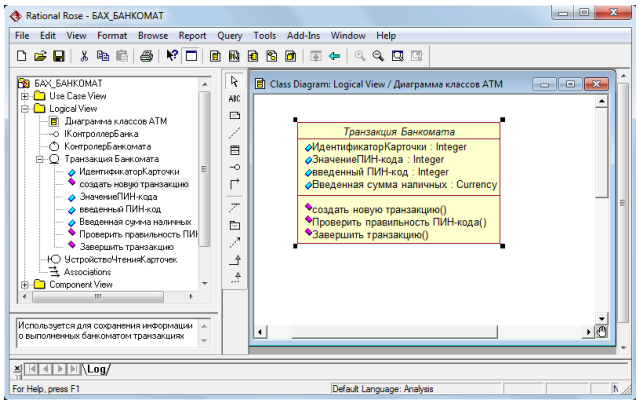


Рисунок 26 – Фрагмент диаграммы классов модели банкомата после добавления атрибутов и операций для класса Транзакция банкомата

Задание 3. Добавление отношений на диаграмму классов и редактирование их свойств

Диаграмма классов является логическим представлением структуры модели, поэтому она должна содержать столько классов, сколько необходимо для реализации всего проекта. При этом для полного представления структуры модели необходимо установить и специфицировать отношения между классами.

Добавление ассоциации на диаграмму классов и редактирование ее свойств

Добавление на диаграмму *ассоциации* между двумя классами выполняется следующим образом. На специальной панели инструментов необходимо нажать кнопку с изображением пиктограммы направленной *ассоциации* и отпустить левую кнопку мыши. Если *ассоциация* – направленная, то на диаграмме классов надо выделить первый элемент *ассоциации* или источник, от которого исходит стрелка, и, не отпуская нажатую левую кнопку мыши, переместить ее указатель ко второму элементу отношения или приемнику, к которому направлена стрелка. После перемещения ко второму элементу кнопку мыши следует отпустить, в результате чего на диаграмму классов будет добавлена направленная *ассоциация* с именем Untitled между двумя выбранными классами.

Продолжая разработку диаграммы классов модели банкомата, добавим на нее описанным способом направленную *ассоциацию* между классом Контроллер Банкомата и классом Транзакция Банкомата (рис. 27).

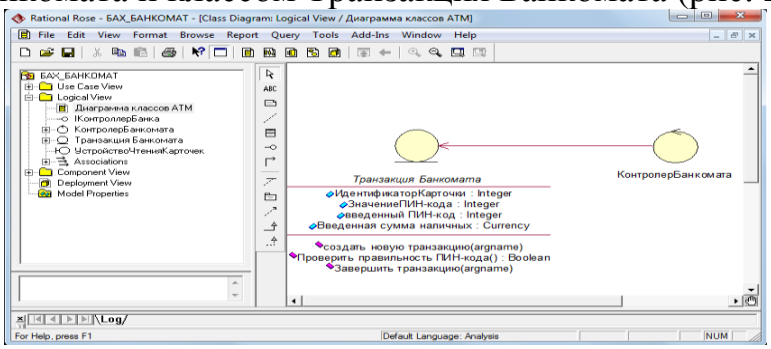


Рисунок 27 – Фрагмент диаграммы классов модели банкомата после добавления на неё направленной ассоциации

Изменим имя для данной *ассоциации*, предложенное средой по умолчанию. Это можно выполнить с помощью окна спецификации свойств *ассоциации*. Доступ к диалоговому окну спецификации свойств *ассоциации* **Association Specification** можно получить после выделения линии *ассоциации* на диаграмме классов или в браузере проекта и двойного щелчка на ней левой кнопки мыши (рис. 28).

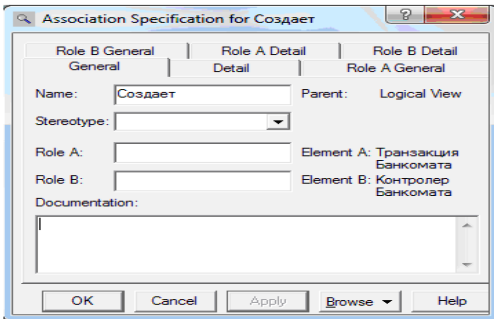


Рисунок 28 – Диалоговое окно спецификации свойств ассоциации

Для задания имени *ассоциации* следует на вкладке **General** (Общие) в поле ввода **Name** (Имя) ввести текст ее имени: Создает и нажать кнопку **Apply** или **OK**, чтобы сохранить результаты редактирования имени *ассоциации*. Для *ассоциации* можно задать также *кратность* каждого из концов *ассоциации*, стереотип, использовать ограничения и роли, а также некоторые другие свойства.

Для добавленной на диаграмму классов *ассоциации* зададим *кратность* конца *ассоциации* у класса Контроллер Банкомата, равную 1. Для этого следует в окне спецификации свойств *ассоциации* перейти на вкладку **Role B Detail** и выбрать значение 1 из вложенного списка **Multiplicity**. Аналогичным образом следует задать *кратность* конца *ассоциации* у класса Транзакция Банкомата равную 1..n, для чего на вкладке **Role A Detail** и следует выбрать значение 1..n из вложенного списка **Multiplicity**. Содержательно это будет означать, что каждый объект класса Контроллер Банкомата может быть связан с одним или несколькими объектами класса Транзакция Банкомата.

Если *ассоциация* является ненаправленной, то порядок выбора классов может быть произвольный, а после добавления *ассоциации* на диаграмму классов следует изменить значение соответствующего свойства данной *ассоциации*. С этой целью необходимо перейти на вкладку **Role A Detail** в окне спецификации свойств *ассоциации* и убрать отметку у свойства **Navigable** (Навигация).

Добавление отношений агрегации и композиции на диаграмму классов и редактирование их свойств

Добавить на диаграмму отношение *агрегации* между двумя классами можно следующими способами:

- Щелкнуть на кнопке с изображением отношение *агрегации* на специальной панели инструментов и провести линию *агрегации* от одного класса к другому.
- Провести линию *ассоциации* между выбранными классами и изменить ее свойства таким образом, чтобы превратить данную *ассоциацию* в *агрегацию*.

В первом случае может оказаться, что по умолчанию на специальной панели инструментов диаграммы классов отсутствует кнопка с пиктограммой *агрегации*. В этом случае необходимо предварительно

добавить ее на панель инструментов одним из описанных ранее способов. Во втором случае следует открыть окно спецификации свойств *ассоциации* **Association Specification** и на вкладке деталей соответствующего конца *ассоциации* выставить отметку в строке выбора **Aggregate** (*Агрегация*).

В качестве примера изменим тип созданной ранее *ассоциации* и сделаем ее агрегацией. Содержательно это будет означать, что класс Контроллер Банкомата будет включать в себя в качестве составной части класс Транзакция Банкомата; при этом уничтожение любого объекта класса Контроллер Банкомата не должно привести к уничтожению ассоциированных с ним объектов класса Транзакция Банкомата. С этой целью на вкладке **Role B Detail** деталей конца *ассоциации* класса Контроллер Банкомата следует выставить отметку в строке выбора **Aggregate** (рис. 29).

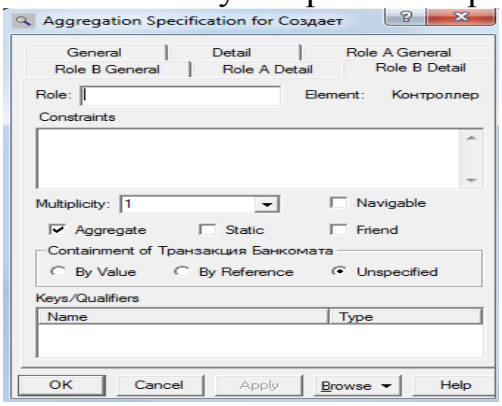


Рисунок 29 – Диалоговое окно спецификации свойств ассоциации

Соответствующий фрагмент диаграммы классов после изменения *ассоциации* между классами Контроллер Банкомата и Транзакция Банкомата на отношение *агрегации* будет иметь следующий вид (рис. 30).

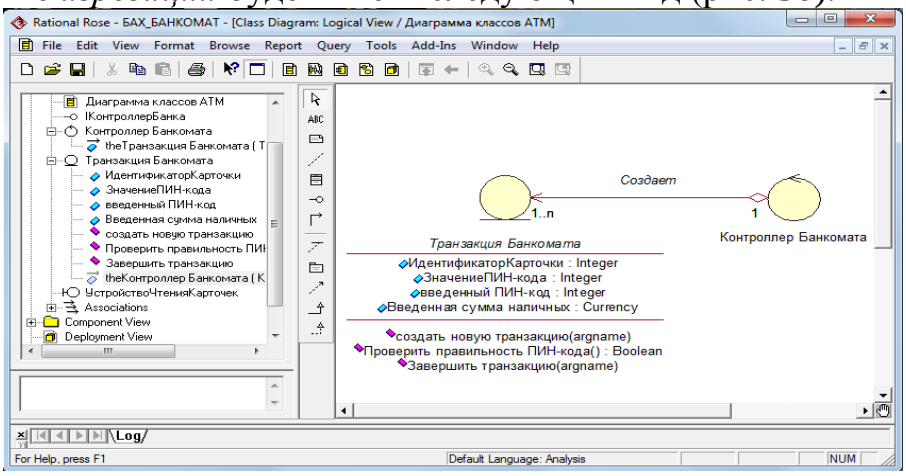


Рисунок 30 – Фрагмент диаграммы классов модели банкомата после добавления на нее отношения агрегации

Для изображения отношения *композиции* можно также вначале изобразить обычную *ассоциацию*, после чего, открыв окно ее свойств на вкладке деталей соответствующего конца *ассоциации*, (рис. 6.3) выставить отметку в строке выбора **Aggregate** (*Агрегация*) и в секции **Containment**

(Локализация) выбрать опцию **By Value** (По значению). По умолчанию эта опция не специфицирована, т.е. выставлена отметка опции **Unspecified**.

Добавление отношения обобщения на диаграмму классов и редактирование ее свойств

Добавление на диаграмму отношения *обобщения* между двумя классами выполняется следующим образом. На специальной панели инструментов необходимо нажать кнопку с изображением пиктограммы *обобщения* и отпустить левую кнопку мыши. Далее на диаграмме классов надо выделить первый элемент *обобщения* или потомок, от которого исходит стрелка, и, не отпуская нажатую левую кнопку мыши, переместить ее указатель ко второму элементу отношения или предку, к которому направлена стрелка. После перемещения ко второму элементу кнопку мыши следует отпустить, в результате чего на диаграмму классов будет добавлена линия *обобщения* с именем Untitled между двумя выбранными классами.

Продолжая разработку диаграммы классов модели банкомата, добавим на нее описанным способом направленную *ассоциацию* между классом Контроллер Банкомата и дополнительно созданным абстрактным классом Контроллер (рис. 31). Последний класс может быть предназначен для спецификации системных атрибутов и операций, необходимых при исполнении соответствующей программы. Напомним, что на абстрактный характер класса указывает написание курсивом его имени, а для спецификации данного свойства класса необходимо на вкладке **Detail** (Подробно) окна спецификации свойств класса Контроллер выставить отметку в строке выбора **Abstract**.

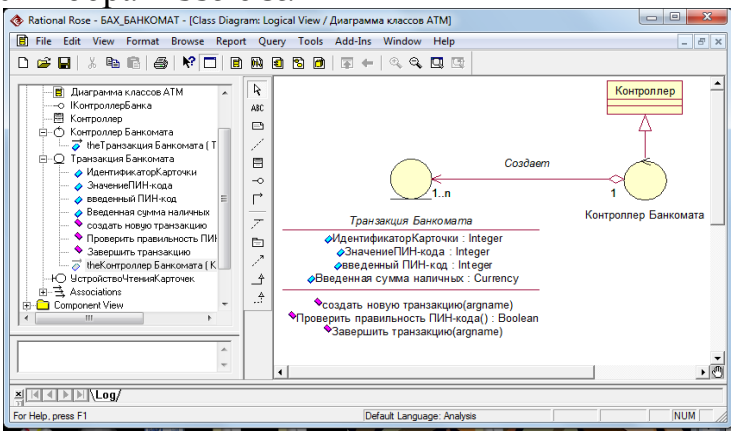


Рисунок 31 – Диаграмма классов модели банкомата после добавления на неё отношения обобщения

Изменим имя отношения *обобщения*, предложенное средой по умолчанию. Это можно выполнить с помощью окна спецификации свойств *обобщения*. Доступ к диалоговому окну спецификации свойств отношения *обобщения* **Generalize Specification** можно получить после выделения линии *обобщения* на диаграмме классов или в браузере проекта и двойного щелчка на ней левой кнопки мыши (рис. 32).

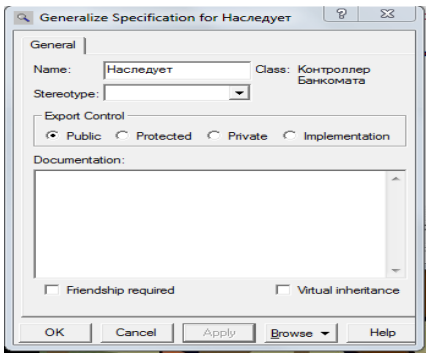


Рисунок 32 – Диалоговое окно спецификации свойств отношения обобщения

Для задания имени *обобщения* следует на единственной вкладке **General** (Общие) в поле ввода **Name** (Имя) ввести текст ее имени: Наследует и нажать кнопку **Apply** или **OK**, чтобы сохранить результаты редактирования имени *ассоциации*.

Окончательное построение диаграммы классов модели банкомата

Для окончательного построения диаграммы классов рассматриваемой модели банкомата следует описанным выше способом добавить оставшиеся классы и *ассоциации*, а также специфицировать стереотипы, атрибуты и операции этих классов. С этой целью следует выполнить следующие действия:

1. Для класса IКонтроллерБанка добавить операцию: проверить идентификатор карточки (идентификатор карточки: Integer) с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Boolean** (логический), а в качестве целочисленного аргумента задать идентификатор карточки. Для задания аргумента необходимо перейти на вкладку **Detail** (Подробно) окна спецификации свойств данной операции и после добавления аргумента с помощью операции контекстного меню **Insert** ввести имя аргумента и его тип **Integer** в соответствующие поля ввода.
2. Для класса IКонтроллерБанка добавить операцию: открыть счет клиента (идентификатор карточки: Integer) с квантором видимости **public**. В качестве целочисленного аргумента этой операции следует задать идентификатор карточки.
3. Для класса IКонтроллерБанка добавить операцию: проверить баланс клиента (идентификатор карточки: Integer, введенная сумма наличных: Currency) с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Boolean** (логический). В качестве первого целочисленного аргумента этой операции следует задать идентификатор карточки, а в качестве второго аргумента – введенная сумма наличных с типом **Currency** (Денежный).
4. Для класса IКонтроллерБанка добавить операцию: уменьшить счет клиента (идентификатор карточки: Integer, введенная сумма наличных: Currency) с квантором видимости **public**. В качестве типа возвращаемого

результата для этой операции следует выбрать тип **Boolean** (логический). В качестве первого целочисленного аргумента этой операции следует задать идентификатор карточки, а в качестве второго аргумента – введенная сумма наличных с типом **Currency** (Денежный).

5. Для класса Устройство чтения карточки добавить операцию: прочитать идентификатор карточки() с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Integer** (целочисленный), а в секцию документации данной операции следует ввести поясняющий текст: «Вызывается после того, как кредитная карточка вставлена в Устройство чтения карточки».

6. Для класса Устройство чтения карточки добавить операцию: прочитать ПИН-код() с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Integer** (целочисленный), а в секцию документации данной операции следует ввести поясняющий текст: «Вызывается после того, как кредитная карточка вставлена в Устройство чтения карточки».

7. Для класса Устройство чтения карточки добавить операцию: вернуть кредитную карточку() с квантором видимости **public**. В секцию документации данной операции следует ввести поясняющий текст: «Вызывается после завершения транзакции».

8. Для класса Устройство чтения карточки добавить операцию: блокировать кредитную карточку() с квантором видимости **public**. В секцию документации данной операции следует ввести поясняющий текст: «Вызывается после того, как установлен факт утраты кредитной карточки владельцем».

9. Добавить класс с именем Экран Банкомата, для которого выбрать стереотип **boundary**. Данный класс также находится на границе моделируемой системы, на что и указывает этот стереотип. В секцию документации данного класса следует ввести поясняющий текст: «Устанавливается на банкомате».

10. Для класса Экран Банкомата добавить операцию: показать меню опций() с квантором видимости **public.**

11. Для класса Экран Банкомата добавить операцию: показать меню снятия суммы() с квантором видимости **public.**

12. Добавить класс с именем Клавиатура Банкомата, для которого выбрать стереотип **boundary**. В секцию документации данного класса следует ввести поясняющий текст: «Устанавливается на банкомате».

13. Для класса Клавиатура Банкомата добавить операцию: ввести ПИН-код() с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Integer**, а в секцию документации данной операции следует ввести поясняющий текст: «Вызывается после того, как клиент ввел значение ПИН-кода с клавиатуры».

14. Для класса Клавиатура Банкомата добавить операцию: ввести тип транзакции() с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Boolean** (логический), а в

секцию документации данной операции следует ввести поясняющий текст: «Возвращает значение Истина, если клиент выбирает снятие наличных, и значение Ложь, если клиент выбирает получение справки о состоянии счета».

15. Для класса Клавиатура Банкомата добавить операцию: ввести сумму снятия наличных() с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Currency** (Денежный), а в секцию документации данной операции следует ввести поясняющий текст: «Вызывается после того, как клиент ввел значение снимаемой суммы с клавиатуры».

16. Добавить класс с именем Устройство выдачи наличных, для которого выбрать стереотип **boundary**. В секцию документации данного класса следует ввести поясняющий текст: «Устанавливается на банкомате».

17. Для класса Устройство выдачи наличных добавить операцию: выдать наличные() с квантором видимости **public**. В секцию документации данной операции следует ввести поясняющий текст: «Вызывается после того, как выполнено снятие запрошенной клиентом суммы со счета».

18. Добавить класс с именем Принтер Банкомата, для которого выбрать стереотип **boundary**. В секцию документации данного класса следует ввести поясняющий текст: «Устанавливается на банкомате».

19. Для класса Принтер Банкомата добавить операцию: распечатать чек() с квантором видимости **public**. В секцию документации данной операции следует ввести поясняющий текст: «Вызывается по дополнительному запросу клиента».

20. Добавить направленную *ассоциацию* от класса Контроллер Банкомата к классу Устройство чтения карточки. В качестве *кратности* концов этой *ассоциации* установить значение 1.

21. Добавить направленную *ассоциацию* от класса Контроллер Банкомата к классу Принтер Банкомата. В качестве *кратности* концов этой *ассоциации* установить значение 1.

22. Добавить направленную *ассоциацию* от класса Контроллер Банкомата к классу Клавиатура Банкомата. В качестве *кратности* концов этой *ассоциации* установить значение 1.

23. Добавить направленную *ассоциацию* от класса Контроллер Банкомата к классу Устройство выдачи наличных. В качестве *кратности* концов этой *ассоциации* установить значение 1.

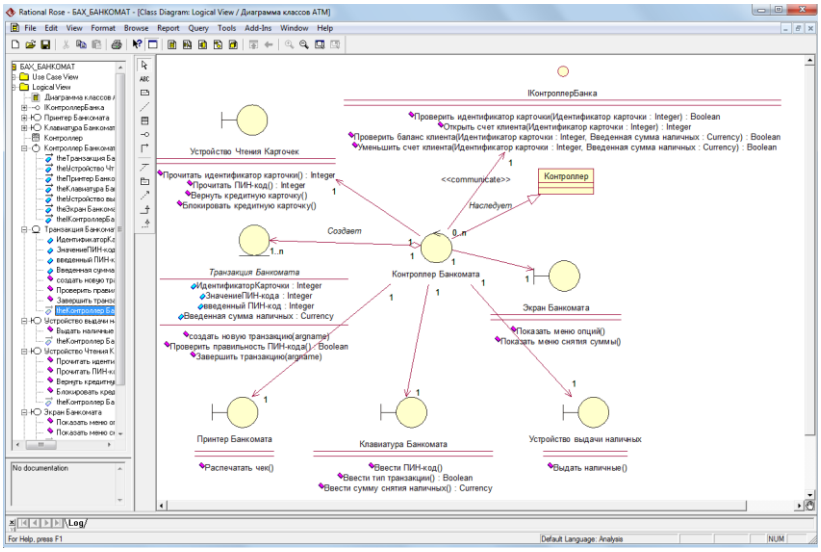


Рисунок 33 – Окончательный вид диаграммы классов для разрабатываемой модели банкомата

24. Добавить направленную *ассоциацию* от класса Контроллер Банкомата к классу Экран Банкомата. В качестве *кратности* концов этой *ассоциации* установить значение 1.

25. Добавить направленную *ассоциацию* от класса Контроллер Банкомата к классу IКонтроллер Банка. В качестве *кратности* конца этой *ассоциации* для первого класса установить значение 0..n, а *кратности* конца *ассоциации* для второго класса установить значение 1. В качестве стереотипа данной *ассоциации* выбрать из вложенного списка значение <<communicate>>. Применение данного стереотипа означает, что между этими классами должна существовать физическая взаимосвязь (рис. 3.7).

Следует заметить, что при изображении диаграммы классов все классы представлены в форме графических стереотипов, при этом выбран способ отображения сигнатуры операций классов. Для более компактного представления диаграммы можно убрать отображение атрибутов, операций или сигнатуры операций отдельных классов с помощью соответствующих операций контекстного меню **Options** (Настройка).

Задание 4. Разработка диаграммы кооперации и редактирование свойств ее элементов

Диаграмма кооперации является разновидностью диаграммы взаимодействия, и в контексте языка UML описывает динамический аспект взаимодействия объектов при реализации отдельных вариантов использования. Активизировать рабочее окно диаграммы кооперации в программе IBM Rational Rose 2003 можно несколькими способами:

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

При этом появляется новое окно с чистым рабочим листом диаграммы кооперации и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы кооперации (табл. 15). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 15 – Назначение кнопок специальной панели инструментов диаграммы кооперации

Графическое изображение	**Всплывающая подсказка**	**Назначение кнопки**
	Selection Tool	Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме
	Text Box	Добавляет на диаграмму текстовую область
	Note	Добавляет на диаграмму примечание
	Anchor Note to Item	Добавляет на диаграмму *связь* примечания с соответствующим графическим элементом диаграммы
	Object	Добавляет на диаграмму *объект*
	Class Instance	Добавляет на диаграмму экземпляр класса
	Object Link	Добавляет на диаграмму *связь*
	Link To Self	Добавляет на диаграмму рефлексивную *связь*
	Link Message	Добавляет на *связь* диаграммы прямое *сообщение*
	Reverse Link Message	Добавляет на *связь* диаграммы обратное *сообщение*
	Data Token	Добавляет на *связь* диаграммы элемент прямого потока данных
	Reverse Data Token	Добавляет на *связь* диаграммы элемент обратного потока данных

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

кооперации одним из описанных выше способов следует в качестве имени данной диаграммы задать: Снятие наличных по кредитной карточке.

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

Добавление объекта на диаграмму кооперации и редактирование его свойств

Добавить *объект* на диаграмму кооперации можно стандартным образом с помощью соответствующей кнопки на специальной панели инструментов. Однако, в случае наличия построенной ранее диаграммы классов, более удобным представляется следующий способ. В браузере проекта выделить необходимый класс и, удерживая нажатой левую кнопку мыши, перетащить изображение пиктограммы класса из браузера на свободное место рабочего листа диаграммы кооперации. В результате этих действий на диаграмме кооперации появится изображение *объекта* с именем класса и маркерами изменения его геометрических размеров (рис. 34).

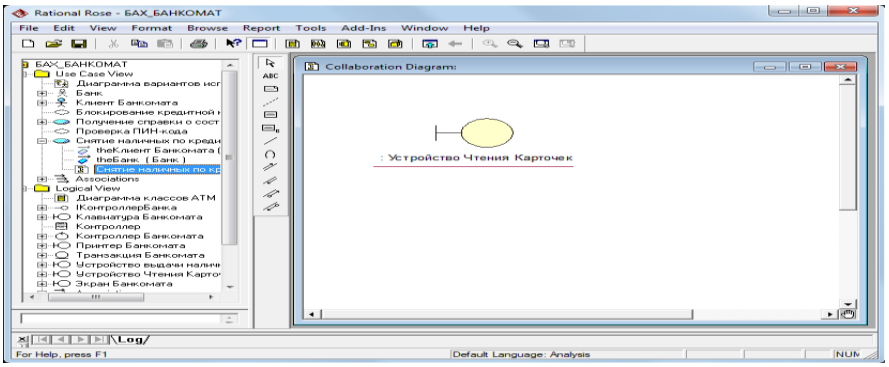


Рисунок 34 – Диаграмма кооперации после добавления на нее анонимного объекта класса Устройство чтения карточки

По умолчанию каждый добавляемый *объект* считается анонимным. При необходимости можно задать собственное имя *объекта*, для чего двойным щелчком на изображении *объекта* на диаграмме кооперации следует вызвать диалоговое окно свойств этого *объекта* (рис. 35).

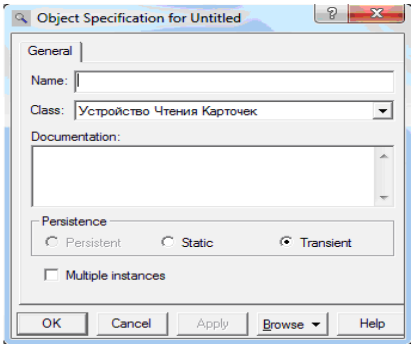


Рисунок 35 – Диалоговое окно спецификации свойств объекта класса Устройство чтения карточки

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

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

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

Добавление связи и редактирование ее свойств

Поместить на диаграмму, способом перетаскивания пиктограммы, актера из браузера проекта и добавить *связь*, соединяющую *объект* класса Клиент Банкомата (актера) с объектом класса Устройство чтения карточки (рис. 36).

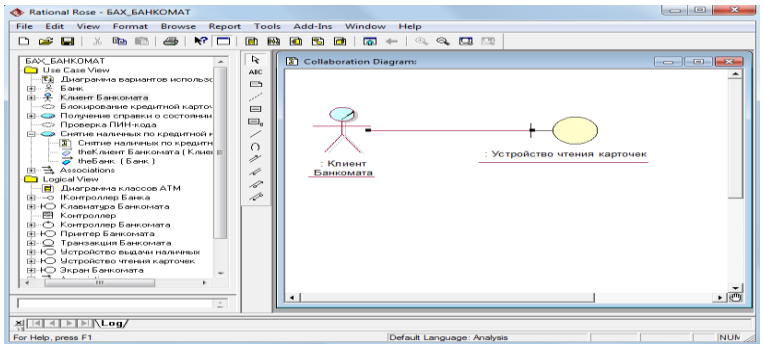


Рисунок 36 – Диаграмма кооперации после добавления связи между объектом класса Клиент Банкомата (актером) и объектом класса Устройство чтения карточки

По умолчанию каждая добавляемая *связь* считается анонимной. При необходимости можно задать имя *связи* с помощью диалогового окна спецификации свойств данной *связи* (рис. 37).

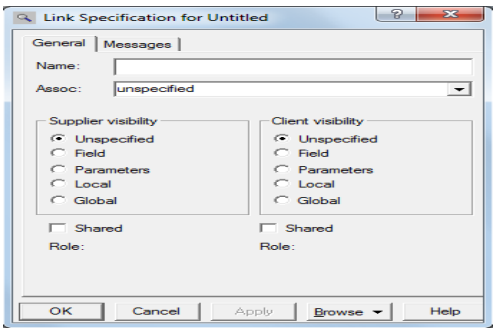


Рисунок 37 – Диалоговое окно редактирования свойств связи

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

Добавление сообщения и редактирование его свойств

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

Однако более удобным представляется способ добавления *сообщений* с помощью диалогового окна свойств *связей*. Для этого двойным щелчком на линии *связи* вызывается окно ее свойств и раскрывается вкладка **Messages** (*сообщения*). После этого следует выполнить операцию контекстного меню **Insert To** (Вставить в направлении), в результате чего появляется вложенный список с предложением выбрать одну из операций целевого класса для спецификации имени *сообщения* (рис. 38).

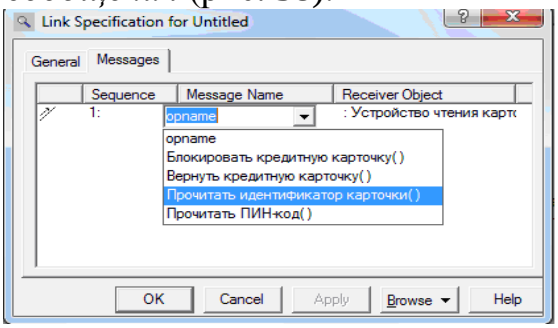


Рисунок 38 – Диалоговое окно добавления сообщения для выбранной связи

Для рассматриваемой модели банкомата для первого *сообщения* следует выбрать операцию прочитать идентификатор карточки(). После выбора операции для данного *сообщения* оно добавляется в список *сообщений* данной *связи*, а рядом с линией *связи* на диаграмме кооперации появится стрелка с номером и именем этого *сообщения* (рис.39).

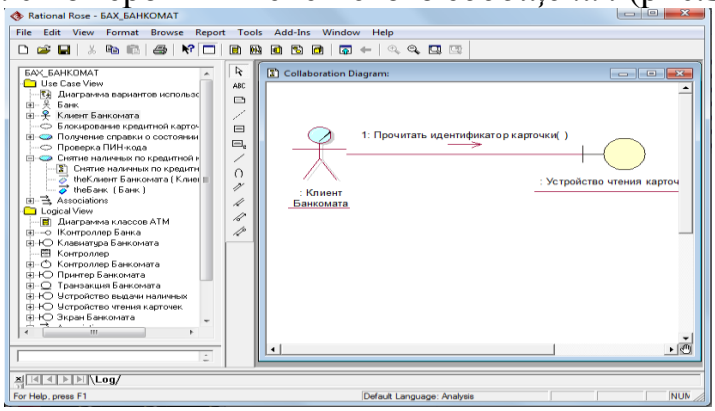


Рисунок 39 – Диаграмма кооперации после добавления связи между объектом класса Клиент Банкомата (актером) и объектом класса Устройство чтения карточки

Кроме имени *сообщения* можно также задать *стереотип* синхронизации и частоту передачи. Для этой цели следует воспользоваться диалоговым окном спецификации свойств *сообщений* (рис. 40), которое можно открыть двойным щелчком на имени *сообщения* в списке рассматриваемой вкладки **Messages** окна спецификации свойств *связи*.

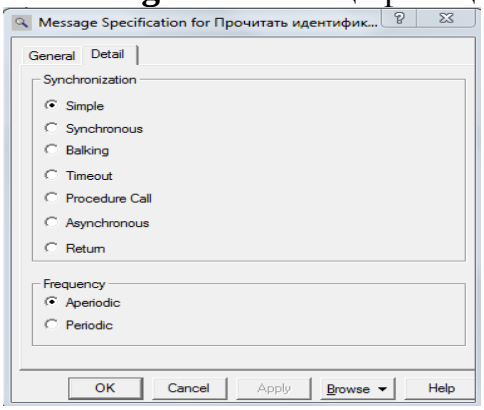


Рисунок 40 – Диалоговое окно спецификации свойств сообщения

Группа свойств **Synchronization** (Синхронизация) предназначена для определения способа синхронизации передаваемого *сообщения*. При изменении этого *свойства* изменяется графическое изображение стрелки соответствующего *сообщения*. Характеристика отдельных свойств синхронизации *сообщений* и графическое изображение соответствующих стрелок *сообщений* приводится в следующей таблице (табл. 16).

Таблица 16 – Характеристика свойств синхронизации сообщений

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

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

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

Окончательное построение диаграммы кооперации для модели банкомата

Для завершения построения диаграммы кооперации добавить оставшиеся *объекты*, *связи* и *сообщения*. Для этого:

1. Добавить *объекты* классов с именами: Контроллер Банкомата, Транзакция Банкомата, Клавиатура Банкомата, Экран Банкомата, Принтер Банкомата, Устройство выдачи наличных и ІКонтроллерБанка.

2. Добавить *связи*, соединяющие *объекты* классов с именами: Контроллер Банкомата с Устройством чтения карточки, Контроллер Банкомата с Транзакцией Банкомата, Контроллер Банкомата с Клавиатурой Банкомата, Контроллер Банкомата с Экраном Банкомата, Контроллер Банкомата с Принтером Банкомата, Контроллер Банкомата с Устройством выдачи наличных и Контроллер Банкомата с ІКонтроллерБанка.

3. Добавить *сообщение*: проверить идентификатор карточки (Integer) , направленное от *объекта* класса Контроллер Банкомата к *объекту* класса ІКонтроллерБанка.

4. Добавить *сообщение*: ввести ПИН-код(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.

5. Добавить *сообщение*: прочитать ПИН-код(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство чтения карточки.

6. Добавить *сообщение*: создать новую транзакцию(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Транзакция Банкомата.

7. Добавить *сообщение*: проверить правильность ПИН-кода(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Транзакция Банкомата.

8. Добавить *сообщение*: показать меню опций(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Экран Банкомата.

9. Добавить *сообщение*: ввести тип транзакции(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.

10. Добавить *сообщение*: показать меню снятия суммы(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Экран Банкомата.

11. Добавить *сообщение*: ввести сумму снятия наличных(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.

12. Последовательно добавить 3 *сообщения*: открыть счет клиента (Integer) , проверить баланс клиента (Integer, Currency) и уменьшить счет клиента(Integer, Currency), направленные от *объекта* класса Контроллер Банкомата к *объекту* класса ІКонтроллерБанка.

13. Добавить *сообщение*: распечатать чек(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Принтер Банкомата.

14. Добавить *сообщение*: вернуть кредитную карточку(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство чтения карточки.

15. Добавить *сообщение*: выдать наличные(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство выдачи наличных.

16. Добавить *сообщение*: завершить транзакцию(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Транзакция Банкомата.

Диаграмма кооперации, описывающая реализацию типичного хода событий варианта использования Снятие наличных по кредитной карточке для проекта системы управления банкоматом, показана на рис.41.

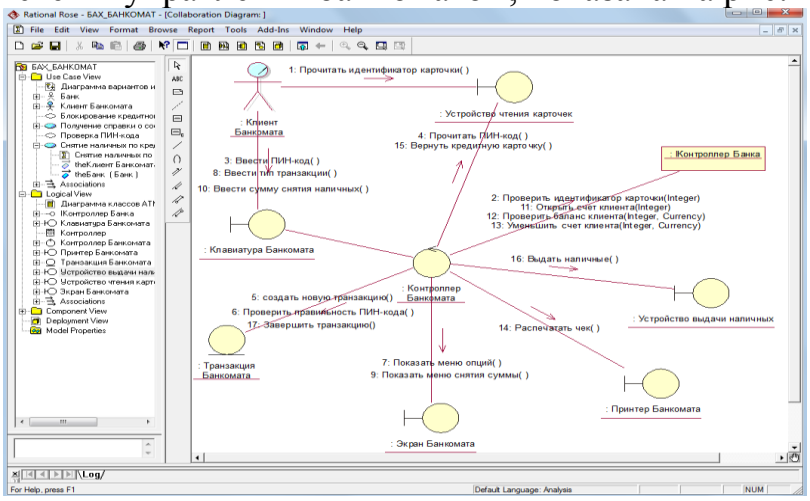


Рисунок 41 – Окончательный вариант диаграммы кооперации

При необходимости можно изменить порядок следования *сообщений* и их спецификацию, а также установить дополнительную синхронизацию *сообщений* и связать с *сообщениями* примечания.

Задание 5. Разработка диаграммы последовательности и редактирование свойств ее элементов

Диаграмма последовательности является другой формой визуализации взаимодействия в модели и, как и диаграмма кооперации, оперирует *объектами* и *сообщениями*. Особенность работы в среде IBM Rational Rose 2003 заключается в том, что этот вид канонической диаграммы может быть создан автоматически после построения диаграммы кооперации и нажатия клавиши <F5>. С помощью этой же клавиши осуществляется переключение между диаграммами последовательности и кооперации в модели.

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

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

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

При этом появляется новое окно с чистым рабочим листом диаграммы классов и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы последовательности (табл. 17). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 17 – Назначение кнопок специальной панели инструментов диаграммы последовательности

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

На специальной панели инструментов по умолчанию присутствует практически все пиктограммы элементов, которые могут быть использованы для построения диаграммы последовательности. Из дополнительных пиктограмм графических элементов на специальную панель инструментов можно добавить лишь *сообщение* типа вызова процедуры и асинхронное *сообщение* (последняя строка табл. 5.1). Относительно изображения асинхронного *сообщения* в форме полустрелки следует заметить, что хотя в версии языка UML 1.5 этот элемент отсутствует, в среде IBM Rational Rose

2003 возможно изобразить этот тип *сообщений* в форме специального графического *стереотипа*.

Добавление объекта на диаграмму последовательности и редактирование его свойств

Добавить *объект* на диаграмму последовательности можно как стандартным образом с помощью соответствующей кнопки на специальной панели инструментов, так и более удобным способом – с помощью перетаскивания изображения пиктограммы класса из браузера на свободное место рабочего листа диаграммы последовательности.

В результате этих действий на диаграмме последовательности появится изображение *объекта* с именем класса, маркерами изменения его геометрических размеров и вертикальной пунктирной линией, означающей *линию жизни* этого *объекта* (рис. 42).

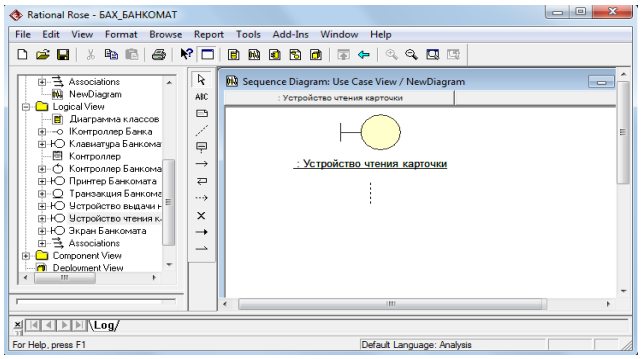


Рисунок 42 – Диаграмма последовательности после добавления анонимного объекта класса Устройство чтения карточки

Так же как и для диаграммы кооперации, для диаграммы последовательности каждый добавляемый *объект* по умолчанию считается анонимным. При необходимости можно задать собственное имя *объекта*, для чего уже известным способом (например, двойным щелчком на изображении *объекта* на диаграмме) следует вызвать диалоговое окно свойств *объекта*, которое аналогично *объектам* диаграммы кооперации (рис. 5.2).

Добавление сообщения на диаграмму последовательности и редактирование его свойств

Для добавления *сообщения* между предварительно размещенными на диаграмме *объектами* нужно с помощью левой кнопки мыши нажать кнопку с изображением *сообщения* на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении *линии жизни* одного *объекта* на диаграмме и отпустить ее на изображении *линии жизни* второго *объекта*.

В результате этих действий на диаграмме появится изображение *сообщения*, передаваемого, например, от экземпляра актера Клиент Банкомата *объекту* класса Устройство чтения карточки. Поскольку кнопка с изображением актера отсутствует на специальной панели инструментов

диаграммы последовательности, соответствующий объект следует предварительно поместить на диаграмму способом перетаскивания пиктограммы актера из браузера проекта. При этом изображение *линии жизни* у соответствующей пары *объектов* изменится на изображение *фокуса управления* (рис. 43).

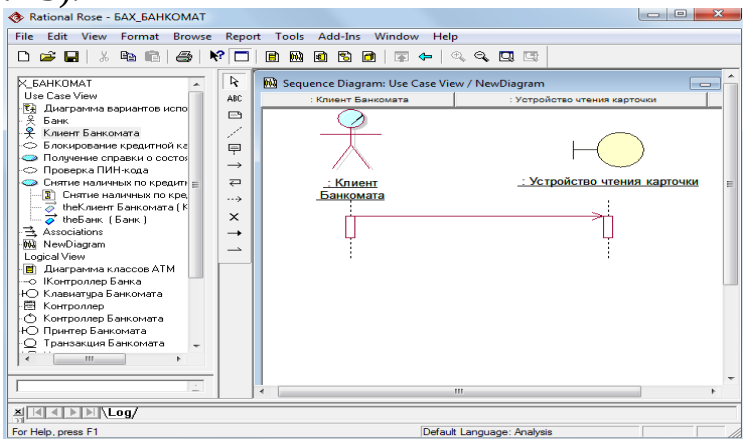


Рисунок 43 – Диаграмма последовательности после добавления сообщения от экземпляра актера Клиент Банкомата к объекту класса Устройства чтения карточки

Для спецификации *свойств* добавленного *сообщения* предназначено специальное окно, которое можно открыть двойным щелчком на изображении *сообщения* на диаграмме последовательности. Имя *сообщения* можно выбрать на вкладке **General** (Общие) из выпадающего списка операций соответствующего класса-приемника (рис. 44).

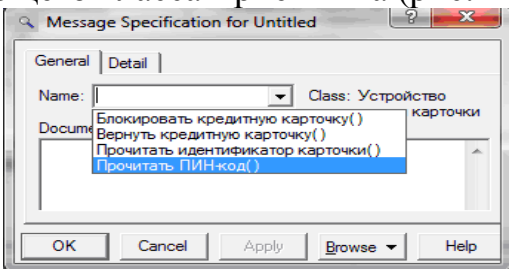


Рисунок 44 – Диалоговое окно спецификации свойств сообщения

Имя *сообщения* можно выбрать также из контекстного меню *сообщения*, в котором перечислены все операции класса-приемника данного *сообщения* (рис. 45). При необходимости в контекстном меню можно задать новую операцию, в этом случае следует выбрать строку **<new operation>**. При этом откроется диалоговое окно спецификации свойств новой операции класса-приемника, особенности редактирования которых были рассмотрены ранее.

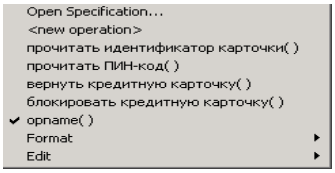


Рисунок 45 – Контекстное меню сообщения на диаграмме последовательности

Для модели банкомата в качестве имени первого *сообщения* следует выбрать операцию прочитать идентификатор карточки(). После выбора операции для данного *сообщения* следует нажать кнопку **Apply** или **OK**, в результате чего имя *сообщения* будет изображено на диаграмме последовательности рядом с линией *сообщения* (рис. 46).

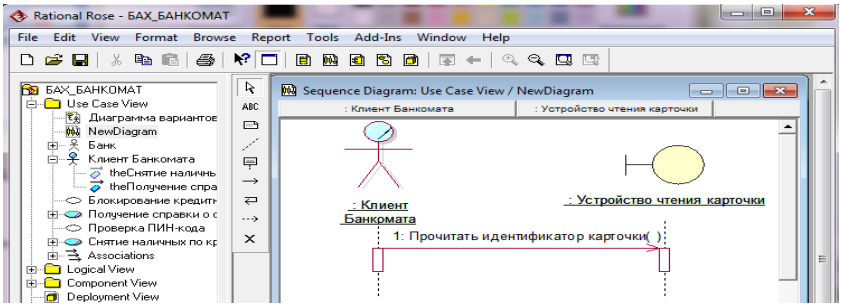


Рисунок 46 – Диаграмма последовательности после добавления сообщения от экземпляра актера Клиент Банкомата к объекту класса Устройство чтения карточки

Построение диаграммы последовательности сводится к добавлению и редактированию свойств отдельных *объектов* и *сообщений*. Доступ к окну спецификации свойств соответствующих элементов возможен также либо через контекстное меню, либо с помощью операции главного меню **Browse→ Specification** (Обзор→Спецификация). При добавлении *сообщений* на диаграмму последовательности они получают по умолчанию свой номер в общей последовательности *сообщений*.

Следует заметить, что по умолчанию нумерация *сообщений* на диаграмме последовательности может быть отключена. При необходимости показать номера *сообщений* следует выполнить операцию главного меню: **Tools→Options** (Инструменты→Параметры), открыть вкладку Diagram (Диаграмма) и выставить отметку выбора строки **Sequence numbering** (Нумерация *сообщений* на диаграмме последовательности) в группе свойств **Display** (рис. 47).

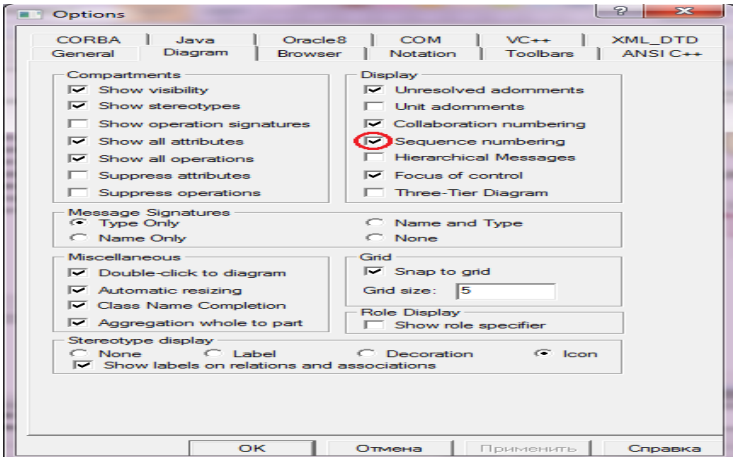


Рисунок 47 – Диалоговое окно спецификации свойств модели

Это же окно спецификации свойств модели можно открыть с помощью операции главного меню: **View→Toolbars→Configure** (Вид→Панели инструментов→Настроить),

Для детальной спецификации *свойств сообщений* на диаграмме последовательности можно использовать также группу *свойств* **Synchronization** (Синхронизация) и **Frequency** (Частота), доступные для выбора на вкладке **Detail** (Подробно) окна спецификации *сообщения*. При изменении способа синхронизации передаваемого *сообщения* изменяется графическое изображение стрелки соответствующего *сообщения*.

Окончательное построение диаграммы последовательности модели банкомата

Для завершения построения диаграммы последовательности рассматриваемого примера следует выполнить следующие действия:

1. Добавить *объекты* классов с именами: Контроллер Банкомата, Транзакция Банкомата, Клавиатура Банкомата, Экран Банкомата, Принтер Банкомата, Устройство выдачи наличных, Устройство чтения карточки и ІКонтроллерБанка.
2. Добавить *сообщение*: проверить идентификатор карточки (Integer), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса ІКонтроллерБанка.
3. Добавить *сообщение*: ввести ПИН-код(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.
4. Добавить *сообщение*: прочитать ПИН-код(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство чтения карточки.
5. Добавить *сообщение*: создать новую транзакцию(), направленное от *объекта* класса Контроллер Банкомата к изображению *объекта* класса Транзакция Банкомата. При этом изображение *объекта* класса Транзакция Банкомата следует переместить вниз на уровень этого *сообщения*, что будет визуально означать создание данного *объекта* в более поздний момент времени, чем начало функционирования моделируемой программной системы.
6. Добавить *сообщение*: проверить правильность ПИН-кода(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Транзакция Банкомата.
7. Добавить *сообщение*: показать меню опций(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Экран Банкомата.
8. Добавить *сообщение*: ввести тип транзакции(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.
9. Добавить *сообщение*: показать меню снятия суммы(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Экран Банкомата.
10. Добавить *сообщение*: ввести сумму снятия наличных(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.

11. Последовательно добавить 3 *сообщения*: открыть счет клиента (Integer),

12. проверить баланс клиента (Integer, Currency)

13. уменьшить счет клиента (Integer, Currency), направленные от *объекта* класса Контроллер Банкомата к *объекту* класса ІКонтроллерБанка.

14. Добавить *сообщение*: распечатать чек(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Принтер Банкомата.

15. Добавить *сообщение*: вернуть кредитную карточку(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство чтения карточки.

16. Добавить *сообщение*: выдать наличные(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство выдачи наличных.

17. Добавить *сообщение*: завершить транзакцию(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Транзакция Банкомата.

18. После добавления *сообщения* завершить транзакцию() поместить на *линию жизни объекта* класса Транзакция Банкомата символ уничтожения этого *объекта*.

Фрагмент диаграммы последовательности, описывающая реализацию типичного хода событий варианта использования Снятие наличных по кредитной карточке для проекта системы управления банкоматом, показан на (рис. 48).

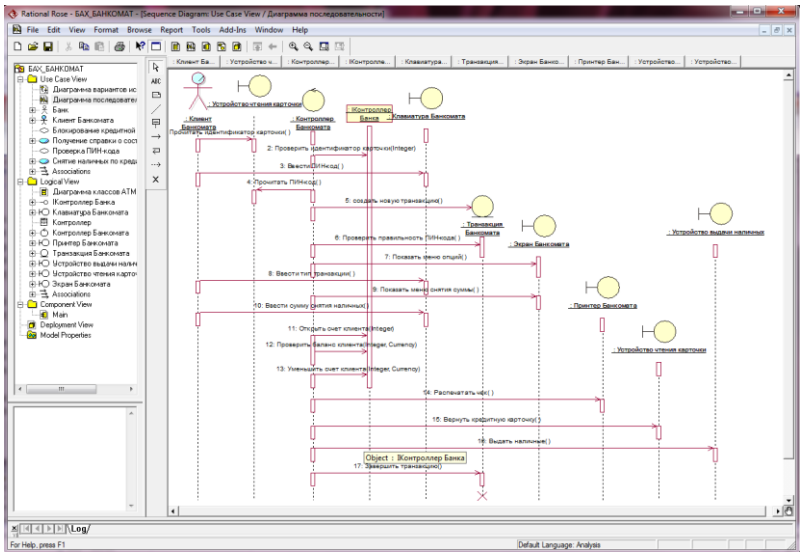


Рисунок 48 – Окончательный вид диаграммы последовательности,

Если необходимо изменить порядок следования *сообщений*, то из двух диаграмм взаимодействия данное действие удобнее выполнить на диаграмме последовательности, чем на диаграмме кооперации.

Задание 6. Разработка диаграммы состояний и редактирование свойств ее элементов

Переходя к рассмотрению диаграммы *состояний*, следует отметить, что в среде IBM Rational Rose 2003 этот тип диаграмм может относиться к отдельному классу, операции класса, варианту использования, пакету или представлению. Для того чтобы построить диаграмму *состояний*, ее вначале необходимо создать и активизировать.

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

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

Таблица 18 – Назначение кнопок специальной панели инструментов диаграммы состояний

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Selection Tool	Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме
	Text Box	Добавляет на диаграмму текстовую область
	Note	Добавляет на диаграмму примечание
	Anchor Note to Item	Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы
	State	Добавляет на диаграмму *состояние*
	Start State	Добавляет на диаграмму начальное *состояние*

	End State	Добавляет на диаграмму конечное *состояние*
	State Transition	Добавляет на диаграмму *переход*
	Transition to Self	Добавляет на диаграмму рефлексивный *переход*
	Horizontal Synchronization	Добавляет на диаграмму горизонтально расположенный символ синхронизации (по умолчанию отсутствует)
	Vertical Synchronization	Добавляет на диаграмму вертикально расположенный символ синхронизации (по умолчанию отсутствует)
	Decision	Добавляет на диаграмму символ принятия решения для альтернативных *переходов* (по умолчанию отсутствует)

По умолчанию на специальной панели инструментов могут отсутствовать кнопки с тремя последними графическими элементами из таблицы 6.1. Продолжая разработку проекта по моделированию системы управления банкоматом, можно приступить к разработке новой диаграммы *состояний*. С этой целью для диаграммы *состояний* модели банкомата зададим имя Диаграмма *состояний*, а в секцию ее документации введем текст «Диаграмма *состояний* описывает конечный автомат банкомата».

Добавление состояния на диаграмму состояний и редактирование его свойств

Для добавления *состояния* на диаграмму *состояний* необходимо с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *состояния* на специальной панели инструментов и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы.

Для диаграммы *состояний* модели банкомата в качестве имени первого добавленного *состояния* зададим имя Ожидание карточки (рис.49).

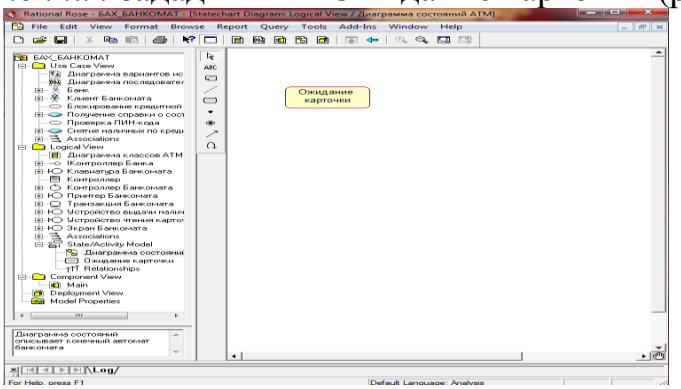


Рисунок 49 – Диаграмма состояний после добавления на нее состояния Ожидание карточки

Для добавленного *состояния* можно открыть диалоговое окно его свойств двойным щелчком мыши. В этом случае активизируется диалоговое окно со специальными вкладками, в поля которых можно занести всю информацию по данному *состоянию* (рис. 50).

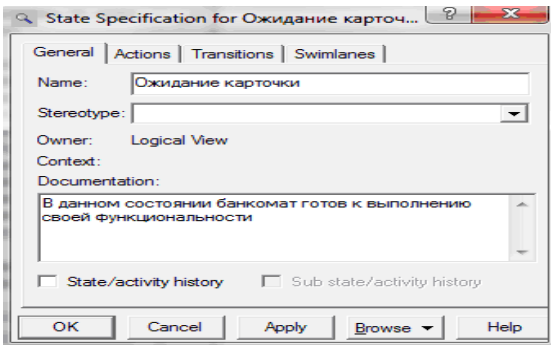


Рисунок 50 – Диалоговое окно спецификации свойств состояния

При необходимости в диалоговом окне спецификации свойств выбранного *состояния* можно задать вложенное *историческое состояние*. Для этого следует выставить отметку у свойства **State/activity history** (*Историческое состояние*/деятельность) и нажать кнопку **Apply**. В результате внутри исходного *состояния* появится вложенное *историческое состояние* (рис. 51, а).

Рисунок 51 – Добавление вложенного исторического состояния (а) и состояния глубокой истории (б) для состояния Ожидание карточки

Чтобы обычное *историческое состояние* превратить в *состояние* глубокой истории, следует дополнительно выставить отметку у свойства **Sub state/activity history** (Историческое под-состояние/деятельность), которое становится доступным для редактирования после выбора первого свойства, и нажать кнопку **Apply**. В результате внутри исходного *состояния* появится вложенное *состояние* глубокой истории (рис. 6.3, б).

Чтобы обычное *состояние* превратить в *композит*, следует при добавлении нового *состояния* поместить его внутри границы того *состояния*, которое необходимо сделать *композитным*. В результате внутри исходного *состояния* появится новое вложенное *состояние* с именем **NewState**, которое при перемещении композита в области диаграммы *состояний* всегда будет находиться внутри своего композита (рис. 52).

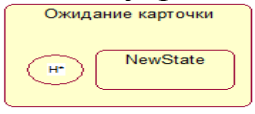


Рисунок 52 – Превращение состояния Ожидание карточки в композитное состояние

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

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

Добавление перехода и редактирование его свойств

Для добавления *перехода* между двумя *состояниями* нужно с помощью левой кнопки мыши нажать кнопку с изображением *перехода* на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходного состояния на диаграмме и отпустить ее на изображении целевого состояния. В результате этих действий на диаграмме появится изображение *перехода*, соединяющего два выбранных состояния. Продолжая разработку модели системы управления банкоматом, добавим на диаграмму *состояний* начальное *состояние* (**Start State**) и соединим его *переходом* с *состоянием* Ожидание карточки (рис. 53).

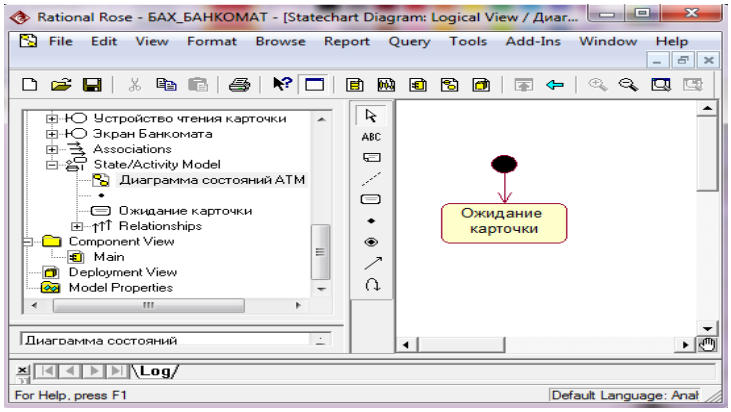


Рисунок 53 – Диаграмма состояний после добавления на нее перехода из начального состояния в состояние Ожидание карточки

После добавления *перехода* на диаграмму *состояний* можно открыть диалоговое окно его свойств и специфицировать дополнительные свойства, доступные на соответствующих вкладках (рис. 54). Следует обратить внимание на две первые строки вкладки **Detail** (Подробно), которые представляются наиболее важными из свойств *перехода*. Первое поле ввода **Guard Condition** служит для задания ***сторожевого условия***, которое определяет правило срабатывания соответствующего *перехода*. Во втором поле ввода **Action** можно специфицировать *действие, которое происходит при срабатывании перехода* до того, как моделируемая система попадет в целевое *состояние*.

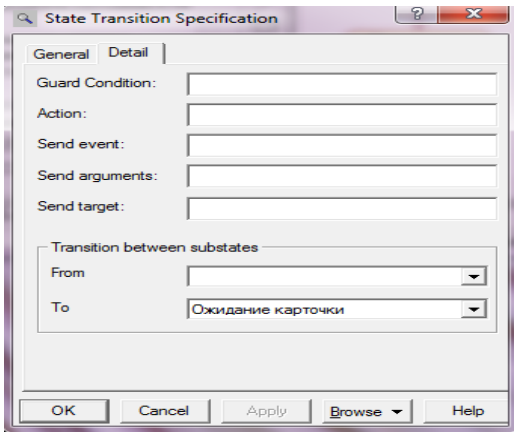


Рисунок 54 – Диалоговое окно спецификации свойств перехода, открытое на вкладке Detail (Подробно)

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

Окончательное построение диаграммы состояний модели банкомата

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

1. Добавить *состояния* с именами: Ожидание ввода ПИН-кода, Проверка ПИН-кода, Ожидание выбора клиента, Обработка запроса на снятие наличных, Обработка запроса на получение справки, Выдача наличных, Печать, Возврат карточки, Завершение транзакции и финальное *состояние*.
2. Добавить *переход*: карточка вставлена, направленный от *состояния* Ожидание карточки к *состоянию* Ожидание ввода ПИН-кода.
3. Добавить *переход*: ПИН-код введен, направленный от *состояния* Ожидание ввода ПИН-кода к *состоянию* Проверка ПИН-кода.
4. Добавить *переход*: отмена транзакции, направленный от *состояния* Ожидание ввода ПИН-кода к *состоянию* Возврат карточки.
5. Добавить *переход* со *сторожевым условием*: [ПИН-код верный], направленный от *состояния* Проверка ПИН-кода к *состоянию* Ожидание выбора клиента.
6. Добавить *переход* со *сторожевым условием*: [ПИН-код неверный], направленный от *состояния* Проверка ПИН-кода к *состоянию* Ожидание ввода ПИН-кода.
7. Добавить *переход*: три неудачи с *действием на переходе* конфискация карточки, направленный от *состояния* Проверка ПИН-кода к *состоянию* Завершение транзакции. Для задания *действия на данном переходе* следует ввести текст конфискация карточки в поле ввода **Action** (Действие) на вкладке **Detail** (Подробно) окна спецификации свойств данного *перехода* (рис. 55).

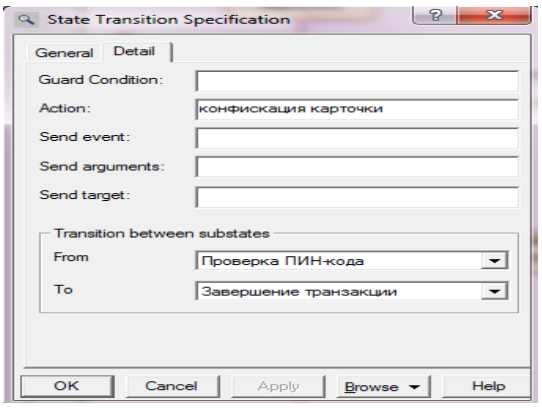


Рисунок 55 – Диалоговое окно спецификации свойств перехода три неудачи при задании действия на переходе

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

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

Диаграмма *состояний* для рассматриваемой модели банкомата будет иметь следующий вид (рис. 56).

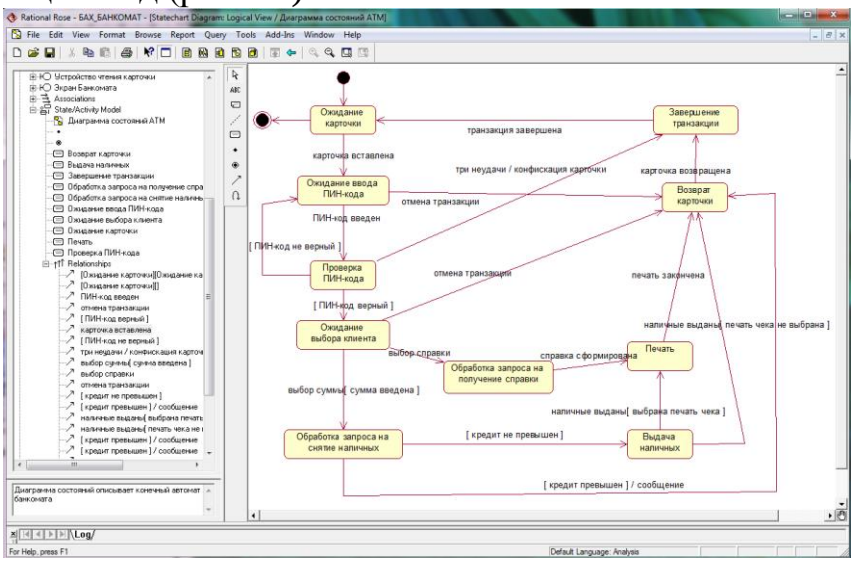


Рисунок 56 – Окончательный вид диаграммы состояний

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

Задание 7. Разработка диаграммы деятельности и редактирование свойств ее элементов

Диаграмма *деятельности* в среде IBM Rational Rose 2003, так же как и диаграмма состояний, может относиться к отдельному классу, операции класса, варианту использования, пакету или представлению. Для того чтобы построить диаграмму *деятельности*, ее вначале необходимо создать и активизировать.

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

- Щелкнуть на кнопке с изображением диаграммы состояний на стандартной панели инструментов, после чего следует выбрать

представление и тип разрабатываемой диаграммы – диаграмма *деятельности*.

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

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы *деятельности* и специальная панель инструментов, содержащая кнопки с изображением графических элементов, необходимых для разработки диаграммы *деятельности* (табл. 19). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 19 – Назначение кнопок специальной панели инструментов диаграммы деятельности

Графическое изображение	**Всплывающая подсказка**	**Назначение кнопки**
	Selection Tool	Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме
	Text Box	Добавляет на диаграмму текстовую область
	Note	Добавляет на диаграмму примечание
	Anchor Note to Item	Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы
	State	Добавляет на диаграмму состояние
	Activity	Добавляет на диаграмму *деятельность*
	Start State	Добавляет на диаграмму начальное состояние
	End State	Добавляет на диаграмму конечное состояние
	State Transition	Добавляет на диаграмму *переход*

	Transition to Self	Добавляет на диаграмму рефлексивный *переход*
	Horizontal Synchronization	Добавляет на диаграмму горизонтально расположенный символ синхронизации
	Vertical Synchronization	Добавляет на диаграмму вертикально расположенный символ синхронизации
	Decision	Добавляет на диаграмму *символ принятия решения* для альтернативных *переходов*
	Swimlane	Добавляет на диаграмму дорожку
	Object	Добавляет на диаграмму объект (по умолчанию отсутствует)
	Object Flow	Добавляет на диаграмму стрелку потока объектов (по умолчанию отсутствует)
	Business Activity	Добавляет на диаграмму бизнес-деятельность (по умолчанию отсутствует)
	Business Transaction	Добавляет на диаграмму бизнес-транзакцию (по умолчанию отсутствует)

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

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

Добавление деятельности на диаграмму деятельности и редактирование ее свойств

Для добавления *деятельности* на диаграмму *деятельности* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *деятельности* на специальной панели инструментов и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. Добавить *деятельность* на диаграмму можно также с помощью операции главного меню: **Tools→Create→Activity** или с помощью операции контекстного меню:

New→Activity, предварительно выделив диаграмму *деятельности* в браузере проекта.

В результате этих действий на диаграмме появится изображение *деятельности* с именем **NewActivity**, предложенное программой по умолчанию. Начиная построение диаграммы *деятельности* модели банкомата, для первой добавленной *деятельности* зададим имя Вставить карточку (рис. 57).

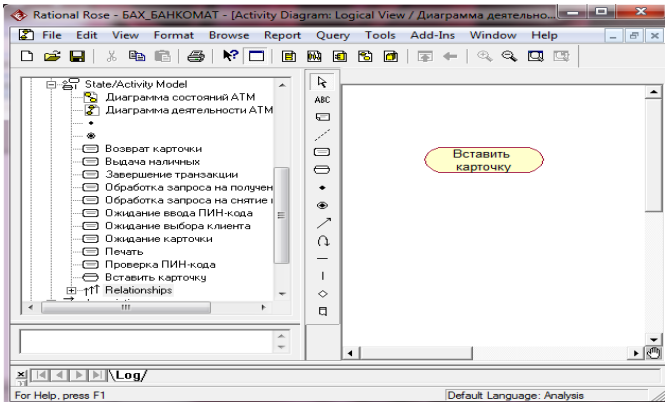


Рисунок 57 – Диаграмма деятельности после добавления на нее деятельности Вставить карточку

После добавления *деятельности* на диаграмму *деятельности* можно открыть диалоговое окно спецификации ее свойств и определить дополнительные свойства *деятельности*, доступные на соответствующих вкладках (рис. 58).

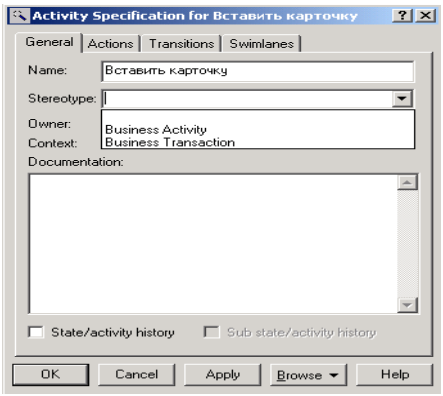


Рисунок 58 – Диалоговое окно спецификации свойств деятельности

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

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

Добавление перехода и редактирование его свойств

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

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

После добавления *перехода* на диаграмму *деятельности* становятся доступными для редактирования его свойства в специальном диалоговом окне (рис. 59), которое можно открыть по двойному щелчку левой кнопкой мыши на изображении *перехода*.

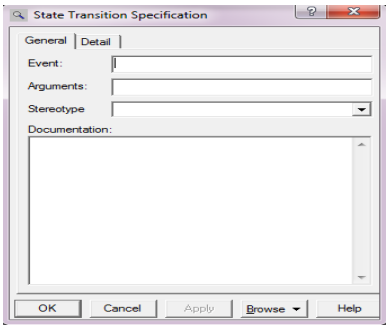


Рисунок 59 – Диалоговое окно спецификации свойств перехода

При спецификации свойств *переходов* следует помнить, что все *переходы* на диаграмме *деятельности* является нетриггерными, т.е. не имеют

имен событий. По этой причине поле ввода с именем **Event** (Событие) для всех *переходов* должно оставаться пустым. Но все *переходы*, выходящие из *символов ветвления (решения)*, должны иметь *сторожевые условия*, которые специфицируются на вкладке **Detail** (Подробно) диалогового окна спецификации свойств *перехода*.

Окончательное построение диаграммы деятельности модели банкомата

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

1. Добавить *деятельности* с именами: Ввести ПИН-код, Выбрать тип транзакции, Ввести сумму, Получить справку о состоянии счета, Получить наличные, Получить чек, Получить карточку и финальное состояние.
2. Добавить *символы ветвления (решения)*, расположив их между *деятельностями* с именами: Ввести ПИН-код и Выбрать тип транзакции, Выбрать тип транзакции и Ввести сумму, Ввести сумму и Получить наличные, Получить наличные и Получить чек, Получить чек и Получить карточку. При этом последний *символ решения* будет использоваться в качестве символа соединения.
3. Добавить *переход*, направленный от *деятельности* Ввести ПИН-код к *символу решения*.
4. Добавить *переход* со *сторожевым условием*: [ПИН-код верный], направленный от *символа решения* к *деятельности* Выбрать тип транзакции.
5. Добавить *переход* со *сторожевым условием*: [ПИН-код неверный], направленный от *символа решения* к символу соединения.
6. Добавить *переход*, направленный от *деятельности* Выбрать тип транзакции к *символу решения*.
7. Добавить *переход* со *сторожевым условием*: [выбор снятия суммы] , направленный от *символа решения* к *деятельности* Ввести сумму.
8. Добавить *переход* со *сторожевым условием*: [выбор получения справки], направленный от *символа решения* к *деятельности* Получить справку о состоянии счета.
9. Добавить *переход*, направленный от *деятельности* Ввести сумму к *символу решения*.
10. Добавить *переход* со *сторожевым условием*: [сумма не превышает кредит], направленный от *символа решения* к *деятельности* Получить наличные.
11. Добавить *переход* со *сторожевым условием*: [сумма превышает кредит], направленный от *символа решения* к символу соединения.
12. Добавить *переход*, направленный от *деятельности* Получить наличные к *символу решения*.
13. Добавить *переход* со *сторожевым условием*: [выбрана печать чека], направленный от *символа решения* к *деятельности* Получить чек.

14. Добавить *переход* со *сторожевым условием*: [печать чека не выбрана], направленный от *символа решения* к символу соединения.

15. Добавить *переход*, направленный от *деятельности* Получить чек к символу соединения.

16. Добавить *переход*, направленный от *деятельности* Получить справку о состоянии счета к символу соединения.

17. Добавить *переход*, направленный от символа соединения к *деятельности* Получить карточку.

18. Добавить *переход*, направленный от *деятельности* Получить карточку к финальному состоянию.

Построенная таким образом диаграмма *деятельности* будет иметь следующий вид (рис. 60).

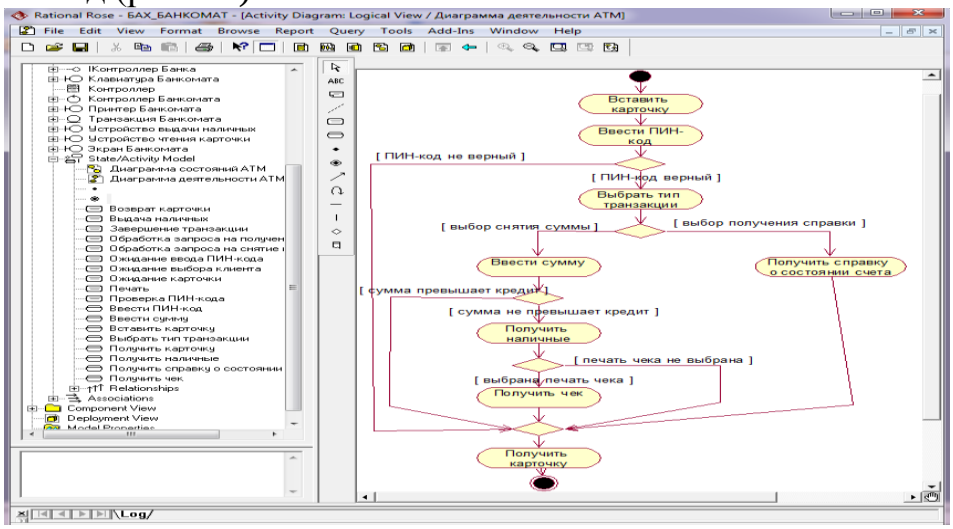


Рисунок 60 – Окончательный вид диаграммы деятельности

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

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

Задание 8. Разработка диаграммы деятельности для моделирования бизнес-процессов

Продолжая рассмотрение особенностей разработки диаграмм деятельности, следует отметить, что программа IBM Rational Rose 2003 может быть успешно использована для выполнения проектов по моделированию *бизнес-процессов*. Наиболее подходящим типом диаграмм для визуального представления схем выполнения *бизнес-процессов* являются диаграммы деятельности, на которых дополнительно размещаются так называемые *дорожки* (**Swimlane**). Назначение ***дорожек*** состоит в том, чтобы указать зоны ответственности за выполнения отдельных деятельностей в рамках моделируемого *бизнес-процесса*. В качестве имен *дорожек* используются либо названия подразделений (департаментов) рассматриваемой компании, либо названия отдельных должностей сотрудников тех или иных подразделений.

Проекты по моделированию *бизнес-процессов* могут выполняться либо с целью реорганизации или реинжиниринга компании, либо с целью собственно документирования *бизнес-процессов*. Особенности данных проектов заключаются в том, что в обоих случаях необходимо построить модели *бизнес-процессов* некоторой существующей компании. Чтобы акцентировать внимание на подобных проектах, их часто называют проектами типа «As is» («Как есть»). Соответственно проекты по разработке новых продуктов или моделей новых систем называют проектами типа «To be» («Как должно быть»).

В данном контексте рассматриваемый ранее проект по разработке системы управления банкоматом следует отнести к проектам типа «Как есть», поскольку при построении диаграмм предполагалась известной существующая технология использования банкоматов для обслуживания клиентов. С другой стороны, если бы стояла цель разработки новой модели банкомата с некоторой дополнительной функциональностью или, например, разработки нового Интернет-магазина, то подобные проекты можно было бы отнести к проектам типа «Как должно быть». Именно этот тип проектов служит базовым для принятой в курсе лекций последовательности разработки канонических диаграмм в нотации UML, начиная от представления диаграмм вариантов использования и заканчивая диаграммами физического представления.

Выполнение проектов типа «Как есть» по моделированию *бизнес-процессов* в большинстве случаев начинают с построения диаграмм деятельности, которые служат для графического представления схем выполнения *бизнес-процессов* и документооборота рассматриваемой компании. После этого, исходя из требований проекта, разрабатывается модель диаграммы вариантов использования и выполняется реорганизация *бизнес-процессов*. Наконец, в случае необходимости разработки или внедрения корпоративной информационной системы, строятся диаграмма

классов, диаграммы взаимодействия и компонентов, которые служат основой для программной реализации соответствующего проекта.

Таким образом, первый этап выполнения проектов типа «Как есть» связан с построением моделей существующих *бизнес-процессов* компании в форме диаграмм деятельности. В качестве примера проекта этого типа в данном задании рассматривается модель *бизнес-процесса* по оптовой продаже товаров со склада торговой компании. Хотя данный пример имеет упрощенный характер, он позволяет наглядно представить основные особенности моделирования *бизнес-процессов* в нотации языка UML с использованием средства IBM Rational Rose 2003.

Для вновь разрабатываемого проекта по моделированию *бизнес-процессов* торговой компании в среде IBM Rational Rose 2003 создадим новый проект с именем: МодельБП. В качестве первой диаграммы проекта будет служить диаграмма деятельности, которая описывает отдельный *бизнес-процесс* в виде последовательности выполнения действий подразделениями компании при оптовой продаже товаров клиентам. Для удобства можно включить эту диаграмму в логическое представление, для чего необходимо в браузере проекта выделить логическое представление (**Logical View**) и выполнить операцию контекстного меню: **New→Activity Diagram** (Новая→Диаграмма деятельности).

Добавление дорожек на диаграмму деятельности

Для представления модели *бизнес-процесса* в форме диаграммы деятельности первоначально необходимо добавить на нее *дорожки*. Для добавления *дорожки* на диаграмму деятельности нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *дорожки* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. Добавить *дорожку* на диаграмму можно также с помощью операции главного меню: **Tools→Create→Swimlane** или с помощью операции контекстного меню: **New→Swimlane**, предварительно выделив диаграмму деятельности в браузере проекта.

В результате этих действий на диаграмме в области диаграммы появится изображение *дорожки* с вертикальной линией и именем *дорожки* **NewSwimlane** в верхней части, предложенное программой по умолчанию. Для задания имени *дорожки* следует открыть диалоговое окно спецификации ее свойств и ввести ее имя в поле ввода **Name** (рис.61).

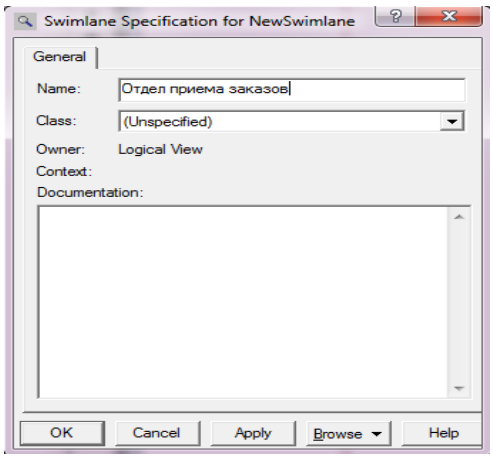


Рисунок 61 – Диалоговое окно спецификации свойств дорожки

Начиная практическую разработку модели *бизнес-процесса* оптовой продажи товаров со склада компании, последовательно добавим на диаграмму деятельности *дорожки* с именами отдельных подразделений компании: Отдел приема заказов, Бухгалтерия, Склад и Отдел доставки (рис.62).

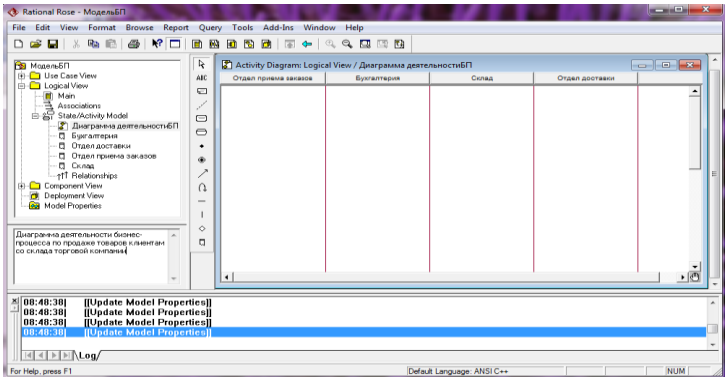


Рисунок 62 – Диаграмма деятельности после добавления на нее дорожек

После добавления *дорожек* на диаграмму состояний можно перейти к добавлению деятельностей и переходов. В качестве первой деятельности добавим деятельность с именем Принять заказ по факсу, которую разместим в первой дорожке с именем Отдел приема заказов. Этот факт будет означать, что деятельность Принять заказ по факсу выполняется в Отделе приема заказов или, другими словами, сотрудники этого отдела несут ответственность за выполнение данной деятельности.

Деятельности Принять заказ по факсу должно предшествовать начальное состояние, которое также следует добавить в эту же *дорожку* и соединить переходом с этой деятельностью. После добавления начального состояния и перехода диаграмма деятельности будет иметь следующий вид (рис. 63).

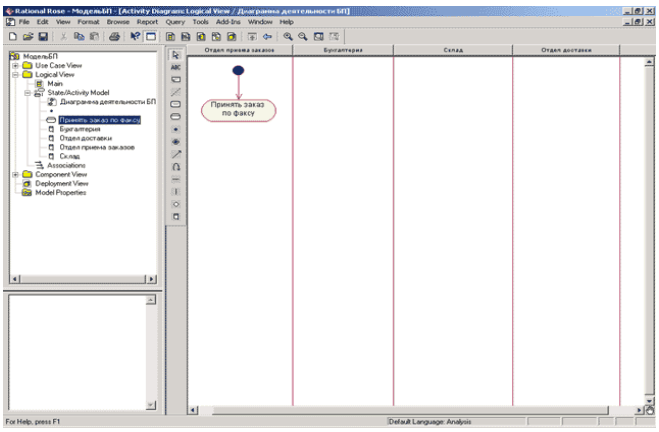


Рисунок 63 – Диаграмма деятельности после добавления на нее перехода из изначального состояния в деятельность Принять заказ по факсу

Построение диаграммы деятельности с дорожками для модели бизнес-процесса

Для построения диаграммы деятельности с *дорожками* для рассматриваемой модели *бизнес-процесса* следует добавить оставшиеся деятельности и переходы. С этой целью следует выполнить следующие действия:

1. Добавить деятельность с именем: Заказать товар на складе в *дорожку* Отдел приема заказов.
2. Добавить деятельности с именами: Выставить счет к оплате и Получить оплату за товар в *дорожку* Бухгалтерия.
3. Добавить деятельности с именами: Подобрать товар и Подготовить товар к отправке в *дорожку* Склад.
4. Добавить деятельность с именем: Отправить товар клиенту в *дорожку* Отдел доставки.
5. Добавить символ горизонтальной синхронизации в *дорожки* Отдел приема заказов и Отдел доставки. Следует заметить, что первый символ будет использован для разделения параллельных потоков деятельностей, а второй – для слияния этих потоков.
6. Добавить переход, направленный от деятельности Принять заказ по факсу к деятельности Заказать товар на складе.
7. Добавить переход, направленный от деятельности Заказать товар на складе к символу горизонтальной синхронизации.
8. Добавить переход, направленный от символа горизонтальной синхронизации к деятельности Выставить счет к оплате.
9. Добавить переход, направленный от символа горизонтальной синхронизации к деятельности Подобрать товар.
10. Добавить переход, направленный от деятельности Выставить счет к оплате к деятельности Получить оплату за товар.
11. Добавить переход, направленный от деятельности Подобрать товар к деятельности Подготовить товар к отправке.

12. Добавить переход, направленный от деятельности Получить оплату за товар к символу горизонтальной синхронизации.

13. Добавить переход, направленный от деятельности Подготовить товар к отправке к символу горизонтальной синхронизации.

14. Добавить переход, направленный от символа горизонтальной синхронизации к деятельности Отправить товар клиенту.

15. Добавить переход, направленный от деятельности Отправить товар клиенту к финальному состоянию.

Построенная таким образом диаграмма деятельности с *дорожками* будет иметь следующий вид (рис. 64).

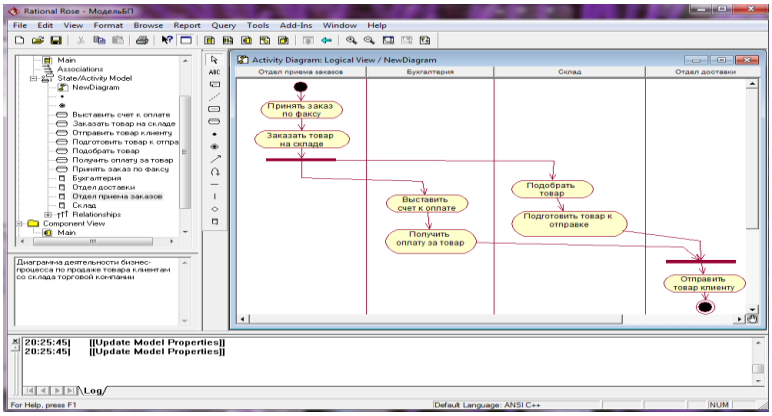


Рисунок 64. – Диаграмма деятельности с дорожками для модели бизнес-процесса

Дополнить диаграмму, которая учитывает условие когда заказанного клиентом товара не окажется на складе, в форме проверки отдельного условия, самостоятельно.

Построение диаграммы деятельности с дорожками и потоком объектов

Для построения диаграммы деятельности с *дорожками* и *потоком объектов* для рассматриваемой модели *бизнес-процесса* следует добавить на диаграмму *объекты* и стрелки *потоков объектов*. *Объекты* на диаграмме деятельности могут обозначать отдельные документы, которые необходимы для выполнения моделируемого *бизнес-процесса*. Соответственно ***поток объектов*** служит моделью документооборота рассматриваемой компании. Для добавления на диаграмму *объекта* следует воспользоваться соответствующей кнопкой на специальной панели инструментов. При этом данную кнопку предварительно следует на нее добавить, поскольку по умолчанию на панели она отсутствует.

В качестве первого *объекта* добавим на диаграмму деятельности объект с именем заказ, для которого зададим состояние: получен. Для задания состояния добавленного *объекта* следует открыть диалоговое окно свойств данного *объекта*, во вложенном списке **State** (Состояние) выбрать нужное состояние или задать новое (рис. 65). При этом будет открыто

дополнительное окно свойств состояния, в которое можно занести всю информацию по данному состоянию.

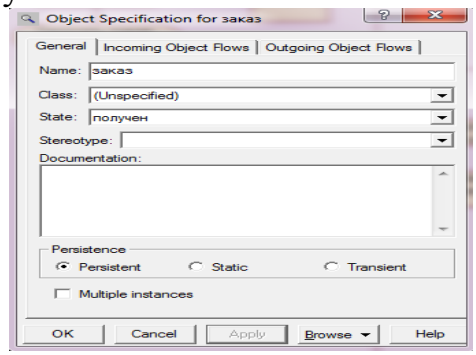


Рисунок 65 – Диалоговое окно спецификации свойств объекта

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

1. Добавить стрелку *потока объектов*, направленную от деятельности Принять заказ по факсу к объекту заказ в состоянии получен.
2. Добавить стрелку *потока объектов*, направленную от *объекта* заказ в состоянии получен к деятельности Заказать товар на складе.
3. Добавим объект с именем заказ, для которого зададим состояние: оформлен. Следует заметить, что для добавления на диаграмму деятельности уже существующего в модели *объекта* его следует просто перетащить из браузера проекта на диаграмму и задать ему новое состояние.
4. Добавить стрелку *потока объектов*, направленную от деятельности Заказать товар на складе к объекту заказ в состоянии оформлен.
5. Добавить стрелку *потока объектов*, направленную от *объекта* заказ в состоянии оформлен к деятельности Выставить счет к оплате.
6. Добавим объект с именем счет, для которого зададим состояние: выставлен.
7. Добавить стрелку *потока объектов*, направленную от деятельности Выставить счет к оплате к объекту счет в состоянии выставлен.
8. Добавить стрелку *потока объектов*, направленную от *объекта* счет в состоянии выставлен к деятельности Получить оплату за товар.
9. Добавим объект с именем счет, для которого зададим состояние: оплачен.
10. Добавить стрелку *потока объектов*, направленную от деятельности Получить оплату за товар к объекту счет в состоянии оплачен.
11. Добавить стрелку *потока объектов*, направленную от *объекта* счет в состоянии оплачен к деятельности Отправить товар клиенту.
12. Добавим объект с именем накладная, для которого зададим состояние: выписана.
13. Добавить стрелку *потока объектов*, направленную от деятельности Заказать товар на складе к объекту накладная в состоянии выписана.

14. Добавить стрелку *потока объектов*, направленную от *объекта* накладная в состоянии выписана к деятельности Подобрать товар.

15. Добавим объект с именем накладная, для которого зададим состояние: оформлена.

16. Добавить стрелку *потока объектов*, направленную от деятельности Подготовить товар к отправке к объекту накладная в состоянии оформлена.

17. Добавить стрелку *потока объектов*, направленную от *объекта* накладная в состоянии оформлена к деятельности Отправить товар клиенту.

Построенная таким образом диаграмма деятельности с *дорожками* и *потоком объектов* будет иметь следующий вид (рис. 66).

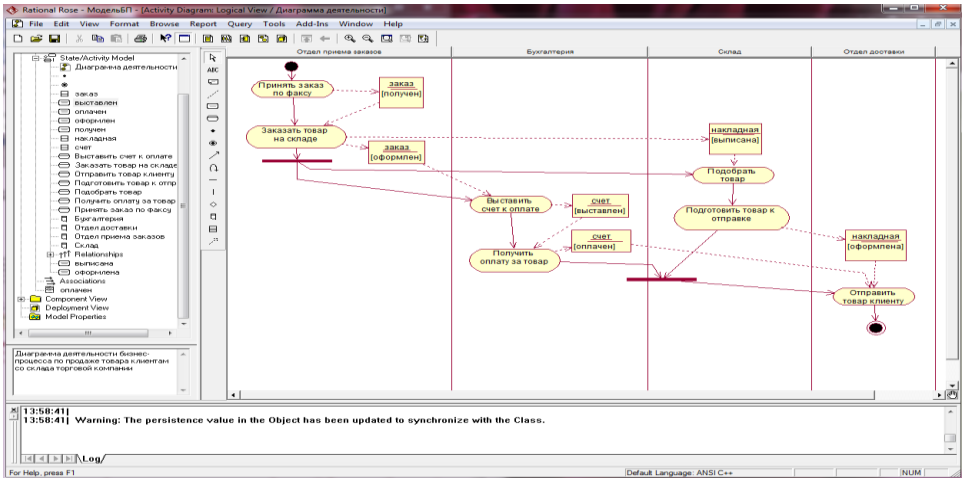


Рисунок 66 – Окончательный вид диаграммы деятельности

Для большей наглядности представления данной модели можно задать для всех деятельностей стереотип **Business Activity** (Бизнес-деятельность), который будет означать в данном контексте деятельность, выполняемую в рамках некоторого *бизнес-процесса*. Напомним, что изменить стереотип деятельности можно с помощью выбора нужного варианта стереотипа в окне спецификации свойств деятельности. Соответствующий вариант изображения диаграммы деятельности представлен на рис. 67.

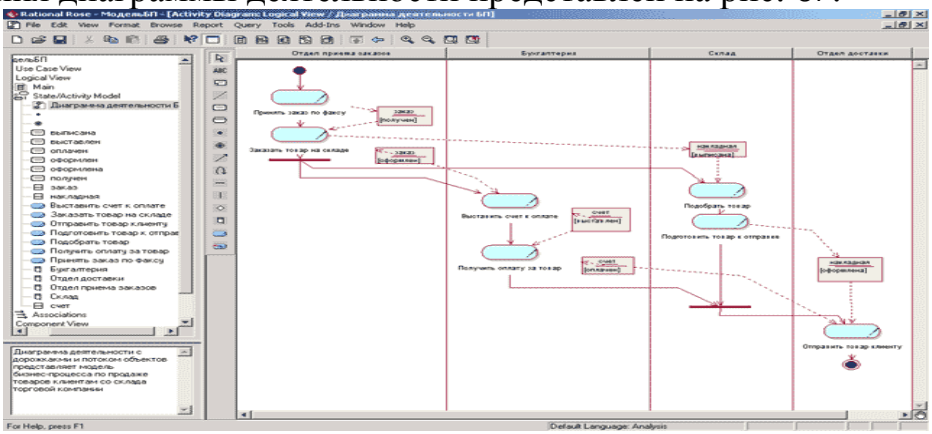


Рисунок 67 – Окончательный вид диаграммы деятельности

Следует заметить, что в разрабатываемой модели диаграмма деятельности не описывает ситуацию, когда клиент отказался от оплаты товара после выставления ему счета. Дополните данную диаграмму деятельности, которая учитывает данное условие, самостоятельно.

Хотя в среде IBM Rational Rose 2003 диаграмма деятельности не является необходимой для генерации программного кода, диаграммы данного типа имеют большое значение для документирования *бизнес-процессов* и их последующей сертификации по международному стандарту ISO 9000. Поэтому разработка диаграмм этого типа занимает центральное место при выполнении проектов по реинжинирингу и оптимизации *бизнес-процессов* с использованием нотации UML.

Задание 9. Разработка диаграммы компонентов и редактирование свойств ее элементов

Диаграмма *компонентов* служит частью физического представления модели, играет важную роль в процессе ООАП и является необходимой для генерации программного кода. Для разработки диаграмм *компонентов* в браузере проекта предназначено отдельное представление *компонентов* (**Component View**), в котором уже содержится диаграмма *компонентов* с пустым содержанием и именем по умолчанию **Main** (Главная).

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

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

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы *компонентов* и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы *компонентов* (табл.20).

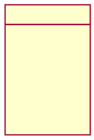
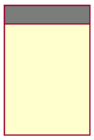
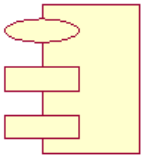
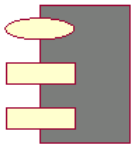
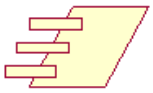
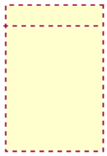
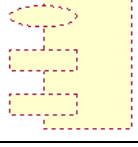
Таблица 20 – Назначение кнопок специальной панели инструментов диаграммы компонентов

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

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

Программа IBM Rational Rose 2003 предлагает целый ряд собственных *стереотипов*. Графическое изображение этих *стереотипов* и их краткая характеристика приводятся в следующей таблице (табл. 21). При этом каждому из *компонентов*, как правило, соответствует отдельный файл исходной сборки программного приложения.

Таблица 21 – Графическое изображение стереотипов компонентов и их характеристика

Графическое изображение и имя по умолчанию	**Название стереотипа**	**Характеристика стереотипа *компонента***
NewSubprogSpec 	Subprogram Specification	Спецификация подпрограммы. Содержит описание переменных, процедур и функций и не содержит определений классов
NewSubprogBody 	Subprogram Body	Тело подпрограммы. Содержит реализацию процедур и функций, не относящихся к каким-то классам, при этом не содержит определений классов или реализаций операций других классов
NewMainSubprog 	Main Program	Главная программа. Реализует базовую логику работы программного приложения и содержит ссылки на другие *компоненты* модели
NewPackageSpec 	Package Specification	Спецификация пакета. Содержит определение класса, его атрибутов и операций. В языке программирования C++ спецификации пакета соответствует отдельный файл с расширением «h»
NewPackageBody 	Package Body	Тело пакета. Содержит код реализации операций класса. В языке программирования C++ спецификации пакета соответствует отдельный файл с расширением «cpp»
NewTaskSpec 	Task Specification	Спецификация задачи. Может содержать определение класса, его атрибутов и операций, которые предполагается использовать в независимом потоке управления
NewTaskBody 	Task Body	Тело задачи. Может содержать реализацию операций класса, которые имеют независимый поток управления.
NewGenericSubprog 	Generic Subprogram	Типовая подпрограмма. Содержит описание переменных, процедур и функций, которые могут быть использованы в нескольких программных приложениях. При этом типовая подпрограмма не содержит определений классов
NewGenericPackage 	Generic Package	Типовой пакет. Содержит определение класса, его атрибутов и операций, которое может быть использовано в нескольких программных приложениях

	Database	База данных. Содержит определение одного или нескольких классов, их атрибутов и, возможно, операций. При этом соответствующие классы могут быть реализованы в форме одной или нескольких таблиц базы данных

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

Добавление компонента на диаграмму компонентов и редактирование его свойств

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

Изменим имя диаграммы Main, на Диаграмма *компонентов*, а для первого добавленного *компонента* зададим имя MainBAX.exe (рис. 68).

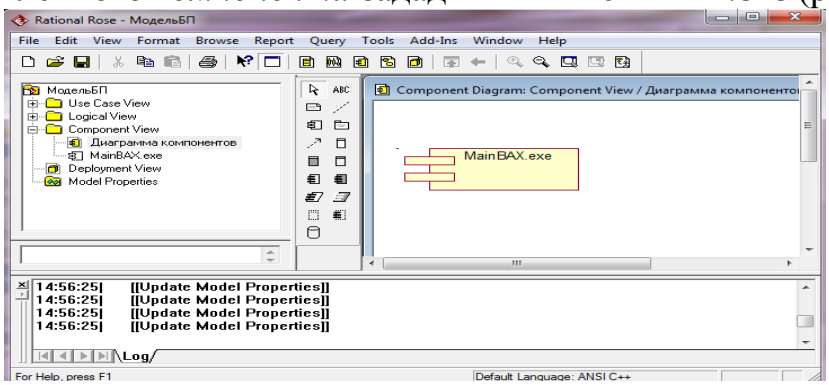


Рисунок 68 – Диаграмма компонентов после добавления компонента MainBAX.exe

Для каждого *компонента* можно определить различные свойства, такие как стереотип, язык программирования, декларации, реализуемые классы. Редактирование этих свойств для произвольного *компонента* осуществляется с помощью диалогового окна спецификации свойств (рис.69).

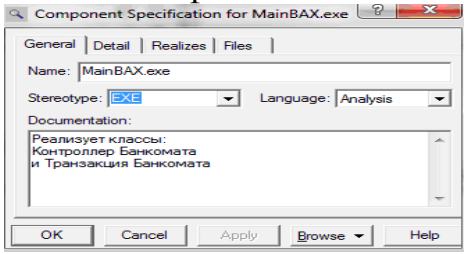


Рисунок 69 – Диалоговое окно спецификации свойств компонента MainBAX.exe

В частности, для *компонента* MainBAX.exe можно выбрать стереотип <<EXE>> из предлагаемого вложенного списка, поскольку применительно к разрабатываемой модели предполагается реализация этого *компонента* в форме исполнимого файла. При этом на вкладке Realizes (Реализует) содержатся все классы, включая и актеров, которые на данный момент присутствуют в модели (рис. 70). Следует заметить, что классы будут показаны в этом окне только при выбранном свойстве **Show all classes** (Показать все классы).

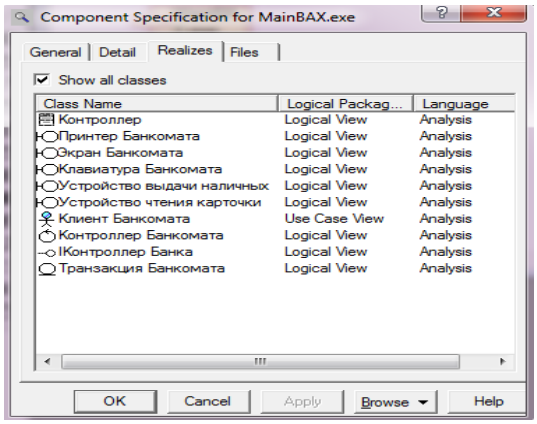


Рисунок 70 – Диалоговое окно спецификации свойств компонента MainBAX.exe, открытое на вкладке Realizes (Реализует)

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

Добавление отношения зависимости и редактирование его свойств

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

зависимости в форме пунктирной линии со стрелкой, соединяющей два выбранных *компонента*.

Применительно к диаграмме *компонентов* модели банкомата рассмотренным способом следует добавить отношение зависимости от *компонента* с именем MainBAX.exe к *компоненту* с именем MainBank. В дополнение к этому для наглядности можно указать в форме примечаний те классы модели, которые предполагается реализовать в данных *компонента*х (рис. 71).

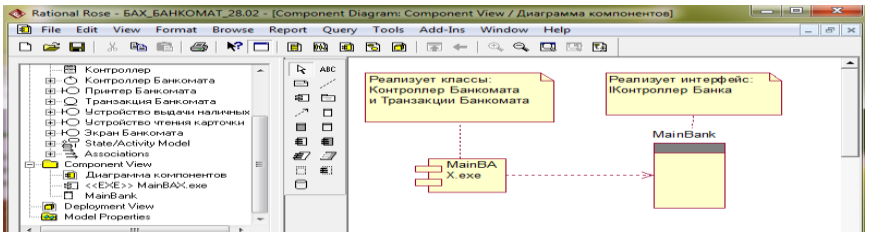


Рисунок 71 – Диаграмма компонентов после добавления отношения зависимости между компонентами MainBAX.exe и MainBank

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

Окончательное построение диаграммы компонентов модели банкомата

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

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

8. Добавить *зависимость* от *компонента* с именем Устройство выдачи наличных к *компоненту* с именем Устройства Банкомата.

Построенная таким образом диаграмма *компонентов* будет иметь следующий вид (рис. 9.5).

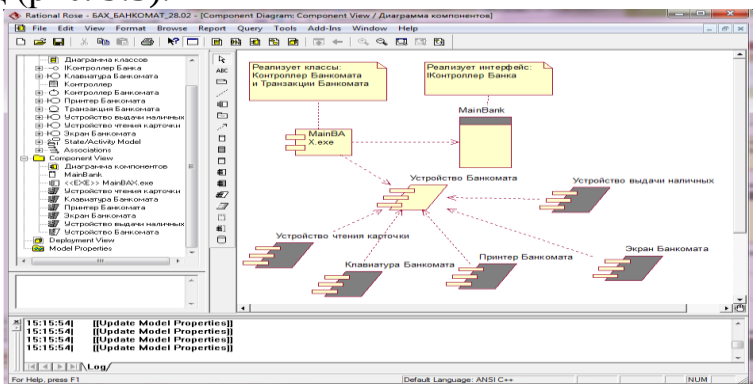


Рисунок 72 – Окончательный вид диаграммы компонентов разрабатываемой модели управления

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

Задание 10. Разработка диаграммы развертывания и редактирование свойств ее элементов

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

Активизация диаграммы развертывания может быть выполнена одним из следующих способов:

- Щелкнуть на кнопке с изображением диаграммы развертывания на стандартной панели инструментов.
- Дважды щелкнуть на пиктограмме представления развертывания (**Deployment View**) в браузере проекта.

- Выполнить операцию главного меню: **Browse→Deployment Diagram** (Обзор→Диаграмма развертывания).

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы развертывания и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы развертывания (табл.22).

Таблица 22 – Назначение кнопок специальной панели инструментов диаграммы развертывания

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

Работа с диаграммой развертывания состоит в создании *процессоров* и *устройств*, их спецификации, установлении связей между ними, а также добавлении и спецификации процессов.

Добавление узла на диаграмму развертывания и редактирование его свойств

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

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

Продолжая разработку модели системы управления банкоматом, построим для нее диаграмму развертывания. С этой целью в качестве первого *узла* выберем тип *процессор* и зададим ему имя Банкомат №1, для которого в форме примечания укажем *помеченное значение*: {адрес = ул. Ставропольская, д.36}. Это *значение* служит для спецификации конкретного адреса одного из банкоматов системы (рис. 73).

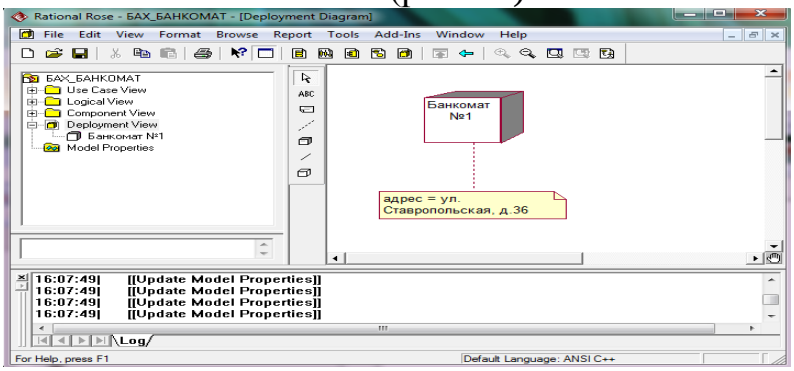


Рисунок 73 – Диаграмма развертывания после добавления узла Банкомат № 1

Для каждого *процессора* можно специфицировать различные свойства, такие как стереотип, характеристику, процессы и их приоритет. Спецификация этих свойств осуществляется с помощью диалогового окна спецификации свойств *процессора* (рис. 74).

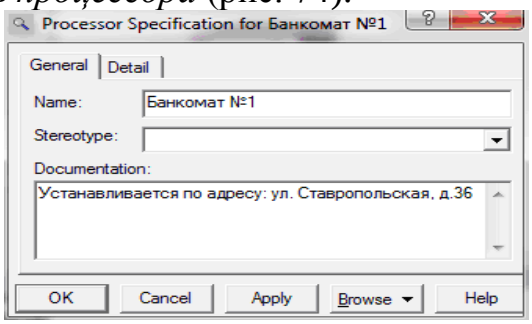


Рисунок 74 – Диалоговое окно спецификации свойств узла Банкомат №1

При этом на вкладке **General** (Общие) можно только изменить имя *процессора*, ввести текст стереотипа, предложенный самим разработчиком, и текст документации, поясняющий особенности физического размещения данного компонента. На вкладке **Detail** (Подробно) окна спецификации свойств *процессора* можно определить его характеристики, выбрать процессы и вариант *планирования* его работы (рис. 75).

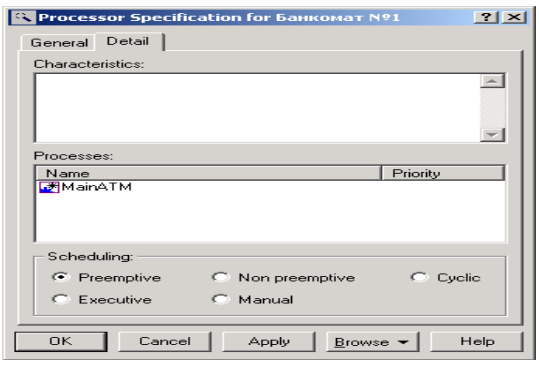


Рисунок 75 – Диалоговое окно спецификации свойств узла Банкомат №1, открытое на вкладке Detail (Подробно)

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

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

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

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

отдельных процессов на выбранном *процессоре* следует выполнить операцию контекстного меню **Show Scheduling** (Показать планирование).

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

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

Добавление соединения и редактирование его свойств

Для добавления соединения между двумя *узлами* нужно с помощью левой кнопки мыши нажать кнопку с изображением соединения на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении одного из *узлов* на диаграмме и отпустить ее на изображении другого *узла*. Добавить соединения на диаграмму развертывания можно также с помощью операции главного меню: **Tools→Create→Connection**.

В результате этих действий на диаграмме появится изображение соединения в форме линии без стрелок, соединяющей два выбранных *узла*. Применительно к диаграмме развертывания модели банкомата одним из рассмотренных способов следует добавить *соединение* для *узлов* с именами Банкомат №1 и Сеть (рис. 76).

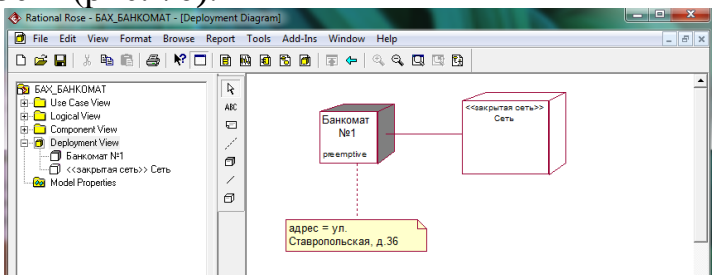


Рисунок 76 – Диаграмма развертывания после добавления соединения между узлами Банкомат № 1 и Сеть

Для завершения построения диаграммы развертывания рассматриваемого примера следует описанным выше способом добавить оставшиеся узлы и соединения. С этой целью следует выполнить следующие действия:

1. Добавить *процессор* с именем: Банкомат №2, для которого задать *помеченное значение* в форме примечания: {адрес = ул. Парковая, д.7}, а на вкладке свойств **Detail** определить новый процесс и выбрать для него имя MainBAX из вложенного списка.
2. Добавить *процессор* с именем: Банкомат №3, для которого задать *помеченное значение* в форме примечания: {адрес = ул. Лесная, д.9}, а на

вкладке свойств **Detail** определить новый процесс и выбрать для него имя MainBAX из вложенного списка.

3. Добавить *процессор* с именем: Сервер Банка, для которого на вкладке свойств **Detail** определить новый процесс с именем MainBank.
4. Добавить *соединение* для *узлов* с именами Банкомат №2 и Сеть.
5. Добавить *соединение* для *узлов* с именами Банкомат №3 и Сеть.
6. Добавить *соединение* для *узлов* с именами Сервер Банка и Сеть.

Построенная таким образом диаграмма развертывания будет иметь следующий вид (рис. 77), причем для данной диаграммы показаны выполняемые на *процессорах* процессы и не показаны процедуры их *планирования*. Это сделано по той причине, что при наличии единственного процесса планирование ресурсов *процессора* теряет свое значение.

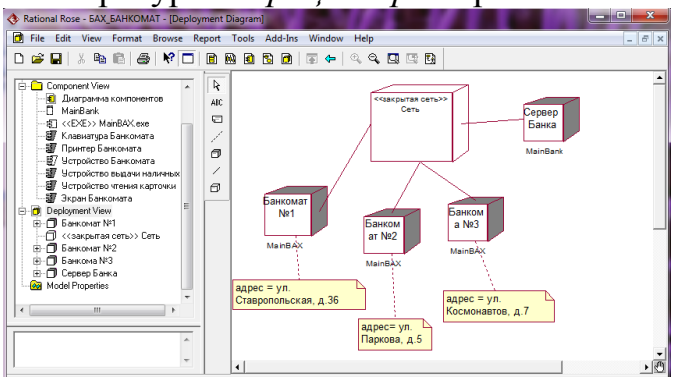


Рисунок 77 – Окончательный вид диаграммы развертывания разрабатываемой модели управления банкоматом

Следует отметить, что программа IBM Rational Rose 2003 не поддерживает возможности графического размещения внутри *узлов* развертываемых на них компонентов. Указать размещение компонентов модели в *узлах* диаграммы развертывания можно с помощью документации соответствующих *узлов*. Выполнить эти действия самостоятельно. После построения диаграммы развертывания разработка визуальной модели системы управления банкоматом в нотации UML может считаться завершенной.

Дальнейшая работа с моделью зависит от целей выполнения проекта. Если проект не предполагает программную реализацию, то можно ограничиться формированием проектной документации. С этой целью следует выполнить операцию главного меню: **Report→SoDA Report_** (Отчет→Отчет с помощью SoDA), в результате чего будет открыто диалоговое окно свойств для выбора шаблонов генерации отчета. После выбора шаблонов будет автоматически сгенерирован отчет о разрабатываемой модели в формате MS Word с использованием специального средства IBM Rational SoDA, если оно доступно в системе после инсталляции IBM Rational Rose 2003.

Задание 11. Особенности генерации программного кода в среде IBM Rational Rose 2003

Подготовка модели для генерации программного кода

Одним из наиболее важных свойств программы IBM Rational Rose 2003 является возможность генерации программного кода на нескольких языках программирования, которая может быть использована разработчиком после построения модели. Для этой цели в среде IBM Rational Rose 2003 присутствует достаточно большой выбор *языков программирования* и схем баз данных. Однако возможность генерации текста программы на том или ином языке программирования зависит от установленной версии IBM Rational Rose 2003.

Общая последовательность действий, которые необходимо выполнить для генерации программного кода в среде IBM Rational Rose 2003, состоит из следующих этапов:

1. Проверка модели на отсутствие ошибок.
2. Создание *компонентов* для реализации *классов*.
3. Отображение *классов* на *компоненты*.
4. Выбор *языка программирования* для генерации текста программного кода.
5. Установка свойств генерации программного кода.
6. Выбор *класса*, *компонента* или пакета.
7. Генерация программного кода.

Особенности выполнения каждого из этапов могут изменяться в зависимости от выбора *языка программирования* или схемы базы данных.

В среде IBM Rational Rose 2003 предусмотрено задание достаточно большого числа свойств, характеризующих как отдельные *классы*, так и проект в целом. Для определенности в качестве языка реализации проекта целесообразно выбрать *язык программирования ANSI C++*, который не требует инсталляции дополнительных программ и поставляется практически во всех конфигурациях IBM Rational Rose 2003. Рассмотрим особенности выполнения каждого из указанных выше этапов для языка реализации модели *ANSI C++*.

Поскольку язык ANSI C++ не допускает использование символов кириллицы в качестве имен *классов*, атрибутов и операций, необходимо соответствующим образом модифицировать диаграмму *классов*. После изменения имен *классов*, атрибутов и операций диаграмма *классов* модели банкомата будет иметь следующий вид (рис. 78).

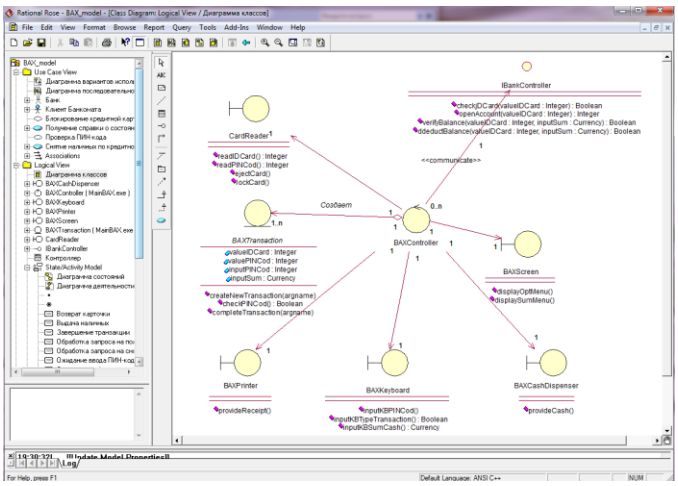


Рисунок 78 – Диаграмма классов модели банкомата после преобразования имен классов, их атрибутов и операций

Проверка модели независимо от выбора языка генерации кода

В общем случае проверка модели может выполняться на любом этапе работы над проектом. Однако после завершения разработки графических диаграмм она является обязательной, поскольку позволяет выявить целый ряд ошибок разработчика. К числу таких ошибок и предупреждений относятся, например, не используемые ассоциации и *классы*, оставшиеся после удаления отдельных графических элементов с диаграмм, а также операции, не являющиеся именами сообщений на диаграммах взаимодействия.

Для проверки модели следует выполнить операцию главного меню: **Tools→Check Model** (Инструменты→Проверить модель). Результаты проверки разработанной модели на наличие ошибок отображаются в окне журнала. Прежде чем приступить к генерации текста программного кода разработчику следует добиться устранения всех ошибок и предупреждений, о чем должно свидетельствовать чистое окно журнала (рис. 79).

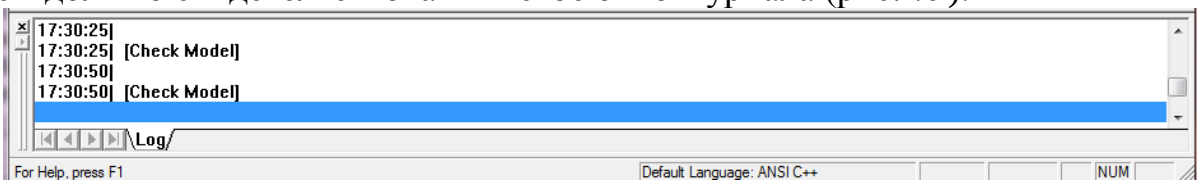


Рисунок 79 – Вид журнала при отсутствии ошибок по результатам проверки модели

Создание компонентов для реализации классов и отображение классов на компоненты

По существу данные этапы выполняются в ходе разработки диаграммы *компонентов*. Хотя программа IBM Rational Rose 2003 позволяет генерировать программный код на языке *ANSI C++* для каждого *класса* модели без предварительного построения диаграммы *компонентов*, имеет смысл воспользоваться разработанной ранее диаграммой *компонентов*.

Применительно к разрабатываемому проекту желательно переименовать *компоненты*, задав им англоязычные имена(рис. 80).

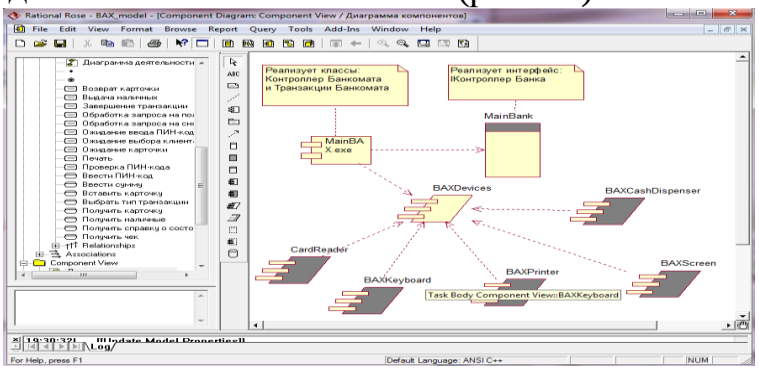


Рисунок 80 – Диаграмма компонентов модели банкомата после преобразования имен компонентов

Для отображения *классов* на *компоненты* можно воспользоваться окном спецификации свойств *компонента*, открытого на вкладке **Realizes** (Реализует). Для включения реализации *класса* в данный *компонент* следует выделить требуемый *класс* на этой вкладке и выполнить для него операцию контекстного меню **Assign** (Назначить). В результате перед именем *класса* на этой вкладке появится специальная отметка.

Применительно к модели банкомата для *компонента* MainBAX.exe выберем для генерации программного кода *классы* BAXTransaction и BAXController (рис. 81).

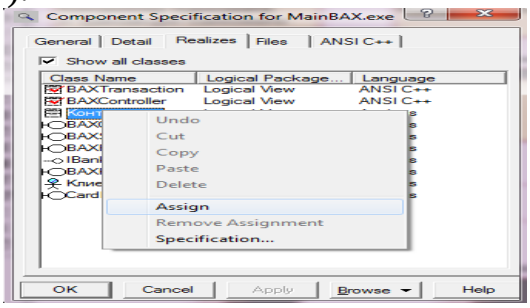


Рисунок 81 – Диалоговое окно настройки свойств реализации классов в компоненте MainBAX.exe

Подобная операция должна быть выполнена для всех *классов* модели, которые предполагается реализовывать на выбранном языке программирования. Имеется и другой способ установления реализации *классов* на компоненте. А именно, можно просто выделить класс в браузере проекта и перетащить его на нужный *компонент* диаграммы *компонентов*.

Выбор языка программирования и редактирование свойств генерации программного кода

Для выбора языка *ANSI C++* в качестве языка реализации модели следует выполнить операцию главного меню: **Tools→Options** (Инструменты→Параметры), в результате чего будет вызвано диалоговое окно настройки параметров модели. Далее на вкладке **Notation** (Нотация) в

строке **Default Language** (Язык по умолчанию) из вложенного списка следует выбрать язык – *ANSI C++*.

Если по какой-то причине языка *ANSI C++* не оказалось во вложенном списке, то следует убедиться в том, что этот *язык программирования* установлен в качестве расширения IBM Rational Rose 2003. Для этого следует открыть окно установленных расширений, выполнив операцию главного меню: **Add-Ins→Add-In Manager** (Расширения→Менеджер расширений), и убедиться в том, что выставлена отметка в строке с именем языка *ANSI C++*. Если ее нет, то ее следует добавить, после чего появится группа доступных операций *ANSI C++* в главном меню **Tools**.

После выбора *языка программирования* по умолчанию следует изменить язык реализации каждого из *компонентов* модели. С этой целью следует изменить язык в строке **Language** (Язык) на вкладке **General** (Общие) окна спецификации свойств *компонента*, для чего из вложенного списка следует выбрать язык – *ANSI C++* (рис. 82).

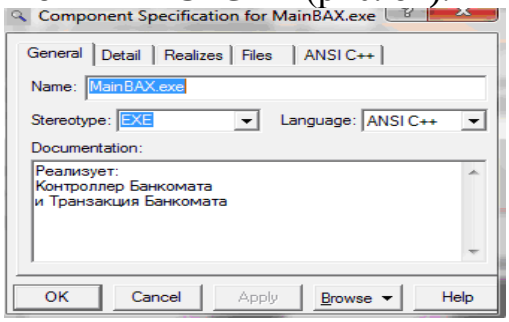


Рисунок 82 – Окно спецификации свойств компонента MainBAX.exe при выборе языка его реализации

Следует заметить, что после выбора *языка программирования* следует привести в соответствие типы атрибутов, типы аргументов и возвращаемых значений операций. С этой целью нужно просмотреть все *классы* диаграммы *классов* и изменить те типы данных, которые не являются синтаксически допустимыми в выбранном языке программирования. Применительно к языку *ANSI C++* следует заменить тип **Integer** на **int**, **Boolean** на **bool**, **Currency** на **float**. В противном случае соответствующие исправления придется выполнять вручную после генерации программного кода.

Редактирование общих свойств генерации программного кода возможно в специальном диалоговом окне, которое может быть открыто в результате выполнения операции главного меню: **Tools**→*ANSI C++*→Open *ANSI C++* Specification (Инструменты→Язык *ANSI C++*→Открыть спецификацию языка *ANSI C++*). Дополнительные свойства генерации программного кода отдельного *класса* можно специфицировать в диалоговом окне, которое может быть открыто в результате выполнения операции контекстного меню: *ANSI C++*→Class Customization (Язык *ANSI C++*→ Настройка свойств *класса*). При этом соответствующий *класс* должен быть выделен в браузере проекта.

При генерации программного кода на языке *ANSI C++* для модели банкомата значения свойств, предлагаемых средой IBM Rational Rose 2003 по умолчанию, первоначально можно оставить без изменения.

Выбор класса или компонента и генерация для него программного кода

Выбор *класса* или *компонента* для генерации программного кода означает выделение соответствующего элемента модели в браузере проекта. Применительно к рассматриваемой модели системы управления банкоматом для генерации программного кода на языке *ANSI C++* выберем *компонент* с именем MainATM.exe.

Генерация программного кода в среде IBM Rational Rose 2003 возможна для отдельного *класса* или *компонента*. Для этого нужный элемент модели предварительно следует выделить в браузере проекта и выполнить операцию контекстного меню: **ANSI C++→Generate Code_** (Язык *ANSI C++*→Генерировать код). В результате этого будет открыто диалоговое окно с предложением выбора *классов* для генерации программного кода на выбранном языке программирования (рис. 83). После выбора соответствующих *классов* и нажатия кнопки OK программа IBM Rational Rose 2003 выполняет кодогенерацию.

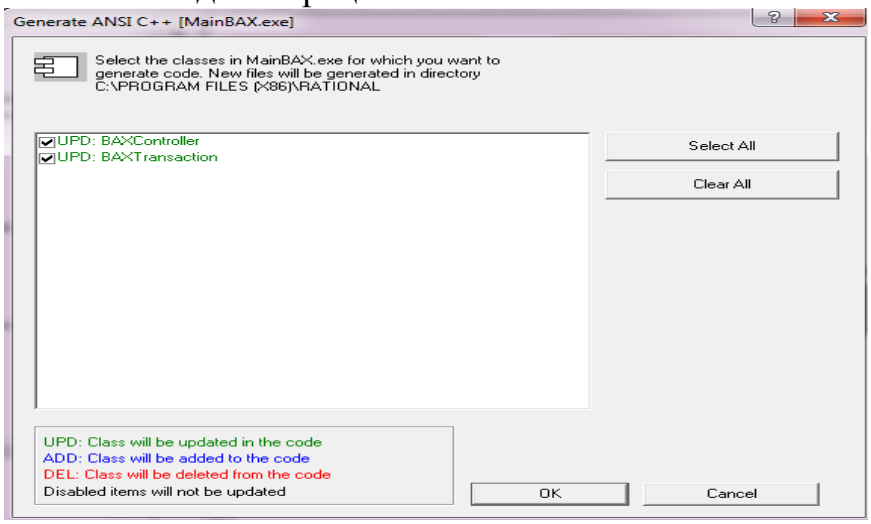


Рисунок 83 – Окно выбора классов для генерации программного кода

Для просмотра и редактирования созданных файлов с текстом программного кода на языке *ANSI C++* предназначен встроенный текстовый редактор, который можно открыть с помощью операции контекстного меню: *ANSI C++*→Browse Header_ (Язык *ANSI C++*→Просмотреть *заголовочный файл*) или **ANSI C++→Browse Body_** (Язык *ANSI C++*→Просмотреть *файл реализации*) для выбранного *класса* в браузере проекта.

После генерации программного кода для *компонента* MainBAX.exe каждому *классу*, реализованному в данном компоненте, будет соответствовать 2 файла с текстом кода на языке *ANSI C++*. Так, например, для *класса* BAXTransaction будет сгенерирован *заголовочный файл* с расширением «h» (рис. 84) и *файл реализации* с расширением «cpp» (рис.85).

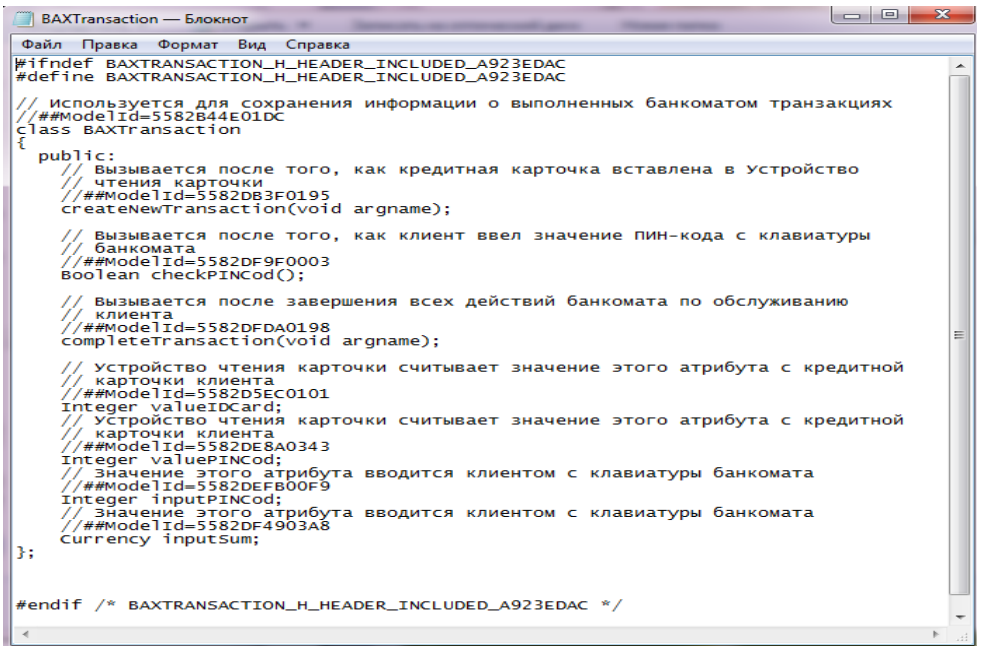


Рисунок 84 – Вид встроенного текстового редактора с загруженным в него заголовочным файлом BAXTransaction.h

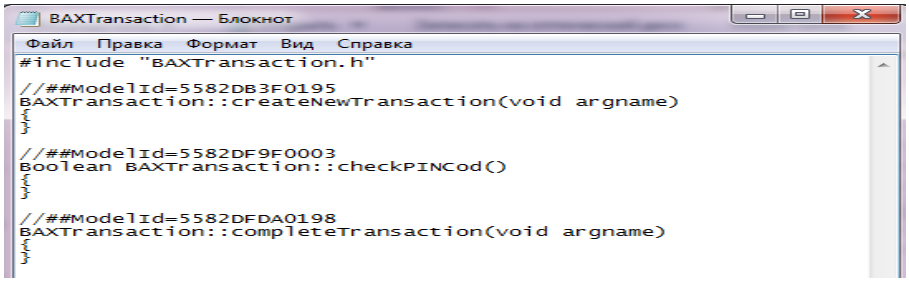


Рисунок 85 – Вид встроенного текстового редактора с загруженным в него заголовочным файлом BAXTransaction.cpp

Как видно из рассмотрения полученного *заголовочного файла*, в нем содержится объявление в соответствии с правилами синтаксиса языка *ANSI C++* всех операций и атрибутов *класса* BAXTransaction. При этом информация о документировании операций и атрибутов помещается в комментарии перед соответствующими элементами программы.

В файле реализации содержится заготовка для реализации всех операций *класса* BAXTransaction в соответствии с правилами синтаксиса языка *ANSI C++*. При этом каждая из операций имеет пустое тело реализации, которое следует написать дополнительно, исходя из функциональных требований модели и синтаксиса *языка программирования ANSI C++*. Данную работу удобнее выполнять в выбранной интегрированной среде программирования, например, MS Visual C++ или Borland C++. При использовании интегрированной среды кроме компиляции, отладки и тестирования исходных модулей программы разработчик получает возможность дополнить приложение графическим интерфейсом, необходимым для взаимодействия с пользователем.

Следует заметить, что при установленной на компьютер разработчика интегрированной среды сгенерированные файлы с текстом программного кода автоматически открываются в этой среде после двойного щелчка на пиктограмме этих файлов. Тем не менее, лучше копировать содержимое этих файлов в предварительно созданные программные проекты для полного контроля в этих средах процесса программирования и отладки приложений.

Сгенерированные программой IBM Rational Rose 2003 файлы с текстом программного кода содержат минимум информации. Для включения дополнительных элементов в программный код следует изменить свойства генерации программного кода, установленные по умолчанию. Сгенерировать файлы с текстом программного кода при различных значениях свойств выбранного *языка программирования* предлагается читателям самостоятельно.

В заключение следует отметить, что эффект от использования средства IBM Rational Rose 2003 проявляется при разработке масштабных проектов в составе команды или проектной группы. Действительно, при рассмотрении модели системы управления банкоматом может сложиться впечатление того, что написать и отладить соответствующую программу гораздо проще непосредственно в той или иной интегрированной среде программирования.

Однако ситуация покажется не столь тривиальной, когда станет необходимо выполнить проект с несколькими десятками вариантов использования и сотней *классов*. Именно для подобных проектов явно выявляется преимущество использования средства IBM Rational Rose 2003 и нотации языка UML для документирования и реализации соответствующих моделей.

Список литературы

1. Грекул В. И., Денищенко Г. Н., Коровкина Н. Л. Проектирование информационных систем. Интернет-университет информационных технологий – ИНТУИТ.ру, 2005.

2. Данилин А., Слюсаренко А. Архитектура и стратегия. "Инь" и "янь" информационных технологий. Интернет-университет информационных технологий – ИНТУИТ.ру, 2005.

3. Маклаков С. В. Моделирование бизнес-процессов с AIIFusion Process Modeler. – М.: Диолог-МИФИ, 2003. – 240 с.

4. Леоненков А. В. Самоучитель UML. – СПб.: БХВ-Петербург, 2001. – 304 с.: ил.

5. Нейбург Э. Д., Максимчук Р. А. Проектирование баз данных с помощью UML. – М.: Издательский дом «Вильямс», 2002

6. Елиферов В. Г., Репин В.В. Бизнес-процессы: регламентация и управление. – М.: ИНФРА-М, 2004

7. Смирнова Г. Н., Сорокин А. А., Тельнов Ю. Ф. Проектирование экономических информационных систем. – М.: Финансы и статистика, 2002

8. Вендров А. М. Проектирование программного обеспечения экономических информационных систем: Учебник. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2005.

9. Тельнов Ю. Ф. Реинжиниринг бизнес-процессов. Компонентная методология. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2004.

БАШИЕВА Анжела Хамидовна

ПРОЕКТЫЙ ПРАКТИКУМ

Учебно-методическое пособие для обучающихся по направлению подготовки 09.03.03 Прикладная информатика

Корректор Чагова О.Х.
Редактор Чагова О.Х.

Сдано в набор 03.06.2024 г.
Формат 60х84/16
Бумага офсетная
Печать офсетная
Усл. печ. л. 5,18
Заказ № 4891
Тираж 100 экз.

Оригинал-макет подготовлен
в Библиотечно-издательском центре СКГА
369000, г. Черкесск, ул. Ставропольская, 36