

Министерство образования и науки Российской Федерации
Московский государственный институт электронной техники
(технический университет)
Кафедра «Вычислительной техники»

М.Н. Пуцин

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Учебное пособие

Москва
2008

УДК 004.451 (076.5)

Рецензенты: канд. техн. наук, *В.И. Ухандеев*, канд. техн. наук, *Д. Сударенко*

М.Н. Пуцин

Проектирование информационных систем: Учеб. пособие. - М: Изд-во МИЭТ, 2008. - 234 с.

Учебное пособие содержит материал по курсу "Проектирование информационных систем" и представляет собой практическое руководство по созданию информационных систем с помощью популярных CASE-средств.

Пособие содержит описание методов структурного анализа и проектирования моделей данных в объеме необходимом для практической работы. Подробно на конкретных примерах рассмотрено применения CASE-технологий и CASE-средств для автоматизации этапов анализа, проектирования и генерации кода информационных систем.

Пособие предназначено для студентов изучающих основы системного анализа и проектирования информационных систем.

Автор выражает глубокую признательность и благодарность студентам МИЭТ кафедры "Вычислительная техника", принявшим активное участие в подготовке учебного пособия.

Введение

Информация в современном мире превратилась в один из наиболее важных ресурсов, а информационные системы (ИС) стали необходимым инструментом практически во всех сферах деятельности.

Анализ современного состояния рынка ИС показывает устойчивую тенденцию роста спроса на информационные системы организационного управления. Причем спрос продолжает расти именно на интегрированные системы управления [4].

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности ИС, создаваемых в различных областях деятельности.

Для успешной реализации ИС, объект проектирования должен быть адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов [1]. Все это способствует появлению программно-технологических средств специального класса – CASE-средств (Computer Aided Software Engineering), реализующих CASE-технологии создания и сопровождения ИС.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного или объектно-ориентированного анализа и проектирования, использующих специфика-

ции в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств [1].

Учебное пособие является компиляцией популярных материалов, раскрывающих практические аспекты применения CASE-средств визуального проектирования, таких профессиональных авторов и практиков в этой области как: *Маклаков С.В., Леоненков А.В., Вендров А.М., Грекул В.И., Денищенко Г.Н., Коровкина Н.Л.*

В первой главе учебного пособия рассматривается ведущий инструмент визуального моделирования бизнес-процессов AllFusion Process Modeler (BPwin). Дано описание методологии и инструментальных средств, а также набор практических заданий, позволяющих освоить технику создания функциональных моделей. Последовательно рассматриваются три нотации моделирования, поддерживаемые BPwin: IDEF0, IDEF3 и DFD.

Вторая глава посвящена средству моделирования данных и проектированию баз данных AllFusion ERwin Data Modeler (ERwin). Описаны возможности проектирования, документирования и сопровождения базы данных и хранилища данных. Описаны механизмы разработки модели на логическом и физическом уровнях. Рассматривается наглядная модель базы данных с практическими заданиями для читателей.

Предметом третьей главы является рассмотрение практических особенностей процесса объектно-ориентированного моделирования и разработки проектов программных приложений с использованием CASE-средства IBM Rational Rose 2003. Описываются элементы рабочего интерфейса программы и рекомендации по выполнению проекта в нотации UML. Процесс разработки моделей в среде IBM Rational Rose 2003 иллюстрируется практическими примерами построения конкретных диаграмм в нотации UML.

Глава 1. Моделирование бизнес-процессов средствами BPwin

1.1. Создание контекстной диаграммы

BPwin поддерживает три методологии моделирования: функциональное моделирование (IDEF0), описание бизнес-процессов (IDEF3), диаграммы потоков данных (DFD), каждая из которых решает свои специфические задачи. В BPwin возможно построение смешанных моделей, т. е. модель может содержать одновременно диаграммы как IDEF0, так и IDEF3 и DFD. Состав палитры инструментов изменяется автоматически, когда происходит переключение с одной нотации на другую [7].

На начальных этапах создания ИС необходимо понять, как работает организация, которую собираются автоматизировать. Руководитель хорошо знает *работу* в целом, но не в состоянии вникнуть в детали *работы* каждого рядового сотрудника. Рядовой сотрудник хорошо знает, что творится на его рабочем месте, но может не знать, как работают коллеги. Поэтому для описания *работы* предприятия необходимо построить модель, которая будет адекватна предметной области и содержать в себе знания всех участников бизнес-процессов организации [4].

Наиболее удобным языком моделирования бизнес-процессов является IDEF0, где система представляется как совокупность взаимодействующих *работ* или функций. Такая чисто функциональная ориентация является принципиальной. Функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Основу методологии IDEF0 составляет графический язык описания бизнес-процессов. Модель в нотации IDEF0 представляет собой совокупность ие-

рархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Модель может содержать четыре типа диаграмм [4]:

- *контекстную диаграмму* (в каждой модели может быть только одна *контекстная диаграмма*);
- *диаграммы декомпозиции*;
- *диаграммы дерева узлов*;
- *диаграммы только для экспозиции (FEO)*.

Процесс моделирования системы в IDEF0 начинается с создания *контекстной диаграммы* — диаграммы наиболее абстрактного уровня описания системы в целом, содержащей определение субъекта моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, определить, что будет в дальнейшем рассматриваться как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будут существенно влиять позиция, с которой рассматривается система, и цель моделирования, это вопросы, на которые построенная модель должна дать ответ. Другими словами, в начале необходимо определить область моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели. Хотя предполагается, что в ходе моделирования область может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования. При формулировании области необходимо учитывать два компонента: широту и глубину. Широта подразумевает определение границ модели, что будет рассматриваться внутри системы, а что снаружи. Глубина определяет, на каком уровне детализации модель является завершённой.

Цель моделирования определяется из ответов на следующие вопросы [4, 7]:

- Почему этот процесс должен быть смоделирован?
- Что должна показывать модель?
- Что может получить клиент?

Под **точкой зрения** (Viewpoint) понимается перспектива, с которой наблюдалась система при построении модели. Точка зрения должна соответствовать цели и границам моделирования. Как правило, выбирается точка зрения человека, ответственного за моделируемую *работу* в целом.

Обычно сначала строится модель существующей организации *работы* AS-IS (как есть). Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес-процессов и насколько глубоким изменениям подвергнется существующая структура организации бизнеса. Детализация бизнес-процессов позволяет выявить недостатки организации даже там, где функциональность на первый взгляд кажется очевидной. Найденные в модели AS-IS недостатки можно исправить при создании модели TO-BE (как будет), модели новой организации бизнес-процессов.

Технология проектирования ИС подразумевает сначала создание модели AS-IS, ее анализ и улучшение бизнес-процессов, то есть создание модели TO-BE, и только на основе модели TO-BE строится модель данных, прототип и затем окончательный вариант ИС.

Модель в VPwin рассматривается как совокупность *работ*, каждая из которых оперирует с некоторым набором данных.

Работы (Activity) обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. *Работы* изображаются в виде прямоугольников. Все

работы должны быть названы и определены. Имя *работы* должно быть выражено отглагольным существительным, обозначающим действие.

Стрелки (Arrow) описывают взаимодействие *работ* и представляют собой некую информацию, выраженную существительными. В IDEF0 различают пять типов стрелок:

Вход (Input) — материал или информация, которые используются или преобразуются *работой* для получения результата (выхода). Допускается, что *работа* может не иметь ни одной *стрелки* входа. Каждый тип стрелок подходит к определенной стороне прямоугольника, изображающего *работу*, или выходит из нее. *Стрелка* входа рисуется как входящая в левую грань *работы*. Очень часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить информация о том, перерабатываются/изменяются ли данные в *работе* или нет. Если изменяются, то, скорее всего, это вход, если нет — управление.

Управление (Control) — правила, стратегии, процедуры или стандарты, которыми руководствуется *работа*. Каждая *работа* должна иметь хотя бы одну *стрелку* управления. *Стрелка* управления рисуется как входящая в верхнюю грань *работы*. Управление влияет на *работу*, но не преобразуется *работой*. Если цель *работы* — изменить процедуру или стратегию, то такая процедура или стратегия будет для *работы* входом. В случае возникновения неопределенности в статусе *стрелки* (управление или вход) рекомендуется рисовать *стрелку* управления.

Выход (Output) — материал или информация, которые производятся *работой*. Каждая *работа* должна иметь хотя бы одну *стрелку* выхода. *Работа* без результата не имеет смысла и не должна моделироваться. *Стрелка* выхода рисуется как исходящая из правой грани *работы*.

Механизм (Mechanism) — ресурсы, которые выполняют *работу*, например персонал предприятия, станки, устройства и т. д. *Стрелка* механизма ри-

суется как входящая в нижнюю грань *работы*. По усмотрению аналитика *стрелки* механизма могут не изображаться в модели.

Вызов (Call) — специальная *стрелка*, указывающая на другую модель *работы*. *Стрелка* вызова рисуется как исходящая из нижней грани *работы*. *Стрелка* вызова используется для указания того, что некоторая *работа* выполняется за пределами моделируемой системы. В BPwin *стрелки* вызова используются в механизме слияния и разделения моделей [7].

Стрелки на *контекстной диаграмме* служат для описания взаимодействия системы с окружающим миром. Они могут начинаться у границы диаграммы и заканчиваться у *работы*, или наоборот. Такие *стрелки* называются *граничными*.

В качестве примера будем рассматривать деятельность вымышленной компании. Компания занимается в основном сборкой карманных персональных компьютеров (КПК) и установкой на них программного обеспечения (ПО). Компания не производит компоненты и программы самостоятельно, а только собирает КПК, устанавливает ПО, и проводит комплексное тестирование.

Основные процедуры компании, следующие:

- продавцы принимают заказы клиентов;
- операторы группируют заказы по типу работ;
- операторы собирают КПК и тестируют;
- операторы устанавливают ПО на КПК и тестируют его;
- операторы упаковывают КПК согласно заказам;
- кладовщик отгружает клиентам заказы.

Компания использует купленную бухгалтерскую информационную систему, которая позволяет оформить заказ, счет и отследить платежи по счетам.

1. Запустите Computer Associates BPwin.

2. Если появляется диалог ModelMart Connection Manager, нажмите на кнопку Cancel.
3. Щелкните по кнопке . Появится диалог I would like to. Внесите имя модели «Деятельность компании» и выберите Type – IDEF0. Нажмите OK. В открывшемся окне Properties for New Models, нажмите OK.
4. Автоматически создается контекстная диаграмма.
5. Обратите внимание на кнопку  на панели инструментов. Эта кнопка включает и выключает инструмент просмотра и навигации – Model Explorer (появляется слева). Model Explorer имеет три вкладки – Activities, Diagrams и Objects. Во вкладке Activities щелчок правой кнопкой по объекту позволяет редактировать его свойства.
6. Если вам не понятно, как выполнить то или иное действие, вы можете вызвать помощь – клавиша F1 или меню Help.
7. Перейдите в меню Model/Model Properties. Во вкладке General диалога Model Properties следует ввести имя модели «Деятельность компании», имя проекта «Модель деятельности компании», имя автора и тип модели – Time Frame: AS-IS.
8. Во вкладке Purpose внести цель – «Purpose: Моделировать текущие (AS-IS) бизнес-процессы компании» и точку зрения – «Viewpoint: Директор».
9. Во вкладке Definition внесите определение «Это учебная модель, описывающая деятельность компании» и цель «Score: Общее управление бизнесом компании: исследование рынка, закупка компонентов, сборка, тестирование и продажа продуктов».
10. Перейдите на контекстную диаграмму и правой кнопкой мыши щелкните по работе. В контекстном меню выберите Name. Во вкладке Name внесите имя «Деятельность компании».
11. Во вкладке Definition внесите определение «Текущие бизнес-процессы компании».

12. Создайте стрелки (кнопка  на палитре инструментов) на контекстной диаграмме (табл. 1.1). Для внесения имен и свойств стрелок щелкните правой кнопкой мыши по ветви стрелки.

Таблица 1.1. Стрелки контекстной диаграммы

<i>Arrow Name</i>	<i>Arrow Definition</i>	<i>Arrow Type</i>
Бухгалтерская система	Оформление счетов, оплата счетов, работа с заказами	Mechanism
Звонки клиентов	Запросы информации, заказы, техподдержка и т.д.	Input
Правила и процедуры	Правила продаж, инструкции по сборке и установке, процедуры тестирования, критерии производительности и т.д.	Control
Проданные продукты	КПК и ПО	Output

13. С помощью кнопки  внесите текст в поле диаграммы – точку зрения и цель (рис. 1.1).

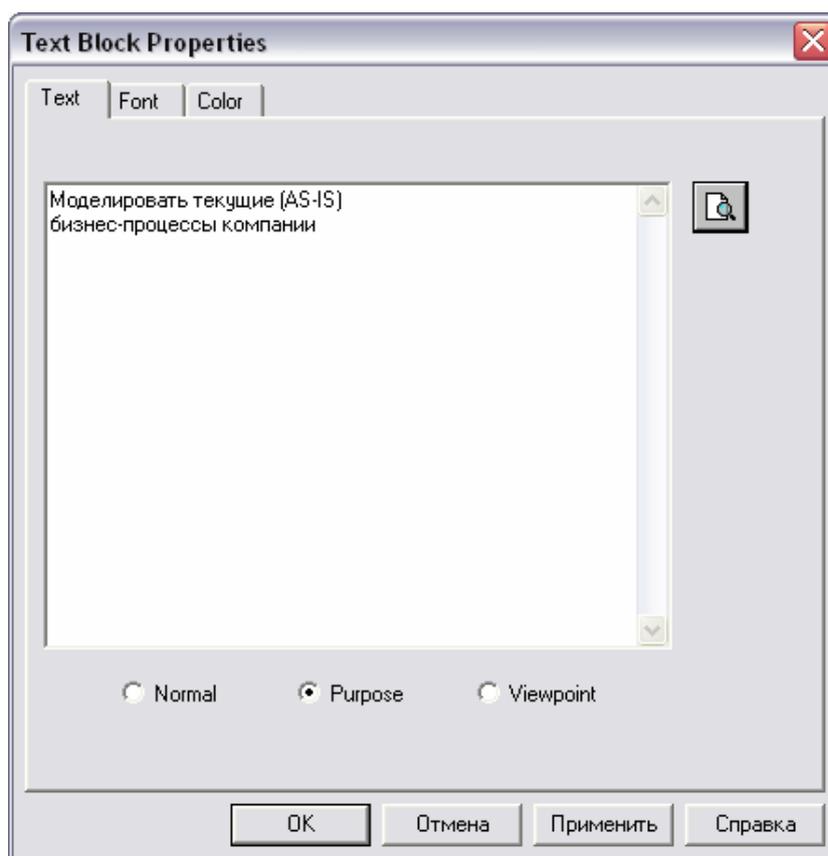


Рис. 1.1. Задание точки зрения и цели в окне Text Block Properties

Результат выполнения задания показан на рис. 1.2.

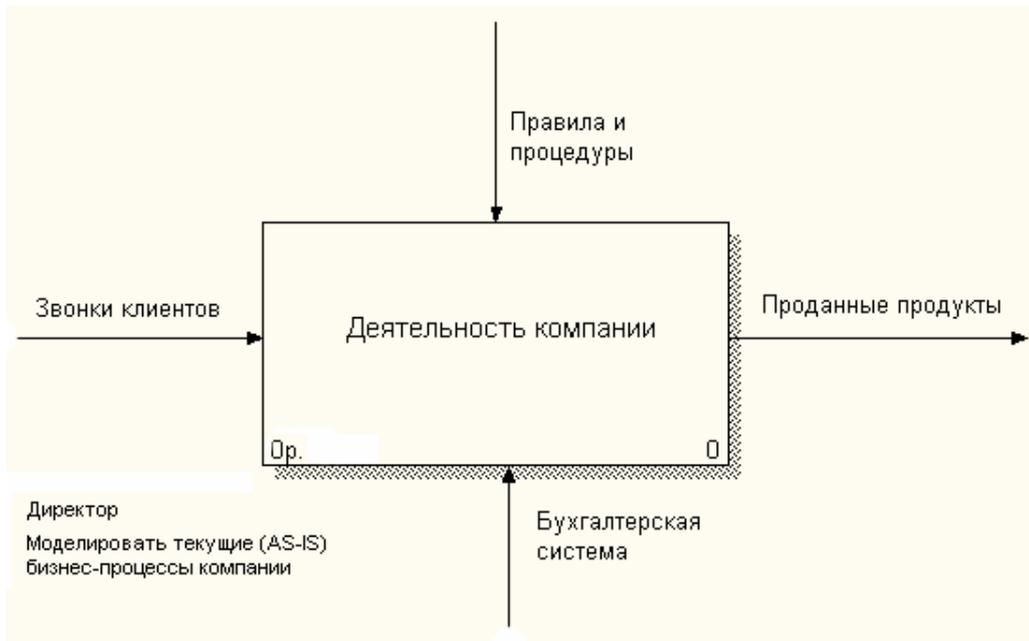


Рис. 1.2. Контекстная диаграмма

14. Создайте отчет по модели. Меню Tools/Reports/Model Report (рис. 1.3).

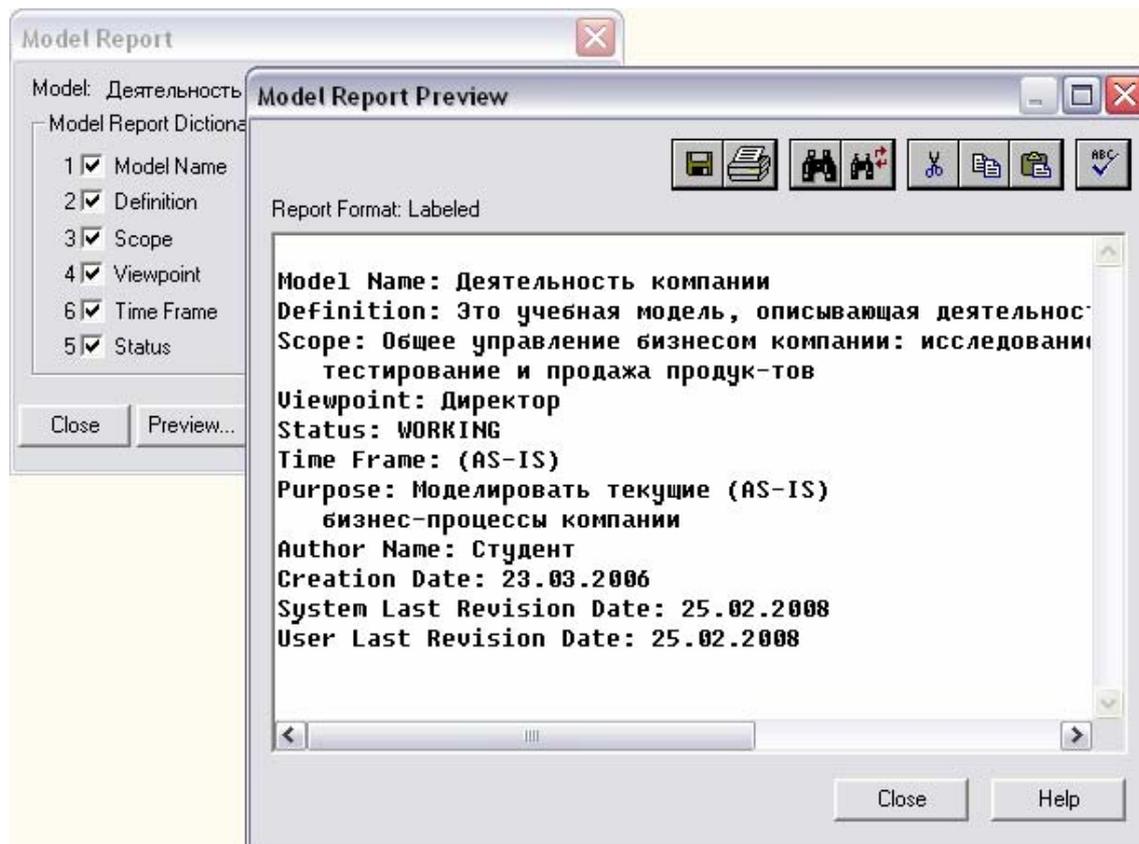


Рис. 1.3. Отчет Model Report

1.2. Создание диаграммы декомпозиции

После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции. После декомпозиции *контекстной диаграммы* проводится декомпозиция каждого большого фрагмента системы на более мелкие и так далее, до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы. Эксперты предметной области указывают на соответствие реальных бизнес-процессов созданным диаграммам. Найденные несоответствия исправляются, и только после прохождения экспертизы без замечаний можно

приступать к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес-процессам на любом и каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели [4].

Диаграмма декомпозиции предназначена для детализации *работы*. В отличие от моделей, отображающих структуру организации, *работа* на диаграмме верхнего уровня в IDEF0 — это не элемент управления нижестоящими *работами*. *Работы* нижнего уровня — это то же самое, что *работы* верхнего уровня, но в более детальном изложении. Как следствие этого границы *работы* верхнего уровня — это то же самое, что границы диаграммы декомпозиции. ICOM (аббревиатура от Input, Control, Output и Mechanism) — коды, предназначенные для идентификации граничных стрелок. Код ICOM содержит префикс, соответствующий типу *стрелки* (I, C, O или M), и порядковый номер.

ВРwin вносит ICOM-коды автоматически. Для отображения ICOM-кодов следует включить опцию ICOM codes на закладке Display диалога Model Properties (меню Model/Model Properties).

Диаграммы декомпозиции содержат родственные *работы*, т.е. дочерние *работы*, имеющие общую родительскую *работу*. *Работы* на диаграммах декомпозиции обычно располагаются по диагонали от левого верхнего угла к правому нижнему. Такой порядок называется порядком доминирования. Согласно этому принципу расположения в левом верхнем углу помещается самая важная *работа* или *работа*, выполняемая по времени первой. Далее вправо вниз располагаются менее важные или выполняемые позже *работы*. Такое размещение облегчает чтение диаграмм, кроме того, на нем основывается понятие взаимосвязей *работ*.

Каждая из *работ* на диаграмме декомпозиции может быть в свою очередь декомпозирована. На диаграмме декомпозиции *работы* нумеруются автома-

тически слева направо. Номер *работы* показывается в правом нижнем углу. В левом верхнем углу изображается небольшая диагональная черта, которая показывает, что данная *работа* не была декомпозирована.

1. Выберите кнопку перехода ▼ на нижней уровень в палитре инструментов и в диалоге Activity Box Count установите число работ на диаграмме нижнего уровня – 3, и нажмите ОК (рис. 1.4).

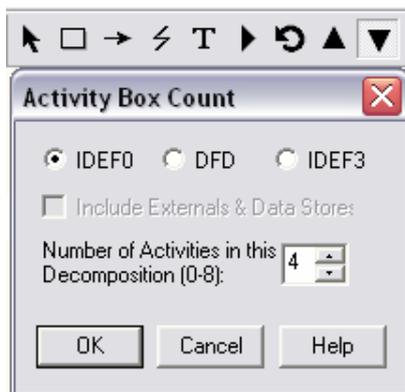


Рис. 1.4. Диалог Activity Box Count

Автоматически будет создана диаграмма декомпозиции. Правой кнопкой мыши щелкните по работе, выберите Name и внесите имя работы. Повторите операцию для всех трех работ. Затем внесите определение, статус и источник для каждой работы согласно табл. 1.2.

Таблица 1.2. Работы диаграммы декомпозиции A0

Activity Name	Definition
Продажи и маркетинг	Телереклама, наружная реклама, реклама в прессе, выставки
Сборка и настройка КПК	Сборка КПК, установка программного обеспечения, тестирование
Отгрузка и получение	Отгрузка заказов клиентам и получение компонентов от поставщиков

2. Для изменения свойств работ после их внесения в диаграмму можно воспользоваться словарем работ. Вызов словаря – меню Dictionary/Activity (рис. 1.5).



Рис. 1.5. Словарь Activity Dictionary

Если описать имя и свойства работы в словаре, ее можно будет внести в диаграмму позже с помощью кнопки  в палитре инструментов. Невозможно удалить работу из словаря, если она используется на какой-либо диаграмме. Если работа удаляется из диаграммы, из словаря она не удаляется. Имя и описание такой работы может быть использовано в дальнейшем. Для добавления работы в словарь необходимо перейти в конец списка и щелкнуть правой кнопкой мыши по последней строке. Возникает новая строка, в которой нужно ввести имя и свойства работы [7]. Для удаления всех имен работ, не использующихся в модели, щелкните по кнопке Корзина (Purge).

3. При декомпозиции *работы* входящие в нее и исходящие из нее *стрелки* (кроме *стрелки* вызова) автоматически появляются на диаграмме декомпозиции (миграция стрелок), но при этом не касаются *работ*. Такие *стрелки* называются несвязанными и воспринимаются в BPwin как синтаксическая ошибка.

Для связывания стрелок входа, управления или механизма необходимо перейти в режим редактирования стрелок (кнопка  на палитре инструментов), щелкнуть по наконечнику *стрелки* и потом по соответствующему сегменту *работы*. Для связывания *стрелки* выхода необходимо перейти в режим

редактирования стрелок, щелкнуть по сегменту выхода *работы* и затем по *стрелке*.

Для разветвления *стрелки* нужно в режиме редактирования *стрелки* щелкнуть по фрагменту *стрелки* и по соответствующему сегменту *работы*. Для слияния двух стрелок выхода нужно в режиме редактирования *стрелки* сначала щелкнуть по сегменту выхода *работы*, а затем по соответствующему фрагменту *стрелки*.

Свяжите граничные стрелки так, как показано на рис. 1.6.

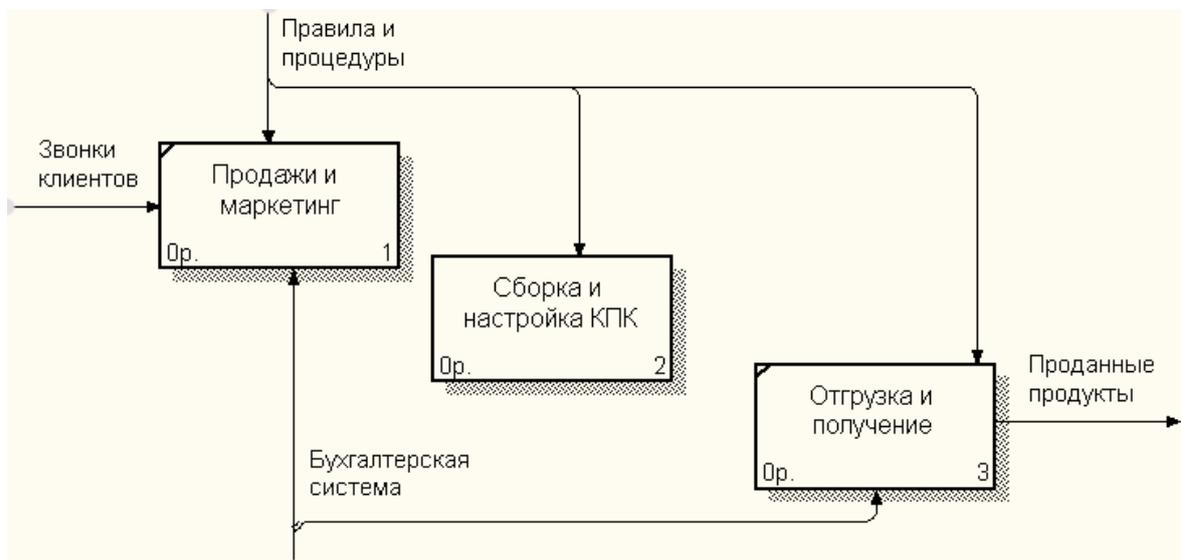


Рис. 1.6. Связанные граничные стрелки на диаграмме А0

4. Смысл разветвляющихся и сливающихся стрелок передается именованнием каждой ветви стрелок. Существуют определенные правила именования таких стрелок. Если *стрелка* именована до разветвления, а после разветвления ни одна из ветвей не именована, то подразумевается, что каждая ветвь моделирует те же данные или объекты, что и ветвь до разветвления. Если *стрелка* именована до разветвления, а после разветвления какая-либо из ветвей тоже именована, то подразумевается, что эти ветви соответствуют именованию. Если при этом какая-либо ветвь после разветвления осталась неименованной, то подразумевается, что она моделирует те же данные или объекты, что и ветвь до разветвления.

Недопустима ситуация, когда *стрелка* до разветвления не именована, а после разветвления не именована какая-либо из ветвей. ВРwin определяет такую *стрелку* как синтаксическую ошибку [4].

Правила именования сливающихся стрелок полностью аналогичны — ошибкой будет считаться *стрелка*, которая после слияния не именована, а до слияния не именована какая-либо из ее ветвей. Для именования отдельной ветви разветвляющихся и сливающихся стрелок следует выделить на диаграмме только одну ветвь, после чего вызвать редактор имени и присвоить имя *стрелке*. Это имя будет соответствовать только выделенной ветви.

Правой кнопкой мыши щелкните по ветви стрелки управления работы «Сборка и настройка КПК» и переименуйте ее в «Правила сборки и настройки» (рис. 1.7).

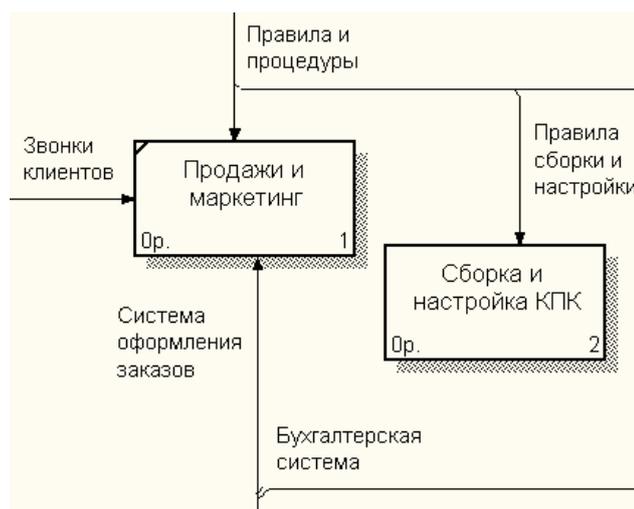


Рис. 1.7. Стрелка «Правила сборки и настройки»

Внесите определения для новой ветви: «Инструкции по сборке, последовательность установки, процедуры тестирования, критерии производительности и т.д.»

Правой кнопкой мыши щелкните по ветви стрелки механизма работы «Продажи и маркетинг» и переименуйте ее в «Система оформления заказов».

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

Словарь стрелок решает очень важную задачу. Диаграммы создаются аналитиком для того, чтобы провести сеанс экспертизы, т. е. обсудить диаграмму со специалистом предметной области. В любой предметной области формируется профессиональный жаргон, причем очень часто жаргонные выражения имеют нечеткий смысл и воспринимаются разными специалистами по-разному. В то же время аналитик — автор диаграмм должен употреблять те выражения, которые наиболее понятны экспертам. Поскольку формальные определения часто сложны для восприятия, аналитик вынужден употреблять профессиональный жаргон, а чтобы не возникло неоднозначных трактовок, в словаре стрелок каждому понятию можно дать расширенное и, если это необходимо, формальное определение [4].

Содержимое словаря стрелок можно распечатать в виде отчета (меню Tools/ Report /Arrow Report...) и получить толковый словарь терминов предметной области, использующихся в модели.

6. Для связи *работ* между собой используются внутренние *стрелки*, то есть *стрелки*, которые не касаются границы диаграммы, начинаются у одной и кончаются у другой *работы*. Для рисования внутренней *стрелки* необходимо в режиме рисования стрелок щелкнуть по сегменту (например, выхода) од-

ной *работы* и затем по сегменту (например, входа) другой. Создайте новые внутренние стрелки так, как показано на рис. 1.8.

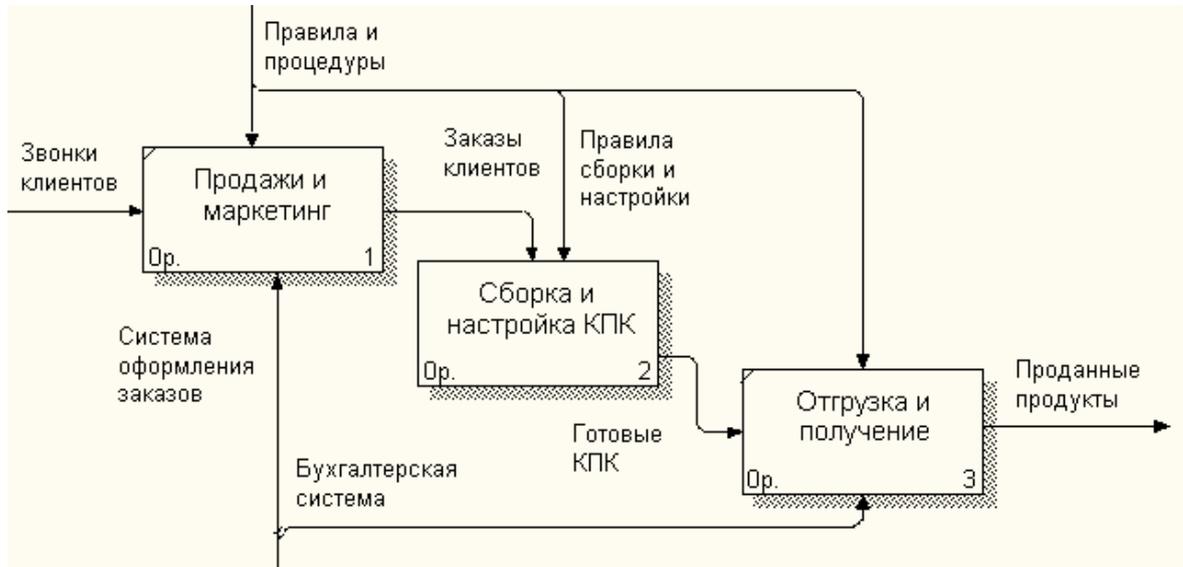


Рис. 1.8. Внутренние стрелки диаграммы A0

7. Когда выход нижестоящей *работы* направляется на управление вышестоящей, такая стрелка называется обратной связью по управлению. Обратная связь по управлению часто свидетельствует об эффективности бизнес-процесса. Создайте стрелку обратной связи по управлению «*Результаты сборки и настройки*», идущую от работы «*Сборка и настройка КПК*» к работе «*Продажа и маркетинг*». Измените стиль стрелки (толщина линий) и установите опцию Extra Arrowhead (из контекстного меню). Перенесите имена стрелок так, чтобы их было удобнее читать. Если необходимо, установите Squiggle (из контекстного меню). Результат изменений показан на рис. 1.9.

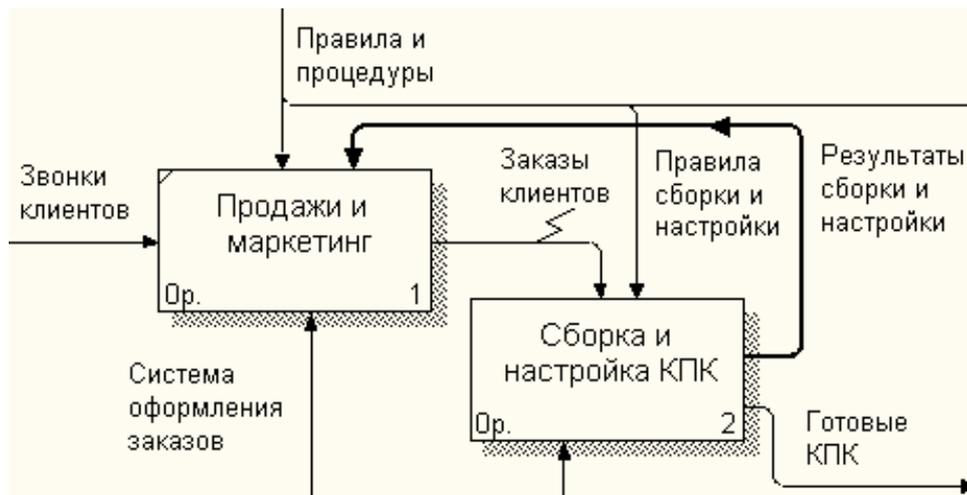


Рис. 1.9. Результат редактирования стрелок на диаграмме А0

8. Создайте новую граничную стрелку выхода «Маркетинговые материалы», выходящую из работы «Продажи и маркетинг». Эта стрелка автоматически не попадает на диаграмму верхнего уровня и имеет квадратные скобки на кончике. Щелкните правой кнопкой мыши по квадратным скобкам и выберите пункт меню Arrow Tunnel. В появившемся диалоге Border Arrow Editor если щелкнуть по кнопке Resolve Border Arrow, стрелка мигрирует на диаграмму верхнего уровня, если по кнопке Change To Tunnel — стрелка будет туннелирована и не попадет на другую диаграмму. Туннельная стрелка изображается с круглыми скобками на конце. В диалоге Border Arrow Editor выберите опцию Resolve it to Border Arrow.

Туннелирование может быть применено для изображения малозначимых стрелок. Если на какой-либо диаграмме нижнего уровня необходимо изобразить малозначимые данные или объекты, которые не обрабатываются или не используются работами на текущем уровне, то их необходимо направить на вышестоящий уровень (на родительскую диаграмму). Если эти данные не используются на родительской диаграмме, их нужно направить еще выше, и т. д. В результате малозначимая стрелка будет изображена на всех уровнях и затруднит чтение всех диаграмм, на которых она присутствует. Выходом явля-

ется туннелирование *стрелки* на самом нижнем уровне. Такое туннелирование называется «не в родительской диаграмме» [4].

Для стрелки «*Маркетинговые материалы*» выберите опцию Trim из контекстного меню. Результат создания диаграммы декомпозиции показан на рис. 1.10.

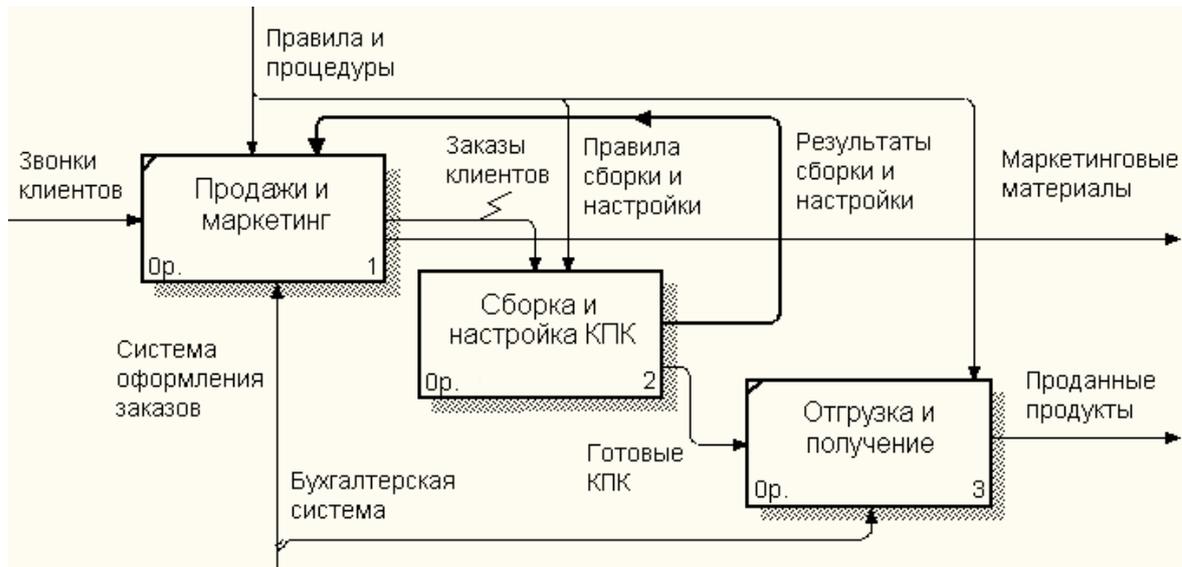


Рис. 1.10. Результат создания диаграмма декомпозиции A0

1.3. Создание диаграммы декомпозиции A2

Все *работы* модели нумеруются. Номер состоит из префикса и числа. Может быть использован префикс любой длины, но обычно используют префикс А. Контекстная (корневая) *работа* дерева имеет номер А0. *Работы* i декомпозиции А0 имеют номера А1, А2, А3 и т. д. *Работы* декомпозиции нижнего уровня имеют номер родительской *работы* и очередной порядковый номер, например *работы* декомпозиции А3 будут иметь номера А31, А32, А33, А34 и т. д. *Работы* образуют иерархию, где каждая *работа* может иметь одну родительскую и несколько дочерних *работ*, образуя дерево. Такое дерево называют деревом узлов, а вышеописанную нумерацию — нумерацией по

узлам. Диаграммы IDEF0 имеют двойную нумерацию. Во-первых, диаграммы имеют номера по узлу. *Контекстная диаграмма* всегда имеет номер A-0, декомпозиция *контекстной диаграммы* — номер A0, остальные диаграммы декомпозиции — номера по соответствующему узлу (например, A1, A2, A21, A213 и т. д.). ВРwin автоматически поддерживает нумерацию по узлам, т. е. при проведении декомпозиции создается новая диаграмма и ей автоматически присваивается соответствующий номер [4].

Декомпозируем работу «Сборка и настройка КПК».

В результате проведения экспертизы получена следующая информация. Производственный отдел получает заказы клиентов от отдела продаж по мере их поступления [7].

Диспетчер контролирует работу сборщиков, сортирует заказы, группирует их и дает указания на отгрузку КПК, как они готовы.

Каждые два часа диспетчер группирует заказы – отдельно для сборки КПК и установки ПО на них – и направляет на участок сборки.

Сотрудники участка сборки и настройки собирают КПК и устанавливают ПО согласно спецификациям заказа и инструкциям по сборке и настройке. Когда группа КПК, соответствующая группе заказов, собрана, она направляется на тестирование. Тестируется каждый КПК, и в случае необходимости заменяют неисправные аппаратные или программные компоненты.

Отдел тестирования направляет результаты тестирования диспетчеру, который на основании этой информации принимает решение о передаче компьютеров, соответствующих группе заказов, на отгрузку.

1. На основе этой информации внесите новые работы и стрелки (табл. 1.3 и 1.4).

Таблица 1.3. Работы диаграммы декомпозиции A2

Activity Name	Definition
Отслеживание расписания и управления сборкой, ус-	Просмотр заказов, установка расписания выполнения заказов, просмотр результатов тести-

тановкой и тестированием	рования, формирование групп заказов на сборку и отгрузку
Сборка КПК	Сборка карманных персональных компьютеров в соответствии с инструкциями и указаниями диспетчера
Установка ПО	Установка программного обеспечения на КПК в соответствии с инструкциями и указаниями диспетчера
Тестирование КПК	Тестирование КПК, компонентов и ПО. Замена неработающих компонентов

Таблица 1.4. Стрелки диаграммы декомпозиции A2

Arrow Name	Arrow Source	Arrow Source Type	Arrow Dest.	Arrow Dest. Type
Диспетчер	Персонал производственного отдела		Отслеживание расписания и управление сборкой, установкой и тестированием	Mechanism
Заказы клиентов	Граница диаграммы	Control	Отслеживание расписания и управление сборкой, установкой и тестированием	Control
Заказы на сборку КПК	Отслеживание расписания и управление сборкой, установкой и тестированием	Output	Сборка КПК	Control
Заказы на установку ПО	Отслеживание расписания и управление сборкой, установкой и тестированием	Output	Установка ПО	Control
Компоненты	«Tunnel»	Input	Сборка КПК	Input
			Установка ПО	Input
			Тестирование	Input

			КПК	
КПК	Сборка КПК	Output	Установка ПО	Input
КПК	Сборка КПК	Output	Тестирование КПК	Input
КПК с ПО	Установка ПО	Output	Тестирование КПК	Input
Персонал производственного отдела	«Tunnel»	Mechanism	Сборка КПК	Mechanism
			Установка ПО	Mechanism
Правила сборки и настройки	Граница диаграммы		Сборка КПК	Control
			Установка ПО	Control
			Тестирование КПК	Control
Результаты сборки и настройки	Сборка КПК	Output	Граница диаграммы	Output
	Установка ПО	Output		
	Тестирование КПК	Output		
Результаты тестирования	Тестирование КПК	Output	Отслеживание расписания и управление сборкой, установкой и тестированием	Input
Готовые КПК	Тестирование КПК	Output	Граница диаграммы	Output
Тестирующий	Персонал производственного отдела		Тестирование КПК	Mechanism
Указание передать КПК на отгрузку	Отслеживание расписания и управление сборкой, установкой и тестированием	Output	Тестирование КПК	Control

2. Туннелируйте и свяжите на верхнем уровне граничные стрелки, если это необходимо. Результат выполнения задания показан на рис. 1.11.

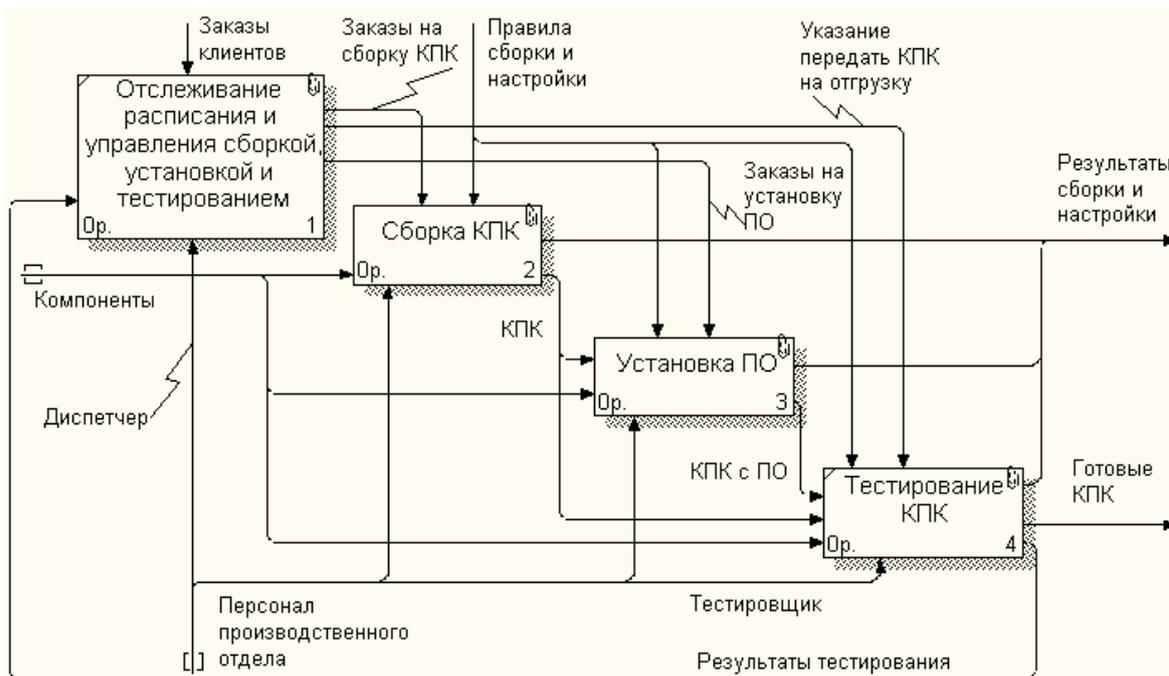


Рис. 1.11. Результат создания диаграмма декомпозиции A2

1.4. Создание диаграммы узлов

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Процесс создания модели работ является итерационным, следовательно, работы могут менять свое расположение в дереве узлов многократно. *Диаграмм деревьев узлов* может быть в модели сколь угодно много, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня. Чтобы не запутаться и проверить способ декомпозиции, следует после каждого изменения создавать *диаграмму дерева узлов* [4].

1. Выберите меню Diagram/Add Node Tree. В первом диалоге гида Node Tree Wizard внесите имя диаграммы, укажите диаграмму корня дерева и количество уровней (рис. 1.12).



Рис. 1.12. Первый диалог гюда Node Tree Wizard

2. По умолчанию нижний уровень декомпозиции показывается в виде списка, остальные *работы* — в виде прямоугольников. Во втором диалоге установите опции, как на рис. 1.13.

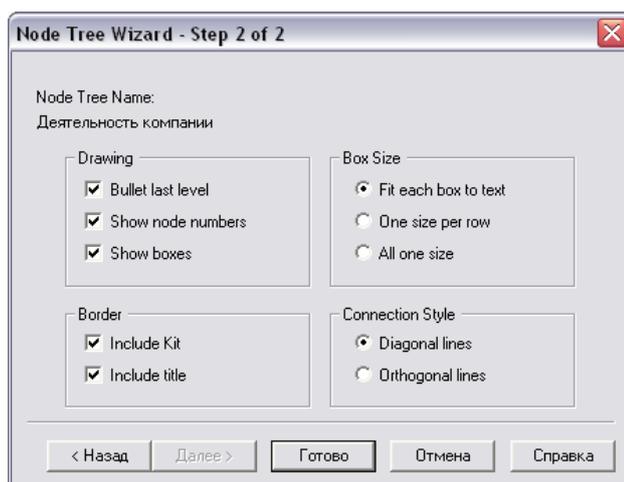


Рис. 1.13. Второй диалог гюда Node Tree Wizard

Щелкните по Finish. Создается диаграмма дерева узлов. Результат на рис. 1.14.



Рис. 1.14. Диаграмма дерева узлов

Диаграмму дерева узлов можно модифицировать. Нижний уровень может быть отображен не в виде списка, а в виде прямоугольников.

Для модификации диаграммы правой кнопкой мыши щелкните по свободному месту, не занятому объектами, выберите меню Node Tree Diagram Properties и на вкладке Style диалога Node Tree Properties выключите опцию Bullet Last Level. Щелкните по ОК. Результат создания диаграммы узлов показан на рис. 1.15.



Рис. 1.15. Результат создания диаграммы узлов

1.5. Создание FEO диаграммы

Диаграммы для экспозиции (FEO) строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения, либо для специальных целей, которые не поддерживаются явно синтаксисом IDEF0.

Диаграммы FEO позволяют нарушить любое синтаксическое правило, поскольку по сути являются просто картинками — копиями стандартных диаграмм и не включаются в анализ синтаксиса [4].

Предположим, что при обсуждении бизнес-процессов возникла необходимость детально рассмотреть взаимодействие работы «Сборка и настройка КПК» с другими работами. Чтобы не портить диаграмму декомпозиции, создайте FEO-диаграмму, на которой будут только стрелки работы «Сборка и настройка КПК».

1. Перейдите на диаграмму декомпозиции A0.
2. Выберите пункт меню Diagram/Add FEO Diagram.
3. В диалоге Add New FEO Diagram выберите тип и внесите имя диаграммы FEO. Щелкните ОК.
4. Для определения диаграммы перейдите в Diagram/Diagram Properties и во вкладке Diagram Text внесите определение.
5. Удалите лишние стрелки на диаграмме FEO. Результат показан на рис. 1.16.

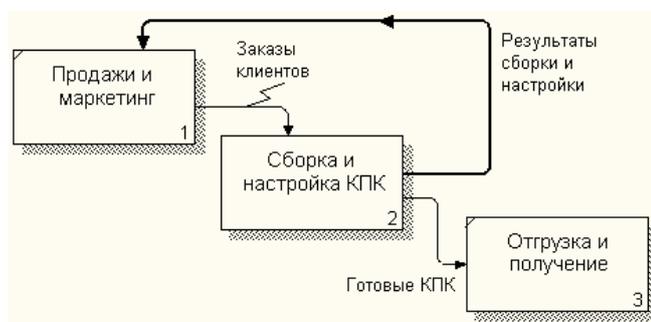


Рис. 1.16. Диаграмма FEO

Для перехода между стандартной диаграммой, деревом узлов и FEO используйте кнопку  на палитре инструментов.

Диаграммы размещаются на рабочем листе с граничными рамками, которые называются каркасом диаграммы.

Каркас содержит заголовок (верхняя часть рамки) и подвал (нижняя часть). Заголовок каркаса используется для отслеживания диаграммы в процессе моделирования. Нижняя часть используется для идентификации и позиционирования в иерархии диаграммы.

Смысл элементов каркаса приведен в табл. 1.5 и 1.6.

Таблица 1.5. Поля заголовка каркаса

Поле	Смысл
Used At	Используется для указания на родительскую <i>работу</i> в случае, если на текущую диаграмму ссылались посредством <i>стрелки</i> вызова
Autor, Date, Rev, Project	Имя создателя диаграммы, дата создания и имя проекта, в рамках которого была создана диаграмма. REV-дата последнего редактирования диаграммы
Notes 123456789 10	Используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии диаграммы) указать число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания
Status	Статус отображает стадию создания диаграммы, отображая все этапы публикации
Working	Новая диаграмма, кардинально обновленная диаграмма или новый автор диаграммы
Draft	Диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению
Recommended	Диаграмма и все ее сопровождающие документы прошли экспертизу. Новых изменений не ожидается
Publication	Диаграмма готова к окончательной печати и публикации
Reader	Имя читателя (эксперта)
Date	Дата прочтения (экспертизы)

Context	Схема расположения <i>работ</i> в диаграмме верхнего уровня. <i>Работа</i> , являющаяся родительской, показана темным прямоугольником, остальные – светлым. На <i>контекстной диаграмме</i> (A-0) показана надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы.
---------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Таблица 1.6. Поля подвала каркаса

Поле	Смысл
Node	Номер узла диаграммы (номер родительской <i>работы</i>)
Title	Имя диаграммы. По умолчанию — имя родительской <i>работы</i>
Number	C-Number, уникальный номер версии диаграммы
Page	Номер страницы, может использоваться как номер страницы при формировании папки

Значения полей каркаса задаются в диалоге Diagram /Diagram Properties.

1.6. Расщепление и слияние моделей

Возможность слияния и расщепления моделей обеспечивает коллективную *работу* над проектом. Так, руководитель проекта может создать декомпозицию верхнего уровня и дать задание аналитикам продолжить декомпозицию каждой ветви дерева в виде отдельных моделей. После окончания *работы* над отдельными ветвями все подмодели могут быть слиты в единую модель. С другой стороны, отдельная ветвь модели может быть отщеплена для использования в качестве независимой модели, для доработки или архивирования [4].

VRwin использует для слияния и разветвления моделей *стрелки* вызова. Для слияния необходимо выполнить следующие условия:

- Обе сливаемые модели должны быть открыты в VRwin.

- Имя модели-источника, которое присоединяют к модели-цели, должно совпадать с именем *стрелки* вызова *работы* в модели-цели.
- *Стрелка* вызова должна исходить из недекомпозируемой *работы* (*работа* должна иметь диагональную черту в левом верхнем углу).
- Имена контекстной *работы* подсоединяемой модели-источника и *работы* на модели-цели, к которой мы подсоединяем модель-источник, должны совпадать.
- Модель-источник должна иметь, по крайней мере, одну диаграмму декомпозиции.

1. Перед этим заданием, сохраните проект на диске в отдельном файле, чтоб потом можно было вернуться к нему. Перейдите на диаграмму A0. Правой кнопкой мыши щелкните по работе «Сборка и настройка КПК» и выберите Split model.

2. В диалоге Split Option внесите имя новой модели «Сборка и настройка КПК», установите опции, как на рисунке, и щелкните по ОК (рис. 1.17).



Рис. 1.17. Диалог Split Option

3. Посмотрите на результат: в Model Explorer появится новая модель, а на диаграмме A0 модели «Деятельность компании» появится стрелка вызова «Сборка и настройка КПК».

4. Создайте в модели «Сборка и настройка КПК» новую стрелку «Неисправные компоненты». На диаграмме A-0 это будет граничная стрелка выхода, на диаграмме A0 – граничная стрелка выхода от работ «Сборка КПК», «Тестирование КПК» и «Установка ПО».

5. Перейдите на диаграмму A0 модели «Деятельность компании».

6. Правой кнопкой мыши щелкните по работе «Сборка и настройка КПК» и выберите Merge model.

7. В диалоге Merge Model включите опцию Cut/Paste entire dictionaries и щелкните по ОК.

Посмотрите на результат. В Model Explorer видно, что две модели слились. Модель «Сборка и настройка КПК» осталась и может быть сохранена в отдельном файле. На диаграмме A0 модели «Деятельность компании» исчезла стрелка вызова «Сборка и настройка КПК». Появилась неразрешимая граничная стрелка «Неисправные компоненты». Направьте эту стрелку к входу работы «Отгрузка и получение».

Удалите созданную командой Split model модель «Сборка и настройка КПК», поскольку она уже не нужна. Если у вас что-либо не получилось при выполнении этого задания, вернитесь к версии файла, сохраненной перед началом этого задания.

1.7. Создание диаграммы IDEF3

Для описания логики взаимодействия информационных потоков более подходит IDEF3, называемая также workflow diagramming, — методология моделирования, использующая графическое описание информационных потоков, взаимоотношений между *процессами* обработки информации и объектов, являющихся частью этих *процессов*. Диаграммы Workflow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием *процесса* и может быть использован для документирования каждой функции.

IDEF3 — это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда *процессы* выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном *процессе*.

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний *процессов* IDEF3 не ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей.

IDEF3 может быть также использован как метод создания *процессов*. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Каждая работа в IDEF3 описывает какой-либо сценарий бизнес-процесса и может являться составляющей другой работы. Поскольку сценарий описывает цель и рамки модели, важно, чтобы работы именовались отглагольным существительным, обозначающим *процесс* действия, или фразой, содержащей такое существительное.

Точка зрения на модель должна быть документирована. Обычно это точка зрения человека, ответственного за работу в целом. Также необходимо документировать цель модели — те вопросы, на которые призвана ответить модель [4].

Диаграмма является основной единицей описания в IDEF3. Важно правильно построить диаграммы, поскольку они предназначены для чтения другими людьми.

Единицы работы — Unit of Work (UOW) — также называемые работами (activity), являются центральными компонентами модели. В IDEF3 работы изображаются прямоугольниками с прямыми углами и имеют имя, выраженное отглагольным существительным, обозначающим *процесс* действия, оди-

ночным или в составе фразы, и номер (идентификатор); другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы.

Связи показывают взаимоотношения работ. Все *связи* в IDEF3 односторонние и могут быть направлены куда угодно, но обычно связи стараются направить слева направо. В IDEF3 различают три типа стрелок, изображающих *связи*:

- Старшая (Precedence) - сплошная линия, связывающая единицы работ (UOW). Рисуются слева направо или сверху вниз. Показывает, что работа-источник должна закончиться прежде, чем работа-цель начнется.
- Отношения (Relational Link) - пунктирная линия, используемая для изображения *связей* между единицами работ (UOW) а также между единицами работ и объектами ссылок.
- Потоки объектов (Object Flow) - стрелка с двумя наконечниками, применяется для описания того факта, что объект используется в двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой.

Для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы, используются **перекрестки** (Junction). Различают перекрестки для слияния (Fan-in Junction) и разветвления стрелок (Fan-out Junction). Перекресток не может использоваться одновременно для слияния и для разветвления. Смысл каждого типа приведен в таблице 1.7 [4].

Объект ссылки в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой. В качестве имени объекта можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть

связаны с единицами работ или перекрестками пунктирными линиями. Официальная спецификация IDEF3 различает три стиля объектов ссылок — безусловные (unconditional), синхронные (synchronous) и асинхронные (asynchronous).

Таблица 1.7. Типы перекрестков

Обозначение	Наименование	Смысл в случае слияния стрелок (Fan-in Junction)	Смысл в случае разветвления стрелок (Fan-out Junction)
	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Synchronous OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

При внесении объектов ссылок помимо имени следует указывать тип объекта ссылки. Типы объектов ссылок приведены в таблице 1.8 [4].

Таблица 1.8. Типы объектов ссылок

Тип объекта ссылки	Цель описания
ОБЪЕСТ	Описывает участие важного объекта в работе
GOTO	Инструмент циклического перехода (в повторяющейся последовательности работ), возможно на текущей диаграмме,

	но не обязательно. Если все работы цикла присутствуют на текущей диаграмме, цикл может также изображаться стрелкой, возвращающейся на стартовую работу. GOTO может ссылаться на перекресток
UOB (Unit of behaviour)	Применяется, когда необходимо подчеркнуть множественное использование какой-либо работы, но без цикла. Например, работа «Контроль качества» может быть использована в <i>процессе</i> «Изготовление изделия» несколько раз, после каждой единичной операции. Обычно этот тип ссылки не используется для моделирования автоматически запускающихся работ
NOTE	Используется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме. NOTE является альтернативой внесению текстового объекта в диаграмму
ELAB (Elaboration)	Используется для усовершенствования графиков или их более детального описания. Обычно употребляется для детального описания разветвления и слияния стрелок на <i>перекрестках</i>

В IDEF3 **декомпозиция** используется для детализации работ. Методология IDEF3 позволяет декомпозировать работу многократно, т. е. работа может иметь множество дочерних работ.

1. Перейдите на диаграмму A2 и декомпозируйте (кнопка ) работу «Сборка КПК». В диалоге Activity Box Count (рис 1.18) установите число работ 4 и нотацию IDEF3.



Рис. 1.18. Выбор нотации IDEF3 в диалоге Activity Box Count

Возникает диаграмма IDEF3, содержащая работы (UOW). Правой кнопкой мыши щелкните по работе, выберите в контекстном меню Name и внесите имя работы «Подготовка компонентов». Затем во вкладке Definition внесите определение «Подготавливаются все компоненты КПК согласно спецификации заказа».

2. Во вкладке UOW внесите свойства работы (табл. 1.9).

Таблица 1.9. Свойства UOW

<i>Objects</i>	Компоненты: процессоры, корпуса, платы, карты, разъемы для карт памяти, интерфейсы, модемы
<i>Facts</i>	Доступные интерфейсы: Wi-Fi, Bluetooth, IrDa, USB
<i>Constrains</i>	Установка некоторого оборудования требует установки дополнительного программного обеспечения

3. Внесите в диаграмму еще 3 работы (кнопка ).

Внесите имена работ:

- Установка процессора;
- Установка оперативной памяти;
- Установка флеш-памяти;
- Установка интерфейса Wi-Fi;
- Установка интерфейса Bluetooth;
- Установка слота для карт памяти.

4. С помощью кнопки  палитры инструментов создайте объект ссылки. Внесите имя объекта внешней ссылки «Компоненты».

Свяжите стрелкой объект ссылки и работу «Подготовка компонент».

5. Свяжите стрелкой работы «Подготовка компонент» (выход) и «Установка процессора». Измените стиль стрелки на Object Flow.

В IDEF3 имя стрелки может отсутствовать, хотя VPwin показывает отсутствие имени как ошибку. Результат создания UOW и объекта ссылки показан на рис. 1.19.

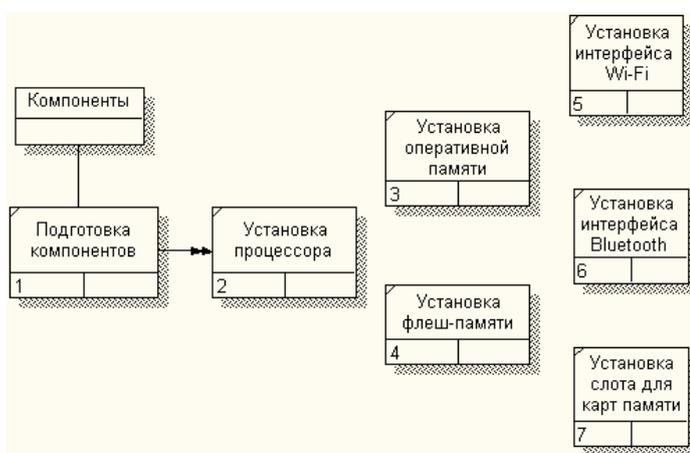


Рис. 1.19. Результат создания UOW и объекта ссылки

6. С помощью кнопки  на палитре инструментов внесите два перекрестка типа «асинхронное и» и два перекрестка типа «асинхронное или», и свяжите работы с перекрестками, как показано на рис. 1.20.

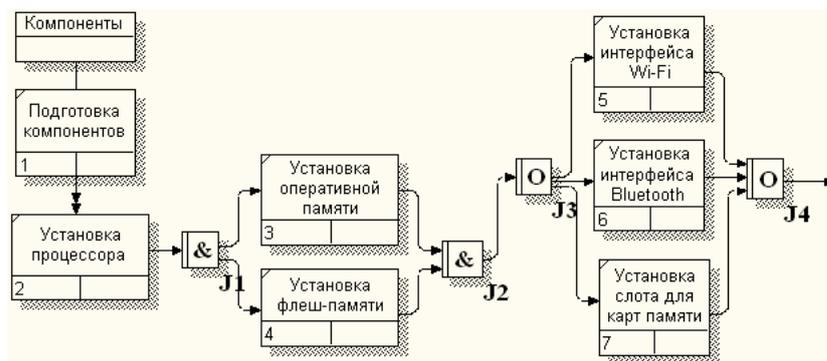


Рис. 1.20. Диаграмма IDEF3 после создания перекрестков

7. Правой кнопкой щелкните по перекрестку для разветвления J3, выберите Name и внесите имя «Компоненты, требуемые в спецификации заказа».

1.8. Создание сценария

1. Перейдите к диаграмме A22.1 – «Сборка КПК».
2. Выберите пункт меню Diagram/Add IDEF3 Scenario. Создайте диаграмму сценария на основе диаграммы IDEF3 «Сборка КПК» (A22.1).
3. Удалите элементы, не входящие в сценарий (рис. 1.21).

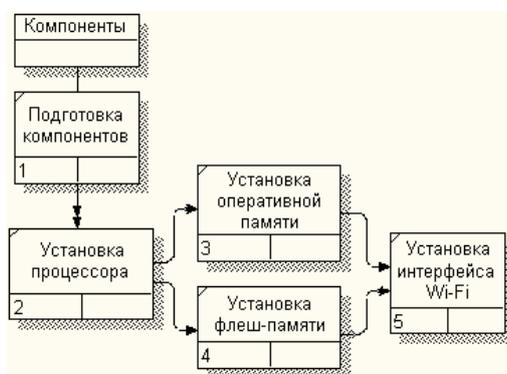


Рис. 1.21. Результат создания диаграммы сценария

1.9. Создание отчетов в VPwin

VPwin имеет мощный инструмент генерации отчетов. Отчеты по модели вызываются из пункта меню Report. Всего имеется семь типов отчетов [4]:

1. Model Report. Включает информацию о контексте модели — имя модели, точку зрения, область, цель, имя автора, дату создания и др.
2. Diagram Report. Отчет по конкретной диаграмме. Включает список объектов (*работ*, стрелок, хранилищ данных, внешних ссылок и т. д.).
3. Diagram Object Report. Наиболее полный отчет по модели. Может включать полный список объектов модели (*работ*, стрелок с указанием их типа и др.) и свойства, определяемые пользователем.

4. Activity Cost Report. Отчет о результатах стоимостного анализа.
5. Arrow Report. Отчет по *стрелкам*. Может содержать информацию из словаря стрелок, информацию о работе-источнике, работе-назначении *стрелки* и информацию о разветвлении и слиянии стрелок.
6. Data Usage Report. Отчет о результатах связывания модели процессов и модели данных.
7. Model Consistency Report. Отчет, содержащий список синтаксических ошибок модели.

1.10. Стоимостной анализ (Activity Based Costing)

Как было указано ранее, обычно сначала строится функциональная модель существующей организации работы — AS-IS. После построения модели AS-IS проводится анализ бизнес-процессов, потоки данных и объектов перенаправляются и улучшаются, в результате строится модель TO-BE. Как правило, строится несколько моделей TO-BE, из которых по какому-либо критерию выбирается наилучшая. Проблема состоит в том, что таких критериев много и непросто определить важнейший. Для того чтобы определить качество созданной модели с точки зрения эффективности бизнес-процессов, необходима система метрики, т. е. качество следует оценивать количественно [4].

ВРwin предоставляет аналитику два инструмента для оценки модели — *стоимостный анализ*, основанный на работах (Activity Based Costing, ABC), и свойства, определяемые пользователем (User Defined Properties, UDP). *Функциональное оценивание* – это технология выявления и исследования стоимости выполнения той или иной функции. Исходными данными для *функционального оценивания* являются затраты на ресурсы (материалы, персонал и т.д.). В сравнении с традиционными способами оценки затрат, при применении которых часто недооценивается продукция, производимая в незначительном объе-

ме, и переоценивается массовый выпуск, ABC обеспечивает более точный метод расчета стоимости производства продукции, основанный на стоимости выполнения всех технологических операций, выполняемых при ее выпуске.

Стоимостный анализ представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость *процесса*.

Стоимостный анализ основан на модели работ, потому что количественная оценка невозможна без детального понимания функциональности предприятия. Обычно ABC применяется для того, чтобы понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Reengineering, BPR). С помощью *стоимостного анализа* можно решить такие задачи, как определение действительной стоимости производства продукта, определение действительной стоимости поддержки клиента, идентификация наиболее дорогостоящих работ, обеспечение менеджеров финансовой мерой предлагаемых изменений и т.д.

ABC-анализ может проводиться только тогда, когда модель работы последовательная, корректная, полная и стабильная, другими словами, когда создание модели работы закончено.

ABC включает следующие основные понятия [4, 7]:

- *Объект затрат* — причина, по которой работа выполняется, обычно основной выход работы. Стоимость работ есть суммарная стоимость *объектов затрат*;
- *Двигатель затрат* — характеристики входов и управлений работы, которые влияют на то, как выполняется и как долго длится работа;
- *Центры затрат*, которые можно трактовать как статьи расхода.

1. В диалоге Model Properties (вызывается из меню Model/Model Properties) во вкладке ABC Unit (рис. 1.22) установите единицы измерения денег и времени: рубли и дни.

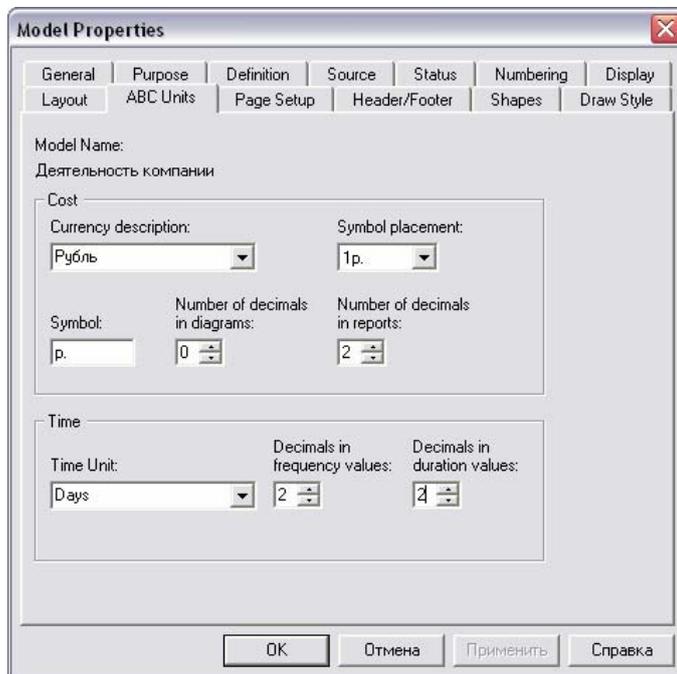


Рис. 1.22. Вкладка ABC Units диалога Model Properties

2. Перейдите в Dictionary/Cost Center (рис. 1.23) и в диалоге Cost Center Dictionary внесите название и определение центров затрат (табл. 1.10).

Name	Definition
Управление	Затраты на управление, связанные с составлением графика работ
Рабочая сила	Затраты на оплату рабочих, занятых сборкой и на-стройкой КПК
Компоненты	Затраты на закупку компонентов

Рис. 1.23. Диалог Cost Center Dictionary

Таблица 1.10. Центры затрат ABC

Центры за-трат	Определение
Управление	Затраты на управление, связанные с составлением графика работ, формированием партий КПК, контролем над сборкой и настройкой

Рабочая сила	Затраты на оплату рабочих, занятых сборкой и настройкой КПК
Компоненты	Затраты на закупку компонентов

3. Для отображения стоимости каждой работы в нижнем левом углу прямоугольника перейдите в меню Model/Model Properties и во вкладке Display диалога Model Properties включите опцию ABC Data (рис. 1.24). Для отображения частоты или продолжительности работы переключите радиокнопки в группе ABC Unit.

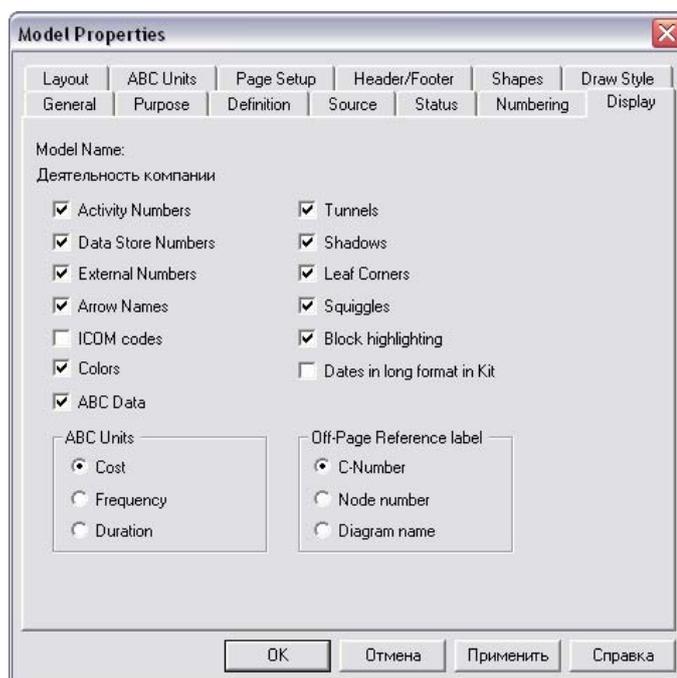


Рис. 1.24. Вкладка Display диалога Model Properties

4. Для работ на диаграмме A2 внесите параметры ABC (табл. 1.11). Для назначения стоимости работе следует щелкнуть по ней правой кнопкой мыши и выбрать в контекстном меню Cost (рис. 1.25).

Таблица. 1.11. Стоимость работ на диаграмме A2

<i>Activity Name</i>	<i>Cost Center</i>	<i>Cost Center Cost, руб.</i>	<i>Frequency</i>	<i>Duration, день</i>
Отслеживание расписанием и управление сборкой, установкой и тести-	Управление	500,00	1,00	1,00

рованием				
Сборка КПК	Рабочая сила	140,00	20,00	1,00
	Компоненты	28000,00		
Установка ПО	Рабочая сила	100,00	12,00	1,00
	ПО	16000,00		
Тестирование КПК	Рабочая сила	60,00	32,00	1,00

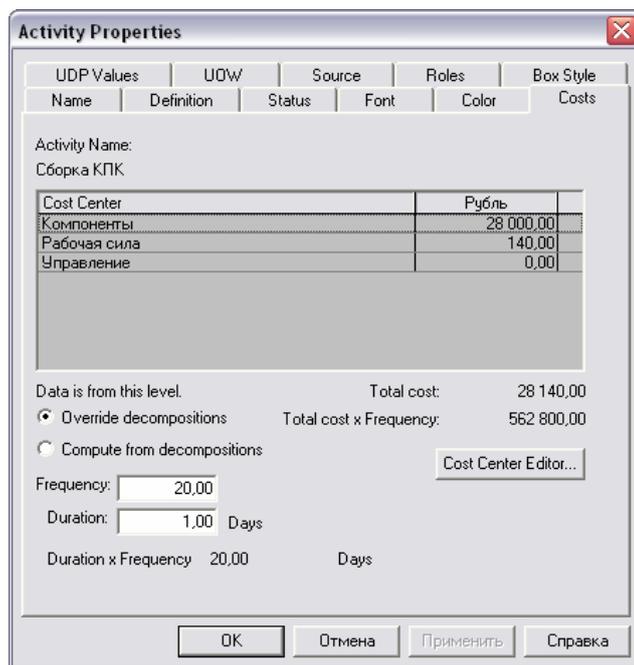


Рис. 1.25. Вкладка Cost диалога Activity Properties

Посмотрите результат – стоимость работы верхнего уровня, диаграммы А2 (рис. 1.26).



Рис. 1.26. Отображение стоимости в нижнем левом углу прямоугольника работы

5. Сгенерируйте отчет Activity Cost Report (рис. 1.27 – 1.28).

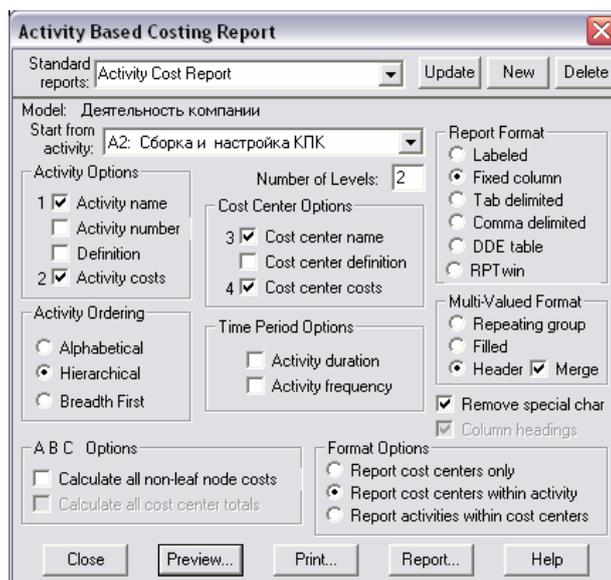


Рис. 1.27. Диалог Activity Based Costing Report

Activity Name	Activity Cost (Рубль)	Cost Center	Cost Center Cost (Рубль)
Сборка и настройка КПК	758 420,00	Компоненты	752 000,00
		Рабочая сила	5 920,00
		Управление	500,00
Отслеживание расписания и управления сборкой, установкой и тестированием	500,00	Управление	500,00
Сборка КПК	28 140,00	Компоненты	28 000,00
		Рабочая сила	140,00
Установка ПО	16 100,00	Компоненты	16 000,00
		Рабочая сила	100,00
Тестирование КПК	60,00	Рабочая сила	60,00

Рис. 1.28. Отчет Activity Cost Report

Этот достаточно упрощенный принцип подсчета справедлив, если работы выполняются последовательно. Встроенные возможности VRwin позволяют разрабатывать упрощенные модели стоимости, которые, тем не менее, оказываются чрезвычайно полезными при предварительной оценке затрат. Если схема выполнения более сложная, можно отказаться от подсчета и задать ито-

говые суммы для каждой работы вручную (Override Decompositions). В этом случае результаты расчетов с нижних уровней декомпозиции будут игнорироваться, и при расчетах на верхних уровнях будет учитываться сумма, заданная вручную. На любом уровне результаты расчетов сохраняются независимо от выбранного режима, поэтому при выключении опции Override Decompositions расчет снизу вверх производится обычным образом.

1.11. Использование категорий UDP

ABC позволяет оценить стоимостные и временные характеристики системы. Если стоимостных показателей недостаточно, имеется возможность внесения собственных метрик — свойств, определенных пользователем (User Defined Properties, UDP). UDP позволяют провести дополнительный анализ, хотя и без суммирующих подсчетов [4].

1. Перейдите в меню Dictionary/UDP Keywords и в диалоге UDP Keywords List внесите ключевые слова UDP (рис. 1.29):

- Расход ресурсов;
- Документация;
- Информационная система.

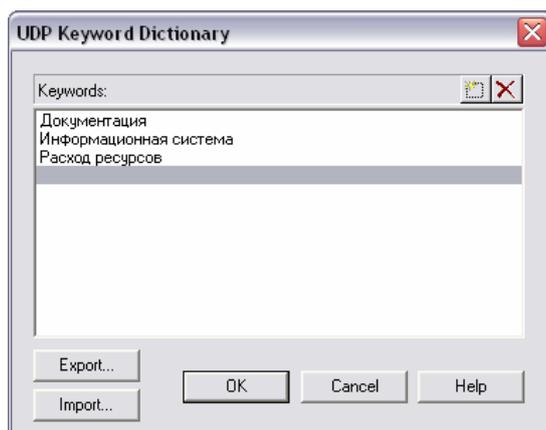


Рис. 1.29. Словарь ключевых слов UDP

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

3. Для UDP «Приложение» в поле UDP Datatype выберите Text List (Multiple selections), при этом появится возможность в поле Value задать список значений. Внесите значение «Модуль оформления заказов» (рис. 1.30).

Name	UDP Datatype	Value	Keyword
Приложение	Text List (Multiple selections)	Edit List	Информационная система
Дополнительная документация	Command List	Edit List	Документация
История изменения	Paragraph Text		Документация
Загрязнение окружающей среды	Text List (Single selection)	Edit List	
Расход электроэнергии	Real Number		Расход ресурсов
	Text		

Рис. 1.30. Словарь UDP

Затем внесите другие значения в соответствии с табл. 1.12. Для подключения к UDP ключевого слова перейдите к полю Keywords и щелкните по полю выбора.

Таблица 1.12. Наименование и свойства UDP

Наименование UDP	Тип	Значение	Ключевое слово
Приложения	Text List (Multiple Selections)	Модуль оформления заказов. Модуль создания и контроля расписания выполнения работ. Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей. Модуль процедур установки ПО.	Информационная система
Дополнительная документация	Command List	Winword.exe sample1.doc Winword.exe sample2.doc Powerpnt.exe sample3.ppt	Документация
История изменения	Paragraph Text		Документация
Загрязнение окружающей	Text List (Single Se-	Очень высокое Высокое	

среды	lections)	Среднее Низкое	
Расход электроэнергии	Real Number		Расход ресурсов

4. Для назначения UDP работе следует щелкнуть по ней правой кнопкой мыши и выбрать в контекстном меню UDP. Появляется вкладка UDP Values диалога Activity Properties (рис. 1.31).

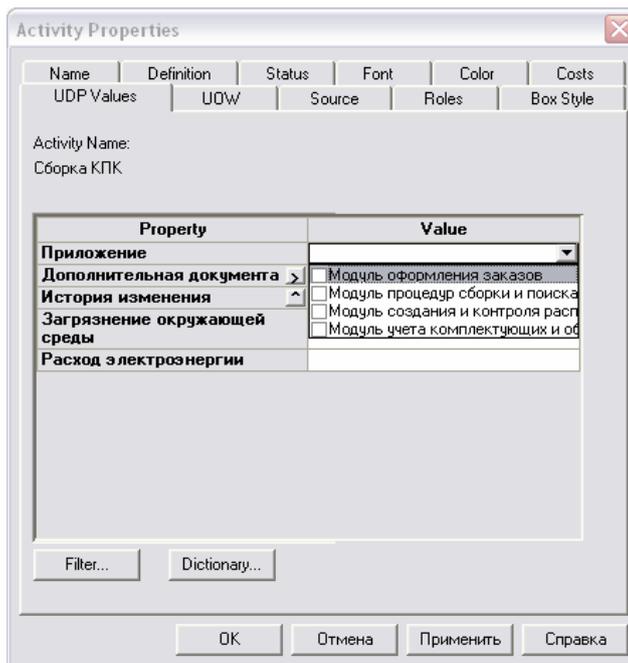


Рис. 1.31. Вкладка UDP Values диалога Activity Properties

Внесите значения UDP для работ диаграммы A2 (табл. 1.13).

Таблица 1.13. Значения UDP

Activity Name	Дополнительная документация	Приложения	История изменения	Расход электроэнергии	Загрязнение окружающей среды
Сборка КПК		Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей.		20,00	Среднее

Установка ПО		Модуль процедур установки ПО.		25,00	Среднее
Тестирование КПК		Модуль учета комплектующих и оборудования. Модуль процедур сборки и поиска неисправностей. Модуль процедур установки ПО.		40,00	Среднее
Отслеживание расписанием и управление сборкой, установкой и тестированием	Win-word.exe sample2.doc	Модуль создания и контроля расписания выполнения работ	История изменения спецификаций	10,00	Низкое

5. После внесения UDP типа Command или Command List щелчок по кнопке  приведет к запуску приложения.

6. В диалоге Activity Properties щелкните по кнопке Filter. В появившемся диалоге Diagram object UDP filter (рис. 1.32) отключите ключевыми словами «Информационная система». Щелкните по ОК. В результате в диалоге Activity Properties не будет отображаться UDP с ключевыми словами «Информационная система».

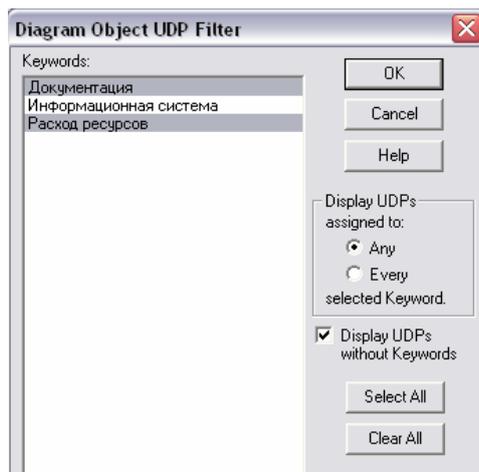


Рис. 1.32. Диалог *Diagram object UDP filter*

Отметим, что свойства UDP можно присвоить не только работам, но и стрелкам [7].

7. Посмотрите отчет по UDP. Меню Tools/Report/Diagram Object Report. Выберите опции отчета:

- Start from Activity: A2. Сборка и настройка КПК;
- Number of Level: 2;
- User Defined Properties: Расход электроэнергии.

1.12. Расщепление модели

1. Сохраните рабочий файл. С помощью команды File/Save As создайте новую версию.

2. Перейдите на диаграмму A0 и щелкните правой кнопкой мыши по работе «Отгрузка и получение». В контекстном меню выберите Split Model.

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

Обратите внимание, что у работы «Отгрузка и получение» появилась стрелка вызова. VPrwin создал также новую модель «Отгрузка и получение».

3. Внесите свойства новой модели «Отгрузка и получение»:

- Time Frame: AS-IS;
- Purpose: Документировать работу «Отгрузка и получение»;
- Viewpoint: Начальник отдела;
- Definition: Модель создается для иллюстрации возможностей BPwin по расщеплению и слиянию моделей;

по расщеплению и слиянию моделей;

- Score: Работы по получению комплектующих и отправке готовой продукции.

4. Декомпозируйте контекстную работу на 3 работы (табл. 1.14).

Таблица. 1.14. Декомпозиция работы «Отгрузка и получение»

<i>Activity Name</i>	<i>Activity Definition</i>
Получить комплектующие	Физически получить комплектующие и сделать соответствующие записи в информационной системе
Доставить комплектующие	Доставить комплектующие сборщикам и настройщикам
Отгрузить товар и возврат	Отгрузить товар клиентам и неисправные компоненты (возврат) поставщикам

5. Свяжите граничные стрелки, как показано на рис. 1.33.

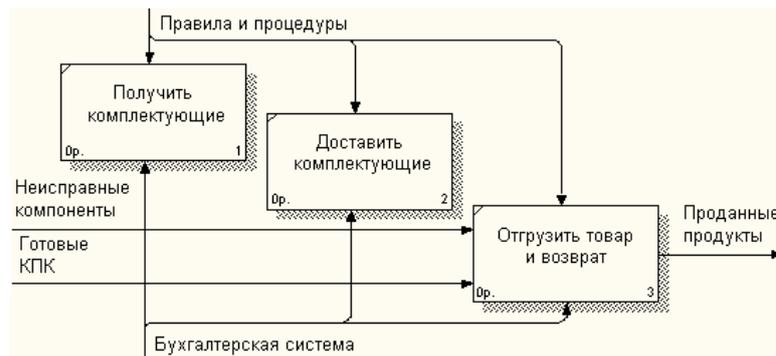


Рис. 1.33. Внутренние стрелки на декомпозиции работы «Отгрузка и получение»

6. Внесите следующие внутренние и граничные стрелки (табл. 1.15).

Таблица. 1.15. Внутренние и граничные стрелки на декомпозиции работы «Отгрузка и получение»

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

7. Туннелируйте граничные стрелки (Resolve Border Arrow). Результат Расщепления модели показан на рис. 1.34.

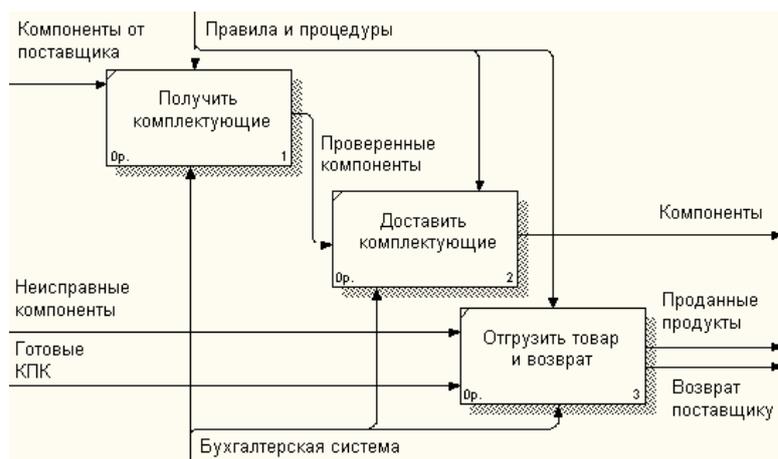


Рис. 1.34. Результат расщепления модели

1.13. Слияние расщепленной модели с исходной моделью

1. Сохраните рабочий файл. С помощью команды File/Save As создайте новую версию.

2. Перейдите в модель «Деятельность компании». На диаграмме A0 щелкните правой кнопкой мыши по работе «Отгрузка и получение». В контекстном

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

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

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

3. На диаграмме A0 туннелируйте и свяжите стрелки согласно рис. 1.35.

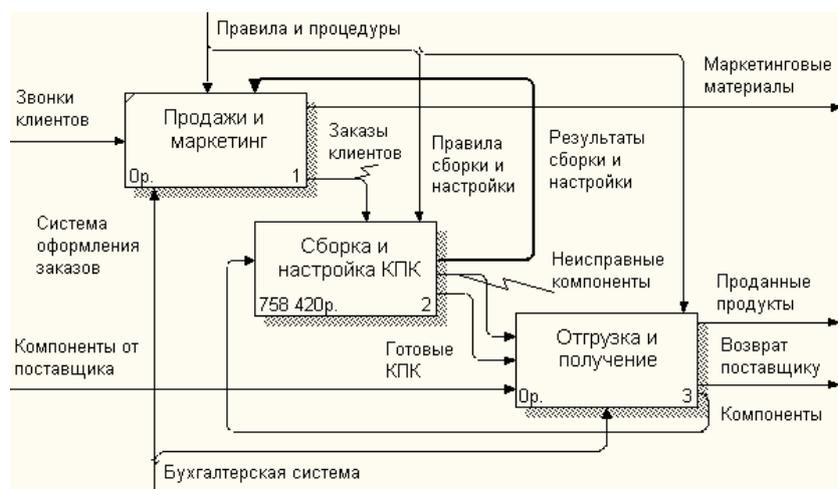


Рис. 1.35. Результат выполнения туннелирования стрелок

4. Удалите созданную командой Split model модель «Отгрузка и получение», поскольку она уже не нужна (закройте рабочее окно этой модели без сохранения). Если у вас что-либо не получилось при выполнении этого задания, вернитесь к версии файла, сохраненной перед началом этого задания.

1.14. Создание модели ТО-ВЕ

Модель ТО-ВЕ создается на основе анализа модели AS-IS. Анализ может проводиться как по формальным признакам (отсутствие выходов или управлений у работ, отсутствие обратных связей и т.д.), так и по неформальным – на основе знаний предметной области [7].

Допустим, в результате анализа принимается решение реорганизовать функции производства и тестирования КПК и оставить функциональности «Продажа и маркетинг» и «Отгрузка и получение» пока без изменений.

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

Работа «Сборка и настройка КПК» должна быть реорганизована и названа «Производство продукта». Будут созданы работы «Разработать конфигурацию», «Планировать производство» и «Собрать продукт».

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

Функции диспетчера в работе «Сборка и настройка КПК» должны быть заменены на функции планировщика.

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

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

1.14.1. Расщепление и модификация модели

1. Измените свойства модели «Деятельность компании»:

- Model name: Предлагаемая модель компании;
- Time Frame: ТО-ВЕ;
- Purpose: Документировать предлагаемые изменения бизнес-процессов компании.

2. Переименуйте работу «Сборка и настройка КПК» в «Производство продукта». Расщепите эту работу в модель с тем же названием.

3. Модифицируйте отщепленную модель. Переместите работу «Тестирование КПК» с диаграммы А0 «Производство продукта» на диаграмму А2.1 «Сборка КПК».

4. Переименуйте работу «Сборка КПК» на диаграмме А0 в «Сборка продукта».

5. Удалите работу «Установка ПО».

6. Переименуйте стрелку «Заказы на КПК» в «Заказы на изготовление».

7. Переименуйте «Отслеживание расписания и управления сборкой, установкой и тестированием» в «Планирование производства».

8. Создайте работу «Разработать конфигурацию». Если появится диалог Continue with merge? выберите Cancel.

9. Создайте ветвь стрелки «Персонал производственного отдела» и направьте как механизм к работе «Разработать конфигурацию». Назовите ее «Дизайнер».

10. Создайте стрелку «Стандарты на продукцию» и направьте ее от выхода «Разработать конфигурацию» к границе диаграммы. Туннелируйте эту стрелку. Создайте ветвь этой стрелки, идущую к управлению работы «Планирование производства» и назовите ее «Список необходимых компонентов».

11. Удалите стрелку «Правила сборки и настройки». Создайте ветвь стрелки «Стандарты на продукцию», идущую к управлению работы «Сборка продукта» и назовите ее «Правила сборки и настройки».
12. Переименуйте стрелку «Диспетчер» в «Планировщик производства».
13. Добавьте стрелку «Прогноз продаж» как граничную управляющую к работе «Планирование производства».
14. Добавьте стрелку «Информация от поставщика» как граничную управляющую к работе «Планирование производства».
15. Добавьте стрелку «Заказ поставщику» как граничную стрелку выхода из работы «Планирование производства».
16. Туннелируйте эти стрелки (Resolve Border Arrow).
17. На диаграмме А-0 туннелируйте стрелку «Готовые КПК» и свяжите ее на диаграмме А0 с выходом работы «Сборка продукта».

Результат расщепления и модификации модели приведен на рис. 1.36 и 1.37.

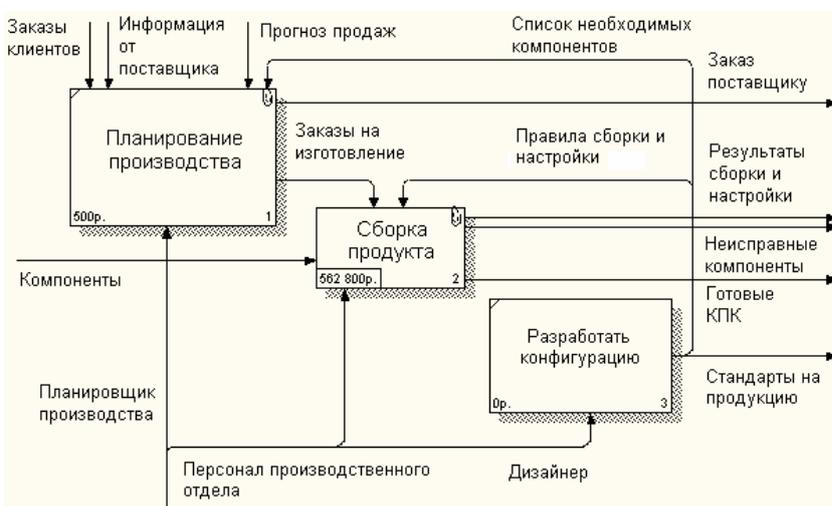


Рис. 1.36. Результат расщепления и модификации модели – диаграмма А0

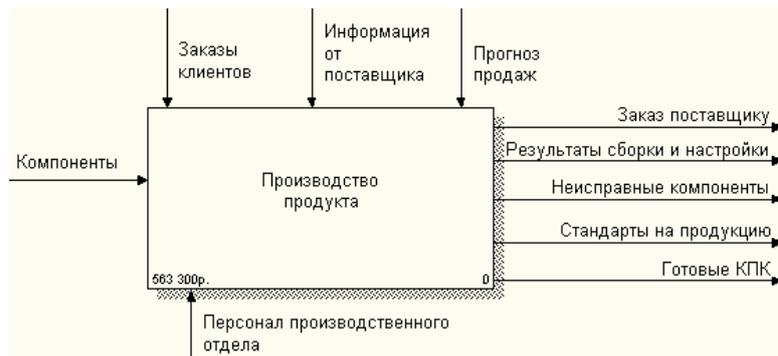


Рис. 1.37. Результат расщепления и модификации модели – диаграмма А-0

1.14.2. Слияние модели

1. Перейдите к работе «*Производство продукта*» в модели «*Деятельность компании*». Щелкните правой кнопкой мыши по работе. В контекстном меню выберите Merge Model. В появившемся диалоге Merge Model установите опцию Cut/Paste entire dictionaries, опцию Overwrite existing fields и щелкните по ОК.

2. На диаграмме А0 туннелируйте стрелки «*Информация от поставщика*» и «*Заказ поставщику*».

3. Направьте стрелку «*Прогноз продаж*» с выхода «*Продажи и маркетинг*» на управление «*Производство продукта*».

4. Направьте стрелку «*Стандарты на продукцию*» с выхода «*Производство продукта*» на управление «*Продажи и маркетинг*».

5. Удалите ветвь стрелки управления «*Правила и процедуры*» для работы «*Производство продукта*».

6. Закройте модель «*Производство продукта*».

Результат слияния и модификации модели приведен на рис. 1.38 и 1.39.

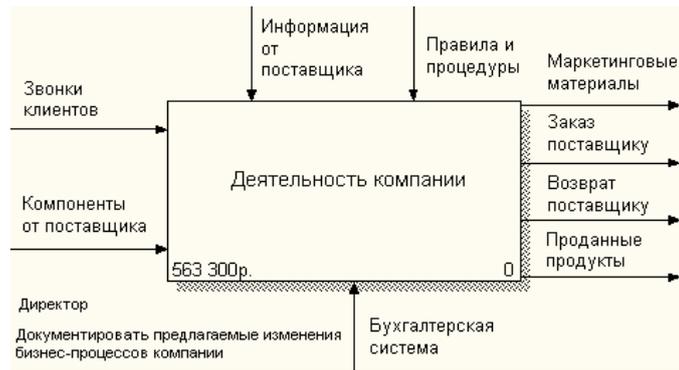


Рис. 1.38. Результат слияния и модификации модели – диаграмма А-0

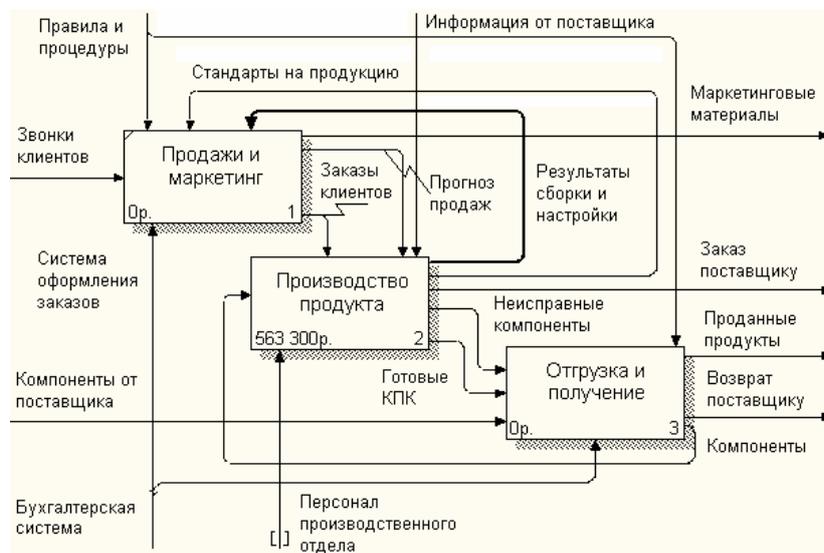


Рис. 1.39. Результат слияния и модификации модели – диаграмма А0

1.14.3. Использование Model Explorer для реорганизации дерева декомпозиции

Существуют причины, по которым работа «Разработать конфигурацию» должна быть на верхнем уровне, на диаграмме А0. Действительно, дизайнер разрабатывает стандарты на продукцию, включая правила сборки и настройки, и список необходимых для закупки компонентов. Тем временем дизайнер управляет производством продукта в целом, кроме того, управляет работой «Продажи и маркетинг».

Было бы логично перенести эту работу на уровень выше.

Используя возможности Model Explorer, перенесите работу «Разработать конфигурацию» с диаграммы A2 «Производство продукта» на диаграмму A0.

Разрешите и перенаправьте стрелки согласно рис. 1.40 и 1.41.

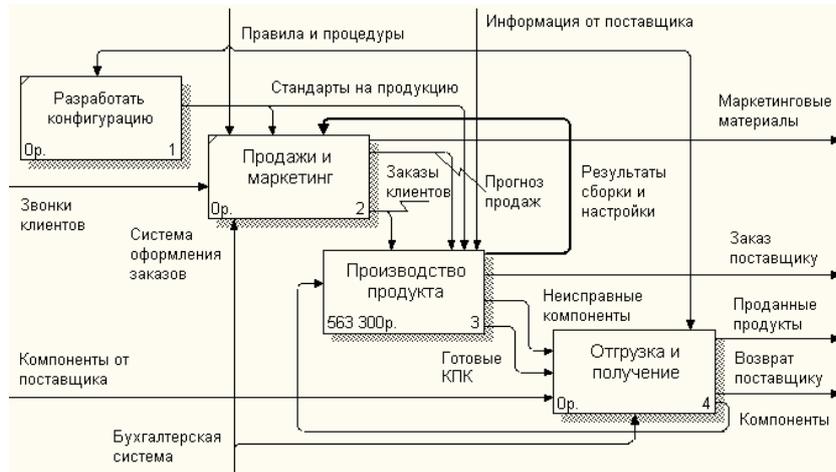


Рис. 1.40. Результат реорганизации дерева декомпозиции – диаграмма A0

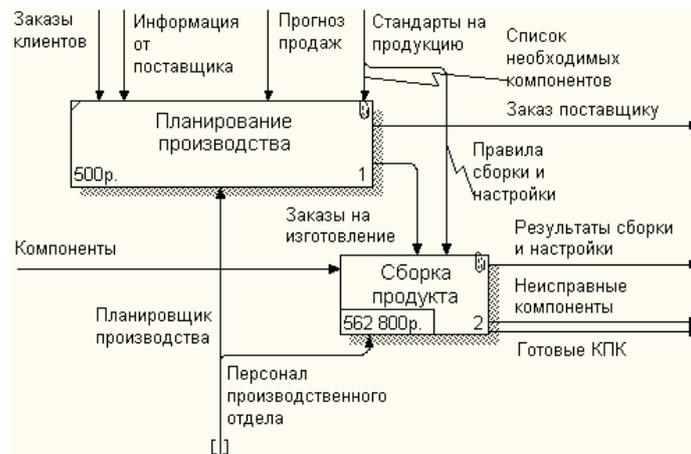


Рис. 1.41. Результат реорганизации дерева декомпозиции – диаграмма A2

1.14.4. Модификация диаграммы IDEF3 «Сборка продукта»

Поскольку мы удалили работу «Установка ПО» из диаграммы, то добавим в модель IDEF3 работы «Сборка продукта» необходимые работы по установке программного обеспечения.

Так же, теперь в работу «Сборка продукта» выключена работа «Тестирование КПК».

Тестирование начинается после окончания процесса сборки КПК и окончания процесса установки программного обеспечения. Если КПК неисправен, в процессе тестирования у него заменят компоненты. Информация о неисправных компонентах может быть направлена на работу «Подготовка компонентов». Такая информация может помочь более тщательно подготавливать компоненты к сборке. Результатом процесса тестирования являются заказанные КПК и неисправные компоненты.

Модифицируйте диаграмму IDEF3 «Сборка продукта» в соответствии приведенной информацией. Результат приведен на рис. 1.42.

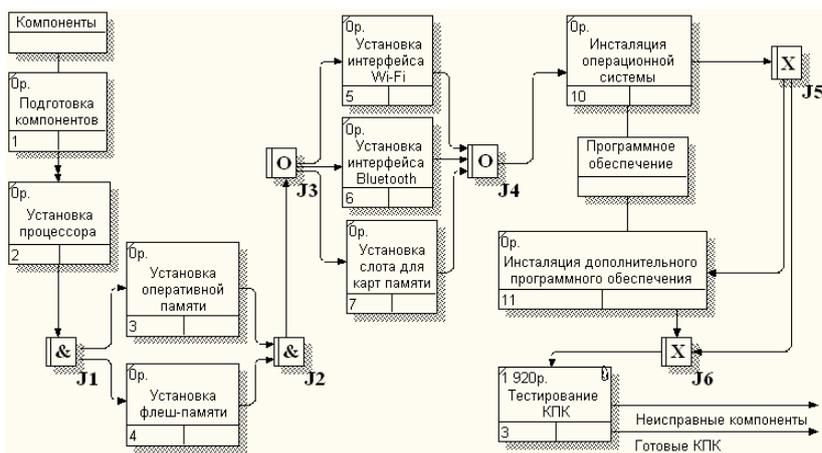


Рис. 1.42. Результат модификации диаграммы IDEF3 – диаграмма А32.1

1.14.5. Декомпозиция работы «Продажи и маркетинг»

Работа по продажам и маркетингу заключается в ответах на телефонные заявки клиентов, предоставлении клиентам информации о ценах, оформлении заказов, внесении заказов в информационную систему и исследовании рынка.

На основе этой информации декомпозируйте работу «Продажи и маркетинга» (IDEF0).

Создайте следующие работы:

- Предоставление информации о ценах;
- Оформление заказов;
- Исследование рынка.

Результат декомпозиции представлен на рис. 1.43.

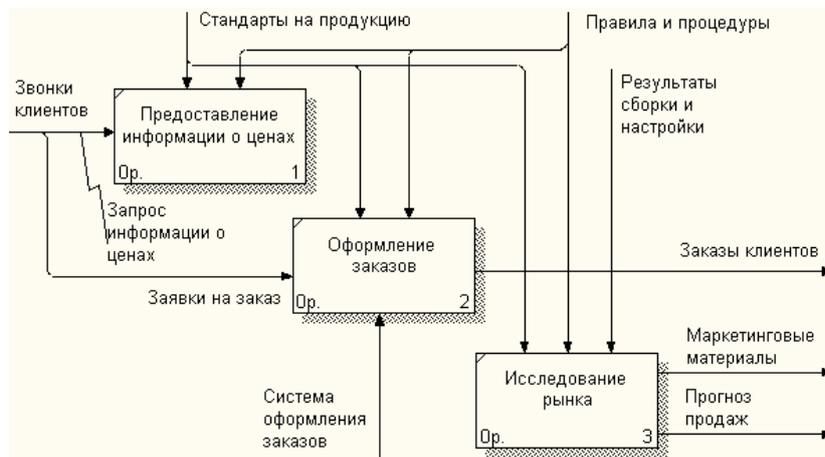


Рис. 1.43. Результат декомпозиции – диаграмма A2

1.15. Создание диаграммы DFD

Диаграммы потоков данных (Data Flow Diagramming) являются основным средством моделирования функциональных требований к проектируемой системе. Требования представляются в виде иерархии *процессов*, связанных потоками данных. Диаграммы потоков данных показывают, как каждый *процесс* преобразует свои входные данные в выходные, и выявляют отношения между этими *процессами*. DFD-диаграммы успешно используются как дополнение к модели IDEF0 для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет моделируемую систему как сеть связанных работ. Основные компоненты DFD – *процессы* или работы, *внешние сущности*, потоки данных, накопители данных (хранилища) [4].

В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются

от одной работы к другой. Это представление потоков совместно с *хранилищами данных* и *внешними сущностями* делает модели DFD более похожими на физические характеристики системы — движение объектов, хранение объектов, поставка и распространение объектов.

В отличие от IDEF0, где система рассматривается как взаимосвязанные работы, DFD рассматривает систему как совокупность предметов. Контекстная диаграмма часто включает работы и внешние ссылки. Работы обычно именуется по названию системы. Включение внешних ссылок в контекстную диаграмму не отменяет требования методологии четко определить цель, область и единую точку зрения на моделируемую систему.

В DFD работы (*процессы*) представляют собой функции системы, преобразующие входы в выходы. Хотя работы изображаются прямоугольниками со скругленными углами, смысл их совпадает со смыслом работ IDEF0 и IDEF3. Так же, как *процессы* IDEF3, они имеют входы и выходы, но не поддерживают управления и механизмы, как IDEF0.

Внешние сущности изображают входы в систему и/или выходы из системы. *Внешние сущности* изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы. Одна *внешняя сущность* может быть использована многократно на одной или нескольких диаграммах. Обычно такой прием используют, чтобы не рисовать слишком длинных и запутанных стрелок.

Потоки работ изображаются стрелками и описывают движение объектов из одной части системы в другую. Поскольку в DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, стрелки могут подходить и выходить из любой грани прямоугольника работы. В DFD также применяются двунаправленные стрелки для описания диалогов типа «команда-ответ» между работами, между работой и *внешней сущностью* и между *внешними сущностями*.

В отличие от стрелок, описывающих объекты в движении, *хранилища данных* изображают объекты в покое.

В материальных системах *хранилища данных* изображаются там, где объекты ожидают обработки, например в *очереди*. В системах обработки информации *хранилища данных* являются механизмом, который позволяет сохранить данные для последующих *процессов*.

В DFD стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя.

Диаграммы DFD могут быть построены с использованием традиционного структурного анализа, подобно тому, как строятся диаграммы IDEF0 [4].

При оформлении заказа можно проверить, существует ли такой клиент в базе данных и, если не существует, внести его в базу данных и затем оформить заказ. Оформление заказа начинается со звонка клиента. В процессе оформления заказа база данных клиентов может просматриваться и редактироваться. Заказ должен включать как информацию о клиенте, так и информацию о заказанных продуктах. Оформление заказа подразумевает чтение и запись информации о прочих заказах [7].

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

1. Декомпозируйте работу «*Оформление заказов*» на диаграмме A2.
2. В диалоге Activity Box Count выберите количество работ 2 и нотацию DFD (рис. 1.44).

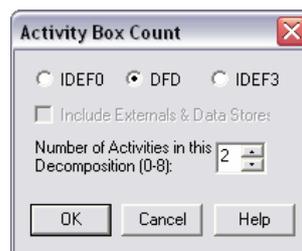


Рис. 1.44. Выбор нотации DFD

3. Внесите в новую диаграмму DFD A22 имена работ:
 - Проверка и внесение клиента;
 - Внесение заказа.
4. Используя кнопку  на палитре инструментов, внесите хранилища данных:
 - Список клиента;
 - Список продуктов;
 - Список заказов.
5. Удалите граничные стрелки с диаграммы DFD A22.
6. Используя кнопку  на палитре инструментов, внесите внешнюю ссылку:
 - Звонки клиентов.
7. Создайте внутренние ссылки согласно рис. 1.45. При именовании стрелок используйте словарь.

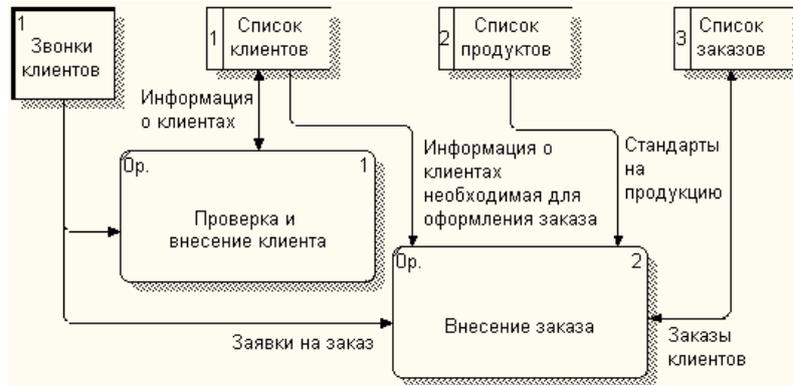


Рис. 1.45. Диаграмма A22

8. Обратите внимание, что стрелки «Информация о клиентах» и «Заказы клиентов» двунаправленные. Для того чтобы сделать стрелку двунаправленной, выберите в контекстном меню пункт Style и во вкладке Style выберите опцию Bidirectional.

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

Вопросы

- 1.1. Что определяет контекстная диаграмма?
- 1.2. Какие основные понятия используются при создании функциональной диаграммы IDEF0?
- 1.3. Укажите, с какой целью строятся диаграммы для экспозиции (FEO).
- 1.4. Что означает появление "туннелей" на диаграмме?
- 1.5. Укажите, чему должна соответствовать точка зрения.
- 1.6. Укажите, что показывает диаграмма дерева узлов.
- 1.7. Какие стрелки называются граничными?
- 1.8. Какие стрелки называются стрелками механизма (Mechanism)?
- 1.9. Укажите, что входит в определение контекста модели.
- 1.10. Укажите основные понятия ABC-анализа.
- 1.11. Какие основные понятия используются при создании функциональной диаграммы IDEF0?
- 1.12. Укажите, какая диаграмма рассматривает систему как совокупность предметов.
- 1.13. Какое назначение имеет стоимостный анализ?
- 1.14. Укажите основные компоненты диаграммы потоков данных.
- 1.15. Укажите основные элементы имитационного моделирования.
- 1.16. Какой вариант правильно описывает цифрами последовательность этапов ABC-анализа?
- 1.17. Укажите, для чего в диаграммах IDEF3 используются перекрестки.
- 1.18. Укажите модели, учитывающие время выполнения функций.

Глава 2. Моделирование информационного обеспечения средствами ERwin

2.1. Отображение модели данных в инструментальном средстве ERwin

ERwin имеет два уровня представления модели – логический и физический.

Логический уровень — это абстрактный взгляд на данные, когда данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире. Объекты модели, представляемые на логическом уровне, называются *сущностями* и *атрибутами*. Логическая модель данных разрабатывается на основе существующих моделей данных (например, реляционной), но никак не связана с конкретной реализацией системы управления базой данных (СУБД) и прочих физических условий реализации. Она может быть построена на основе другой логической модели, например, на основе модели потоков данных или процессов [4, 8].

Логическая модель данных является источником информации для фазы физического проектирования. Она предоставляет разработчику физической модели данных средства проведения всестороннего анализа различных аспектов работы с данными, что имеет исключительное значение для выбора действительно эффективного проектного решения.

Физическая модель данных, напротив, зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация обо всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей [8].

На физическом уровне объекты БД должны называться так, как того требуют ограничения СУБД. На логическом уровне можно этим объектам дать синонимы — имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов. Такое соответствие позволяет лучше документировать модель и дает возможность обсуждать структуру данных с экспертами предметной области.

После описания логической модели проектировщик может выбрать необходимую СУБД, и ERwin автоматически создаст соответствующую физическую модель. На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL-скрипт. Этот процесс называется *прямым проектированием* (Forward Engineering). Тем самым достигается масштабируемость — создав одну *логическую модель данных*, можно сгенерировать физические модели под любую поддерживаемую ERwin СУБД. С другой стороны, ERwin способен по содержимому системного каталога или SQL-скрипту воссоздать *физическую и логическую модель данных* (Reverse Engineering). На основе полученной *логической модели данных* можно сгенерировать физическую модель для другой СУБД и затем создать ее системный каталог. Следовательно, ERwin позволяет решить задачу по переносу структуры данных с одного сервера на другой. Если в логической модели не имеет значения, какой конкретно тип данных имеет *атрибут*, то в физической модели важно описать всю информацию о конкретных физических объектах — таблицах, колонках, индексах, процедурах и т.д. [8].

Интерфейс ERwin выполнен в стиле Windows-приложений, достаточно прост и интуитивно понятен (рис. 2.1).

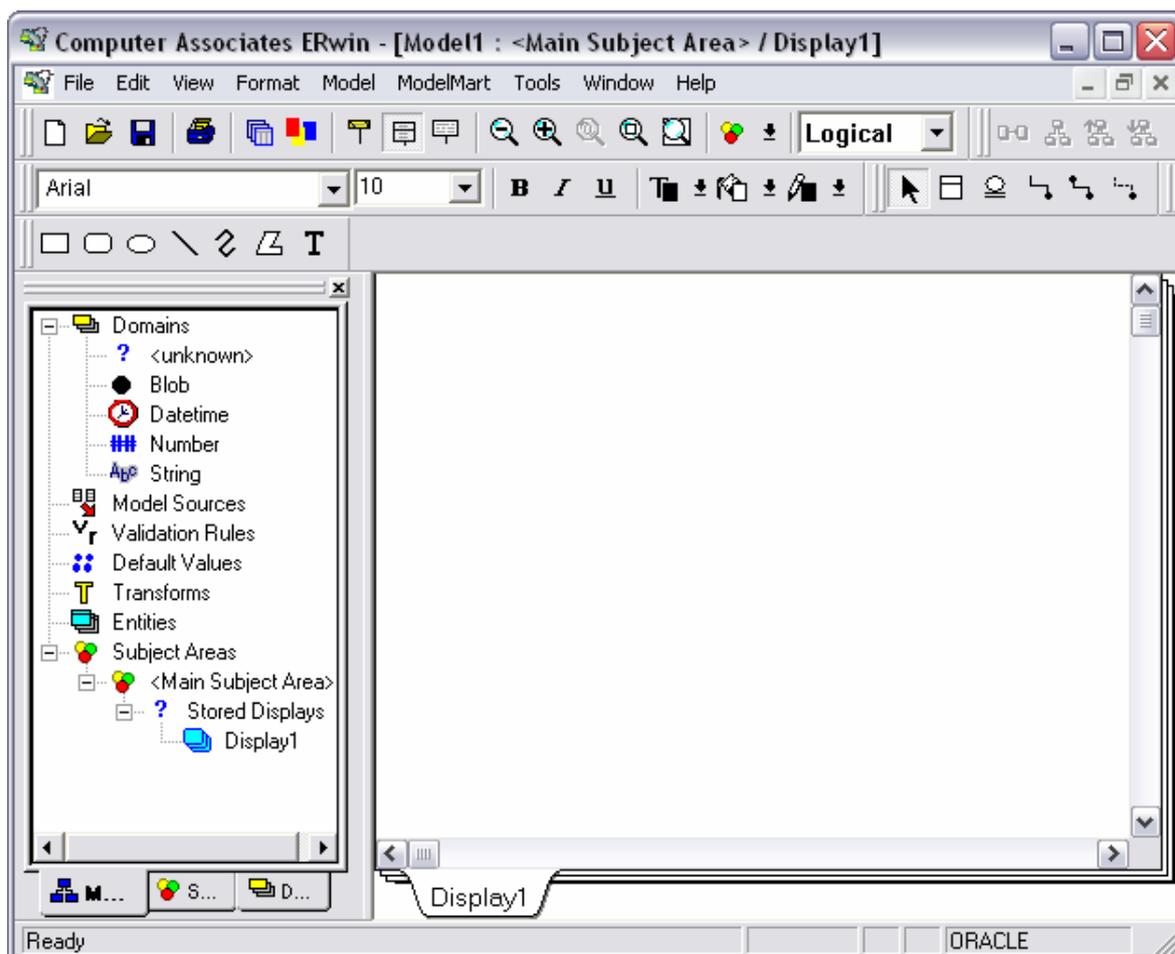


Рис. 2.1. Окно отображение логической модели

Рассмотрим кратко значения кнопок панели инструментов (табл. 2.1).

Таблица 2.1. Основная панель инструментов

Кнопки	Назначение кнопок
	Создание, открытие, сохранение и печать модели
	Изменение уровня просмотра модели: уровень сущностей, уровень атрибутов и уровень определений
	Изменение масштаба просмотра модели
	Переключение между областями модели - Subject Area
	Диалоги для генерации отчетов по модели
	Палитра инструментов

Arial	Панель инструментов Font and Color Toolbar
B <i>I</i> <u>U</u> T [] [] []	
[] [] [] [] [] []	Панель Суперкласс – подкласс
[] [] [] [] [] [] []	Панель для рисования графических объектов

Для создания типов сущностей модели и связывания их между собой используются палитра инструментов, показанная на рис. 2.2. Палитра инструментов выглядит различно на разных уровнях отображения модели. На логическом уровне палитра инструментов имеет следующие значения кнопок (табл. 2.2).



Рис. 2.2. Палитра инструментов

Таблица 2.2. Палитра инструментов

Кнопки	Назначение кнопок	Описание
	Указатель	кнопка указателя (режим мыши) - в этом режиме можно установить фокус на каком-либо объекте модели
	Сущность	кнопка внесения сущности - для внесения сущности нужно щелкнуть левой кнопкой мыши по кнопке внесения сущности и один раз по свободному пространству на модели. Для редактирования сущностей или других объектов модели необходимо перейти в режим указателя
	Категория	категория, или категориальная связь, - специальный тип связи между сущностями, которая будет рассмотрена ниже. Для установления категориальной связи нужно щелкнуть левой кнопкой мыши по кнопке категории, затем один раз щелкнуть по сущности - родовому предку, затем - по сущности-потомку
	Идентифицирующая связь	связь между независимой и зависимой сущностями (более подробно описана

		ниже по тексту)
	Связь “Многие-ко-многим”	экземпляр одной сущности может быть связан со многими экземплярами другой сущности и наоборот (возможна только на уровне логической модели)
	Неидентифицирующая связь	связь между независимыми сущностями (более подробно описана ниже по тексту)

На физическом уровне палитра инструментов имеет:

- вместо кнопки категорий — кнопку внесения *представлений* (view);
- вместо кнопки *связи* "многие-ко-многим" — кнопку *связей представлений*.
- Запустите Computer Associates ERwin. Если появится окно ModelMart Connection Manager, закройте его нажатием на кнопку Cancel.
- Если появляется диалог Computer Associates ERwin, выберите пункт Create a new model и нажмите ОК (рис. 2.3).

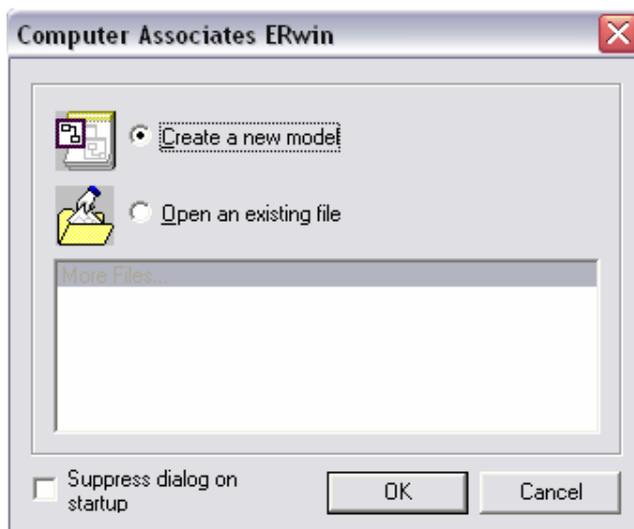


Рис. 2.3. Диалоговое окно Computer Associates ERwin

- В появившемся диалоговом окне Create Model – Select Template выберите пункт Logical/Physical и нажмите ОК (рис. 2.4).

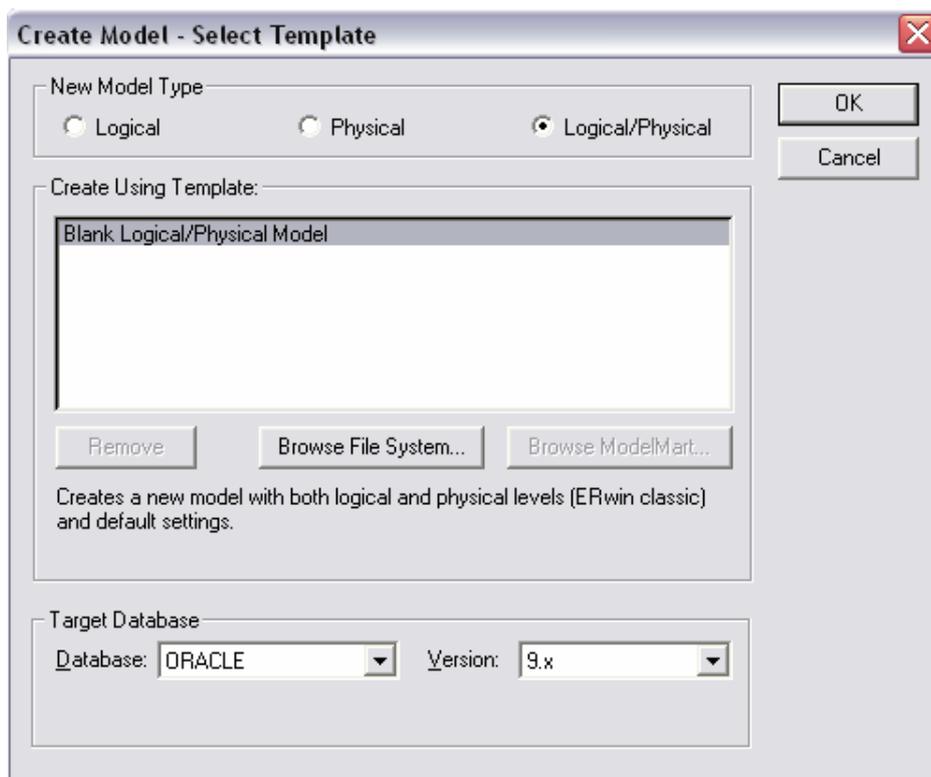


Рис. 2.4. Диалоговое окно *Create Model – Select Template*

- Если вам не понятно, как выполнить то или иное действие, вы можете вызвать помощь – клавиша F1 или меню Help.

2.2. Создание логической модели данных

2.2.1. Уровни логической модели

Различают три уровня логической модели, отличающихся по глубине представления информации о данных [8]:

- *диаграмма сущность-связь* (Entity Relationship Diagram, ERD);
- *модель данных, основанная на ключах* (Key Based model, KB);
- *полная атрибутивная модель* (Fully Attributed model, FA).

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает *сущности* и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирова-

на, в нее включаются основные *сущности* и *связи* между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. *Диаграмма сущность-связь* может включать *связи* "многие-ко-многим" и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Модель данных, основанная на ключах — более подробное представление данных. Она включает описание всех *сущностей* и *первичных ключей* и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Полная атрибутивная модель — наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все *сущности*, *атрибуты* и *связи*.

ERwin имеет несколько уровней отображения диаграммы: уровень *сущностей*, уровень *атрибутов*, уровень определений, уровень *первичных ключей* и уровень иконок. Переключиться между первыми тремя уровнями можно с использованием кнопок панели инструментов. Переключиться на другие уровни отображения можно при помощи контекстного меню, которое появляется, если нажать правую кнопку мыши на любом месте диаграммы, не занятой объектами модели. В контекстном меню следует выбрать пункт Display Level и затем необходимый уровень отображения [8].

2.2.2. Сущности и атрибуты

Основные компоненты диаграммы ERwin – это *сущности*, *атрибуты* и *связи*. Сущность можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, фактически это имя ее экземпляра. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. *Атрибут* выражает определенное свойство объекта. С точки зрения БД (фи-

зическая модель) *сущности* соответствует таблица, *экземпляру сущности* — строка в таблице, а *атрибуту* — колонка таблицы [4].

Entity Editor в контекстном меню для сущности позволяет определить имя, описание, комментарии, иконку. Для описания атрибутов сущности выбирается пункт Attribute Editor. Здесь можно указать имя нового атрибута и домен, который будет использоваться при определении типа колонки на уровне физической модели. Атрибуты должны именоваться в единственном числе, иметь четкое смысловое значение и быть достаточно важными для того, чтобы их моделировать. Именование *сущности* в единственном числе облегчает в дальнейшем чтение модели [8]. Каждый атрибут должен быть определен (закладка Definition), при этом следует избегать циклических определений и производных атрибутов. Для внесения дополнительных комментариев и определений к *сущности* служат свойства, определенные пользователем (UDP). Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения *атрибутов*.

Каждый *атрибут* хранит информацию об определенном свойстве *сущности*, а каждый экземпляр *сущности* должен быть уникальным. *Атрибут* или группа *атрибутов*, которые однозначно идентифицируют экземпляр *сущности*, называется **первичным ключом**. *Атрибуты первичного ключа* на диаграмме не требуют специального обозначения — это те *атрибуты*, которые находятся в списке *атрибутов* выше горизонтальной линии.

Продолжим рассмотрение практического моделирования на примере деятельности вымышленной компании, которая занимается в основном сборкой карманных персональных компьютеров (КПК) и установкой на них программного обеспечения (ПО). Компания не производит компоненты и программы самостоятельно, а только собирает КПК и устанавливает ПО.

Основные процедуры компании, следующие:

- продавцы принимают заказы клиентов;

- операторы группируют заказы по типу работ;
- операторы собирают КПК;
- операторы устанавливают ПО на КПК;
- операторы упаковывают КПК согласно заказам;
- кладовщик отгружает клиентам заказы.

Для начала представим модель базы данных компании. Модель должна содержать 8 сущностей: Заказ, Заказчик, Оператор, КПК, ПО, Кладовщик, Готовый товар, Продавец. Каждая из представленных сущностей имеет свой набор атрибутов.

15. Щелкните по кнопке  и на рабочем поле программы. Появится окошко сущности с названием Е/1. Окошко сущности разделено на две части. В верхнюю часть входят атрибуты, которые являются ключами сущности, а в нижнюю – не ключевые атрибуты (см. рис. 2.5).

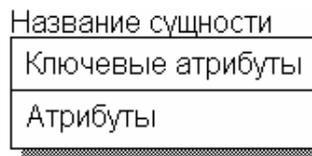


Рис. 2.5. Сущность

16. Двойной щелчок по сущности открывает диалоговое окно Attributes (рис. 2.6).

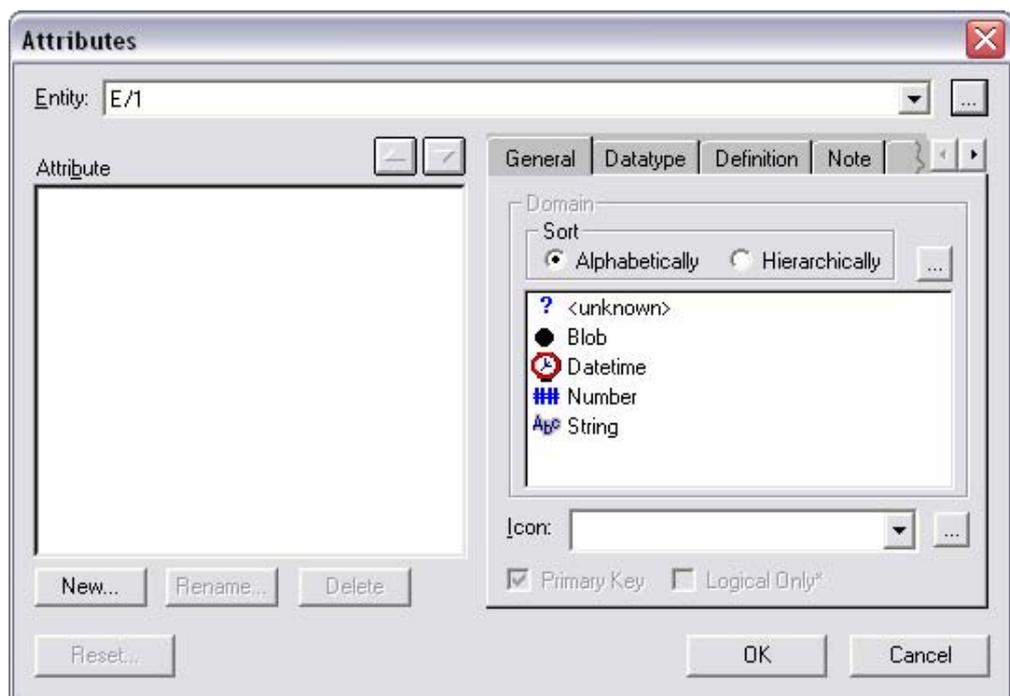


Рис. 2.6. Диалоговое окно Attributes

17. В верхнем правом углу окна расположена кнопка, которая открывает диалоговое окно Entities (рис. 2.7), где в поле Name имеется возможность изменить имя сущности. Измените имя сущности на «Заказ». После завершения ввода нажмите кнопку ОК.

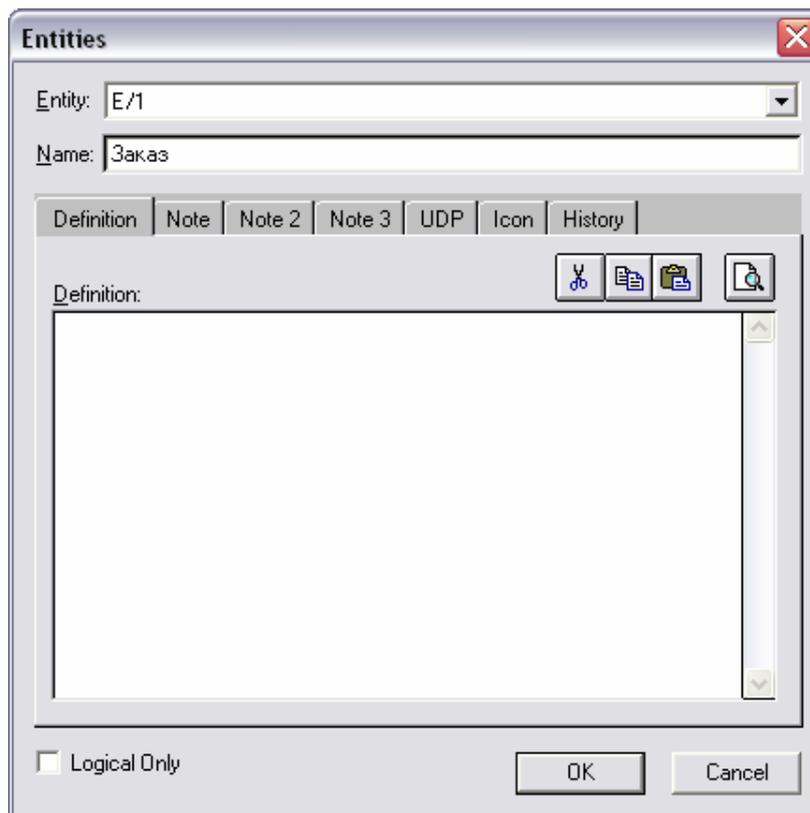


Рис. 2.7. Диалоговое окно Entities

18. Нажатием на кнопку New... в левом нижнем углу окна открывается диалоговое окно New Attribute создания нового атрибута сущности (рис. 2.8).

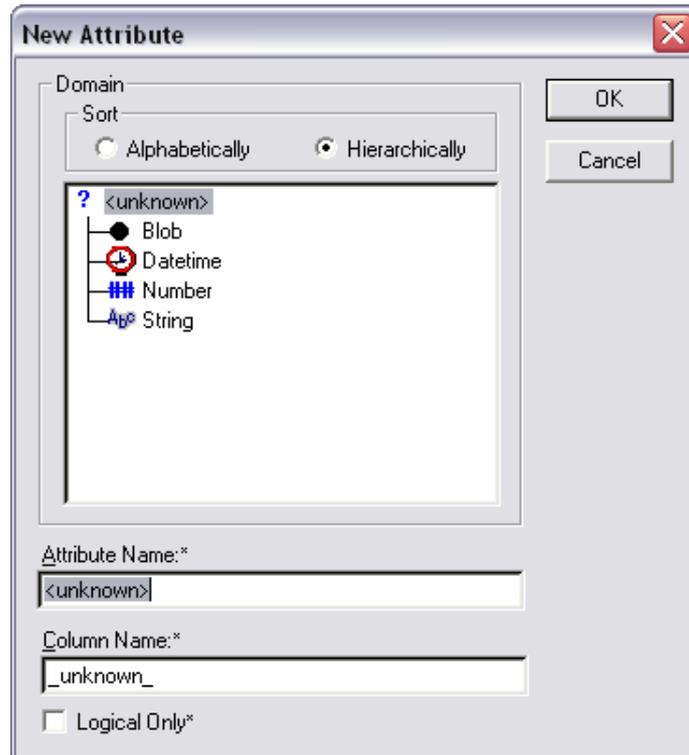


Рис. 2.8. Диалоговое окно *New Attribute*

19. В поле *Attribute Name:** введите название атрибута «Код_заказа», в окошке *Domain* выберите тип вводимых данных. Для завершения процедуры ввода нажмите *OK*. Для задания первичного ключа в окне *Attributes* для выделенного атрибута «Код_заказа» установите галочку *Primary Key*.

20. Возможен так же и другой способ ввода атрибутов в поля сущности. Чтобы заполнить окошко сущности атрибутами, нужно выбрать сущность левой кнопкой мыши и нажать клавишу «*Tab*». После ввода названия атрибута надо нажать «*Enter*», если необходимо остаться в верхней части и продолжить заполнять ключевые атрибуты, или «*Tab*» если необходимо приступить к заполнению других атрибутов. Клавиша «*Tab*» перемещает курсор между тремя полями ввода: названием сущности, ключевыми атрибутами и не ключевыми атрибутами. Задайте еще три атрибута в сущности *Заказ*: «*Стоимость*», «*Дата_заказа*» и «*Дата_выдачи*». Результат показан на рис. 2.9.

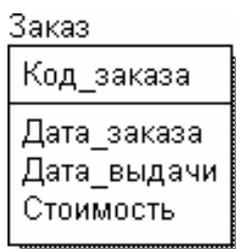


Рис. 2.9. Сущность Заказ

21. Одним из предложенных способов создайте и заполните все необходимые сущности компании, описанные в таблице 2.3.

Таблица 2.3. Сущности и атрибуты компании

Сущности	Атрибуты
Клиент	- Код_клиента (PK) - Фамилия - Имя - Отчество - Адрес - Номер счета
Заказ	- Код заказа (PK) - Дата_заказа - Дата_выдачи - Стоимость
Сотрудник	- Код_сотрудника (PK) - Фамилия - Имя - Отчество - Дата_рождения - Должность - Зарплата
КПК	- Код_КПК (PK) - Процессор - Память - Дисплей - Модем - Интерфейс
ПО	- Код_ПО (PK) - ОС - Мультимедиа - Интернет - Антивирус

	- Игры
Товар	- Код_товара (PK)
Продавец	- Код_продавца (PK)
Кладовщик	- Код_кладовщика (PK)
Оператор сборки	- Код_оператора_сборки (PK)
Оператор установки	- Код_оператора_установки (PK)

2.2.3. Связи

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (Relationship Verb Phrases). Имя связи облегчает чтение диаграммы, например (рис. 2.10):



Рис. 2.10. Пример связей

По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню для свободного места диаграммы выбрать пункт Display Option/relationship и включить опцию Verb Phrase [8].

На логическом уровне можно установить идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим. Тип сущности определяется ее связью с другими сущностями. Различают зависимые и независимые сущности. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERWin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами.

Различают несколько **типов зависимых сущностей** [4].

Характеристическая — зависимая дочерняя *сущность*, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской *сущности*.

Ассоциативная — *сущность*, связанная с несколькими родительскими *сущностями*. Такая *сущность* содержит информацию о *связях сущностей*.

Именующая — частный случай ассоциативной *сущности*, не имеющей собственных *атрибутов* (только *атрибуты* родительских *сущностей*, мигрировавших в качестве внешнего ключа).

Категориальная — дочерняя *сущность* в иерархии наследования.

При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности и помечается в дочерней сущности как внешний ключ (FK). Эта операция называется миграцией атрибутов. В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получают признак NOT NULL, что означает невозможность внесения записи в таблицу заказов без информации о номере клиента.

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Идентифицирующая *связь* показывается на диаграмме сплошной линией с жирной точкой на дочернем конце *связи*, неидентифицирующая – пунктирной.

Во вкладке *General* меню *Relationship Editor* можно задать мощность, имя и тип связи.

Мощность связей (Cardinality) — служит для обозначения отношения числа экземпляров родительской *сущности* к числу экземпляров дочерней.

Различают четыре *типа сущности*:

- общий случай, когда одному экземпляру родительской сущности соответствует 0, 1 или много экземпляров дочерней сущности (не помечается каким-либо символом);
- одному экземпляру родительской сущности соответствует 1 или много экземпляров дочерней сущности (помечается символом P);
- одному экземпляру родительской сущности соответствует 0 или 1 экземпляр дочерней сущности (помечается символом Z);
- одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности (помечается цифрой точного соответствия).

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения имени следует в контекстном меню для диаграммы (в месте не занятом объектами модели) выбрать пункт Display Options/Relationship и затем включить опцию Cardinality.

Имя связи (Verb Phrase) — фраза, характеризующая отношение между родительской и дочерней *сущностями*. Для связи "один-ко-многим", идентифицирующей или неидентифицирующей, достаточно указать имя, характеризующее отношение от родительской к дочерней *сущности* (Parent-to-Child). Для связи многие-ко-многим следует указывать имена как Parent-to-Child, так и Child-to-Parent [8].

Связь многие-ко-многим возможна только на логическом уровне. При переходе к физическому уровню Erwin автоматически преобразует связь многие-ко-многим, добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старых к новой таблице.

Иерархия наследования представляет собой особый тип объединения *сущностей*, которые разделяют общие характеристики. Обычно *иерархию наследования* создают, когда несколько *сущностей* имеют общие по смыслу ат-

рибуты, либо когда *сущности* имеют общие по смыслу *связи*, либо когда это диктуется бизнес-правилами.

Для каждой категории можно указать дискриминатор — *атрибут* родового предка, который показывает, как отличить одну категориальную *сущность* от другой.

Иерархии категорий делятся на два типа — полные и неполные. В полной категории одному экземпляру родового предка обязательно соответствует экземпляр в каком-либо потомке. Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной.

В нашем примере, один клиент может сделать несколько заказов, но один и тот же заказ у двух разных клиентов быть не может, т.е. следует применить связь один ко многим (жирная точка на конце связи означает «многие»). Аналогично рассуждая, расставьте связи между всеми сущностями модели.

1. Выберите кнопку , обозначающую связь «один ко многим» и соедините две необходимые сущности. Чтобы соединить, необходимо один раз кликнуть на сущности «один», а затем на сущности «многие».

2. Выберите кнопку , обозначающую связь «многие ко многим» и соедините необходимые две сущности.

3. Выберите кнопку , обозначающую неидентифицирующую связь, и соедините две необходимые сущности.

4. Аналогично соедините все необходимые сущности, отобразив взаимодействие между ними. Если во время установки связи у вас выскочило диалоговое окно ERwin (рис. 2.11), это означает, что название атрибута, который мигрирует из родительской сущности, уже существует. Выберите соответствующую опцию миграции и нажмите ОК.

2.2.4. Ключи

Как было сказано выше, каждый экземпляр *сущности* должен быть уникален и должен отличаться от других *атрибутов* [4].

Первичный ключ (primary key) — это *атрибут* или группа *атрибутов*, однозначно идентифицирующая экземпляр *сущности*. *Атрибуты* *первичного ключа* на диаграмме не требуют специального обозначения — это те *атрибуты*, которые находятся в списке *атрибутов* выше горизонтальной линии (рис. 2.5).

В одной *сущности* могут оказаться несколько *атрибутов* или наборов *атрибутов*, претендующих на роль *первичного ключа*. Такие претенденты называются **потенциальными ключами** (candidate key) [8].

Ключи могут быть сложными, т. е. содержащими несколько *атрибутов*. Сложные *первичные ключи* не требуют специального обозначения — это список *атрибутов*, расположенных выше горизонтальной линии.

Для того чтобы стать *первичным*, *потенциальный ключ* должен удовлетворять ряду требований [4, 8]:

- Уникальность. Два экземпляра не должны иметь одинаковых значений возможного ключа.
- Компактность. При выборе *первичного ключа* предпочтение должно отдаваться более простым ключам, т.е. ключам, содержащим меньшее количество *атрибутов*. *Атрибуты* *ключа* не должны содержать нулевых значений. Значение *атрибутов* *ключа* не должно меняться в течение всего времени существования экземпляра *сущности*.

Каждая *сущность* должна иметь по крайней мере один *потенциальный ключ*. Многие *сущности* имеют только один *потенциальный ключ*. Такой *ключ* становится *первичным*. Некоторые *сущности* могут иметь более одного возможного *ключа*. Тогда один из них становится *первичным*, а остальные —

альтернативными ключами. *Альтернативный ключ* (Alternate Key) — это *потенциальный ключ*, не ставший первичным.

2.2.5. Нормализация данных

Нормализация данных — процесс проверки и реорганизации *сущностей* и *атрибутов* с целью удовлетворения требований к реляционной *модели данных*. Нормализация позволяет быть уверенным, что каждый *атрибут* определен для своей *сущности*, а также значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте. Процесс нормализации сводится к последовательному приведению структуры данных к нормальным формам — формализованным требованиям к организации данных [8]. Известны шесть нормальных форм [4].

Первая нормальная форма (1FN). Сущность находится в первой нормальной форме в том случае, если все атрибуты содержат атомарные значения.

Для приведения сущности к первой нормальной форме следует разделить сложные атрибуты на атомарные. Для этого необходимо:

- создать новую сущность;
- перенести в нее все «повторяющиеся» атрибуты;
- выбрать возможный атрибут для нового РК (или создать новый РК);
- установить идентифицирующую связь от прежней сущности к новой.

Вторая нормальная форма (2NF). Сущность находится во второй нормальной форме, если она находится в первой нормальной форме и каждый неключевой атрибут полностью зависит от первичного ключа. Вторая нормальная форма имеет смысл только для сущностей, имеющих сложный первичный ключ.

Для приведения сущности ко второй нормальной форме следует выделить атрибуты, которые зависят только от части первичного ключа в новую сущность и установить идентифицирующую связь.

Вторая нормальная форма позволяет избежать аномалий при операциях обновления, вставки и удаления записей.

Третья нормальная форма (3NF). Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой неключевой атрибут не зависит от другого неключевого атрибута.

Для приведения сущности к третьей нормальной форме следует создать новую сущность и перенести в нее атрибуты с одной и той же зависимостью от неключевого атрибута и установить связь.

К сожалению, ERwin не содержит полного алгоритма нормализации и не может проводить нормализацию автоматически, однако его возможности облегчают создание нормализованной *модели данных*.

В результате нормализации все взаимосвязи данных становятся правильно определенными, исключаются аномалии при операциях с данными, модель данных легче поддерживать. Однако часто нормализация данных не ведет к повышению производительности ИС в целом. Поэтому в целях повышения производительности при переходе на физический уровень приходится сознательно отходить от нормальных форм (проводить операцию денормализации) для того, чтобы использовать возможности конкретного сервера или ИС в целом.

2.2.6. Домены

Домен можно определить как совокупность значений, из которых берутся значения *атрибутов*. Каждый *атрибут* может быть определен только на одном *домене*, но на каждом *домене* может быть определено множество *атри-*

бутов. В понятие *домена* входит не только тип данных, но и область значений данных.

В ERwin *домен* может быть определен только один раз и использоваться как в логической, так и в физической модели.

Домены позволяют облегчить работу с данными как разработчикам на этапе проектирования, так и администраторам БД на этапе эксплуатации системы. На логическом уровне *домены* можно описать без конкретных физических свойств. На физическом уровне они автоматически получают специфические свойства, которые можно изменить вручную.

Для создания домена в логической модели служит диалог Domain Dictionary Editor. Его можно вызвать из меню Edit/Domain Dictionary по кнопке, расположенной в верхней левой части закладки General диалога Attribute Editor. Каждый *домен* может быть описан, снабжен комментарием или свойством, определенным пользователем (UDP) [8].

ERwin имеет специальный инструмент, который значительно облегчает создание новых атрибутов в модели, используя описание доменов Independent Attribute Browser. Этот диалог вызывается с помощью CTRL+B [4].

2.3. Создание физической модели данных

Физическая модель содержит всю информацию, необходимую для реализации конкретной БД. Различают два уровня физической модели [8]:

- трансформационная модель;
- модель СУБД.

Трансформационная модель содержит информацию для реализации отдельного проекта, который может быть частью общей ИС и описывать подмножество предметной области. Данная модель позволяет проектировщикам и администраторам БД лучше представить, какие объекты БД хранятся в слова-

ре данных, и проверить, насколько физическая модель удовлетворяет требованиям к ИС.

Модель СУБД автоматически генерируется из трансформационной модели и является точным отображением системного каталога СУБД.

Физический уровень *представления* модели зависит от выбранного сервера. ERwin поддерживает более 20 реляционных и нереляционных БД. Для выбора СУБД служит редактор Target Server (меню Server/Target Server... доступен только на физическом уровне).

По умолчанию ERwin генерирует имена таблиц и *индексов* по шаблону на основе имен соответствующих *сущностей* и ключей логической модели, которые в дальнейшем могут быть откорректированы вручную. Имена таблиц и колонок будут сгенерированы по умолчанию на основе имен *сущностей* и *атрибутов* логической модели. Значения по умолчанию можно при желании изменить путем изменения шаблона (Target Server / Table Name Macro) или вручную [4].

Диалог Target Server позволяет задать тип данных и опцию NULL для новых колонок. Тип данных можно выбрать в списке Default Datatype, который автоматически заполняется типами данных, поддерживаемых выбранным сервером. Группа кнопок Default Non-Key Null Option позволяет разрешить или запретить значения NULL для неключевых колонок.

Физическая модель отличается от логической тем, что каждому атрибуту сущности необходимо присваивать тип вводимых данных и их размер. Так же, в физической модели не допустимы связи «многие ко многим», поэтому эти связи необходимо исключить или каким-нибудь образом заменить на связь «один ко многим». Атрибуты и названия сущностей необходимо писать на английском или транслитом. Всё это надо сделать для того, чтобы была возможность импорта разрабатываемой базы данных в СУБД и корректной работы с ней.

Внесение новых таблиц и связей между ними на физическом уровне производятся также как на логическом уровне. Вызвать редакторы Table Editor или Column Editor для задания свойств таблиц и колонок можно через контекстно-зависимое меню для этих таблиц. Все изменения, сделанные в Table Editor или Column Editor, не отражаются на именах сущностей и атрибутов, поскольку информация на логическом и физическом уровнях в Egwin хранится отдельно.

Редактор Table Editor позволяет задать свойства любой таблицы модели, отличные от значения по умолчанию, в том числе имя таблицы, синонимы, правила валидации, процедуры и т.д. [8].

Представления (view) или иными словами временные или производные таблицы, представляют собой объекты БД, данные в которых не хранятся постоянно, как в таблице, а формируются динамически при обращении к представлению. Представление не может существовать само по себе, а определяется только в терминах одной или нескольких таблиц. Применение представлений позволяет разработчику БД обеспечить каждому пользователю или группе пользователей свой взгляд на данные, что решает проблемы простоты использования и безопасности данных. ERwin имеет специальные инструменты для создания и редактирования представлений. Палитра инструментов на физическом уровне содержит кнопки внесения представлений и установления связей между таблицами и представлениями. По умолчанию представление получает номер V_n, где n – уникальный порядковый номер представления.

Для редактирования представления необходимо выбрать в контекстно-зависимом меню для представления пункт View Editor. Каждой таблице можно задать необходимую информацию, которая будет использоваться в SQL-команде для создания представления [4].

2.3.1. Построение физической модели

1. Для переключения между логической и физической моделью данных служит список выбора в центральной части панели инструментов ERwin (рис. 2.13). Переключитесь на отображение физической модели. Если при переключении физической модели еще не существует, она будет создана автоматически.

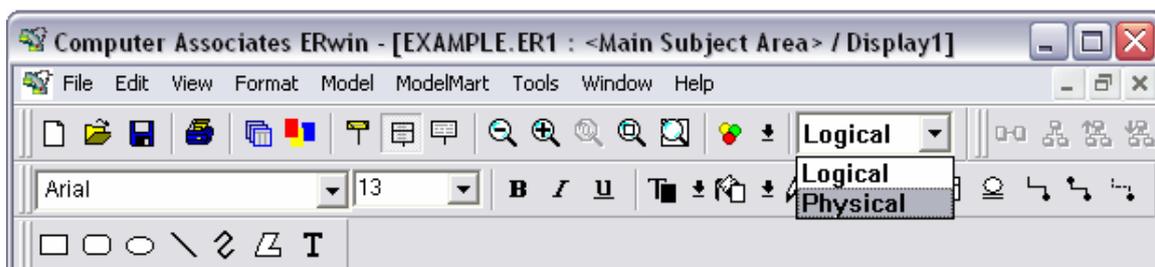


Рис. 2.13. Переключение между логической и физической моделью

2. Если есть необходимость начать новый проект по созданию физической модели, необходимо выбрать File/New... в главном меню программы (Ctrl + N). В появившемся диалоговом окне Create Model – Select Template необходимо выбрать пункт Physical, а в поле Target Database выбрать СУБД с которой планируется работать и нажать ОК (рис. 2.4).

3. Принцип построения физической модели аналогичен построению логической с небольшой поправкой: для каждого атрибута сущности необходимо указывать тип вводимых данных. Для этого в свойствах сущности выберите пункт Columns... и закладку ORACLE в появившемся диалоговом окне (рис. 2.14).

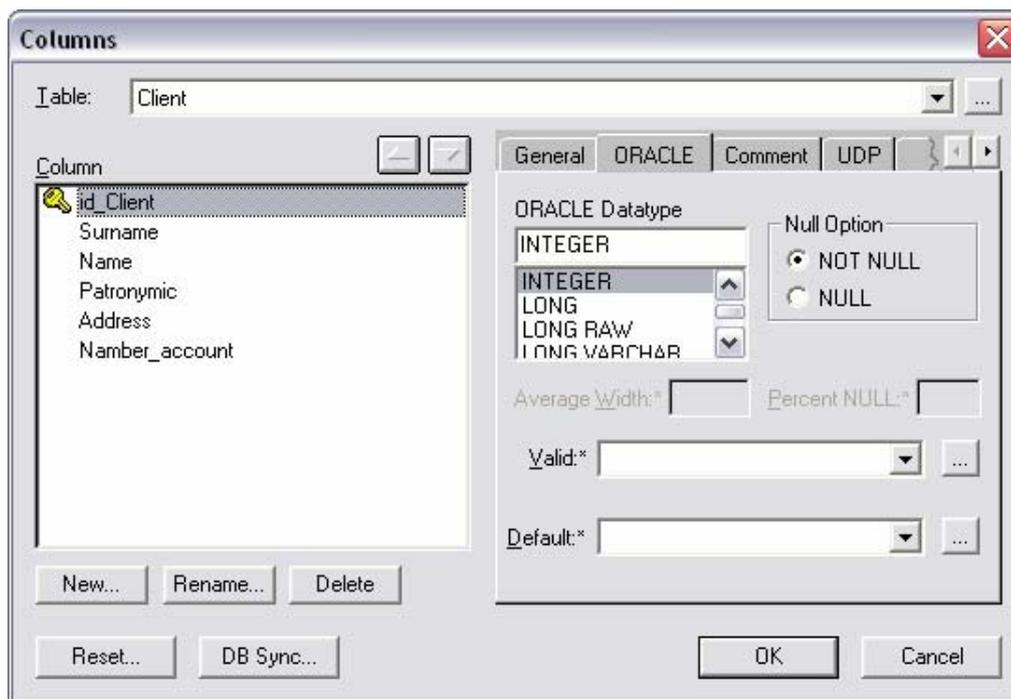


Рис. 2.14. Диалоговое окно Columns с закладкой ORACLE

4. Необходимо проверить названия сущностей и атрибутов с учётом правила именования объектов в Oracle, при которых идентификатор:

- может содержать до 30 символов (латинские буквы, цифры, символ подчеркивания, \$, #);
- начинается с буквы;
- нечувствителен к регистру (eMp = EMP);
- уникален в схеме пользователя;
- отличен от зарезервированных слов.

5. Сущность Заказ физической модели с указанными типами вводимых данных должна выглядеть следующим образом (рис. 2.15):

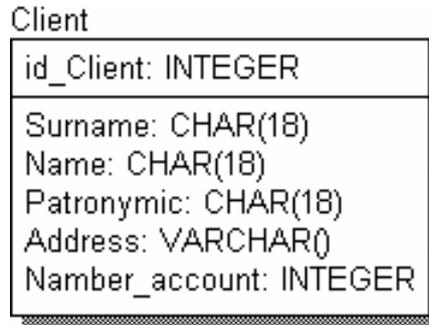


Рис. 2.15. Сущность Заказ физической модели

2.3.2. Прямое и обратное проектирование

Процесс генерации физической схемы БД из логической модели данных называется прямым проектированием (Forward Engineering). При генерации физической схемы Erwin включает триггеры ссылочной целостности, хранимые процедуры, индексы, ограничения и другие возможности, доступные при определении таблиц в выбранной СУБД.

Процесс генерации логической модели из физической БД называется обратным проектированием (Reverse Engineering). Erwin позволяет создать модель данных путем обратного проектирования имеющейся БД. После того, как модель создана, можно переключиться на другой сервер (модель будет конвертирована) и произвести прямое проектирование структуры БД для другой СУБД [4].

2.3.3. Правила валидации и значения по умолчанию

ERwin поддерживает *правила валидации* для колонок, а также значение, присваиваемое колонкам по умолчанию.

Правило валидации задает список допустимых значений для конкретной колонки и/или правила проверки допустимых значений. В список допустимых значений можно вносить новые значения [8]. ERwin позволяет сгенерировать

правила валидации соответственно синтаксису выбранной СУБД с учетом границ диапазона или списка значений.

Значение по умолчанию – значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных. С каждой колонкой или *доменом* можно связать значение по умолчанию. Список значений можно редактировать.

После создания *правила валидации* и значения по умолчанию их можно присвоить одной или нескольким колонкам или *доменами*.

1. В окне Model/Validation Rules задайте максимальное и минимальное значение и тип валидации. Например, значение, вводимое в колонку Зарплата, должно быть больше 500, но меньше 100000. Для описания этого правила можно создать правило валидации с именем «Проверка зарплаты», которое содержит выражение: Зарплата BETWEEN 500 AND 100000 (рис. 2.16). СУБД выдаст сообщение об ошибке, если вводимая зарплата находится вне границ диапазона.

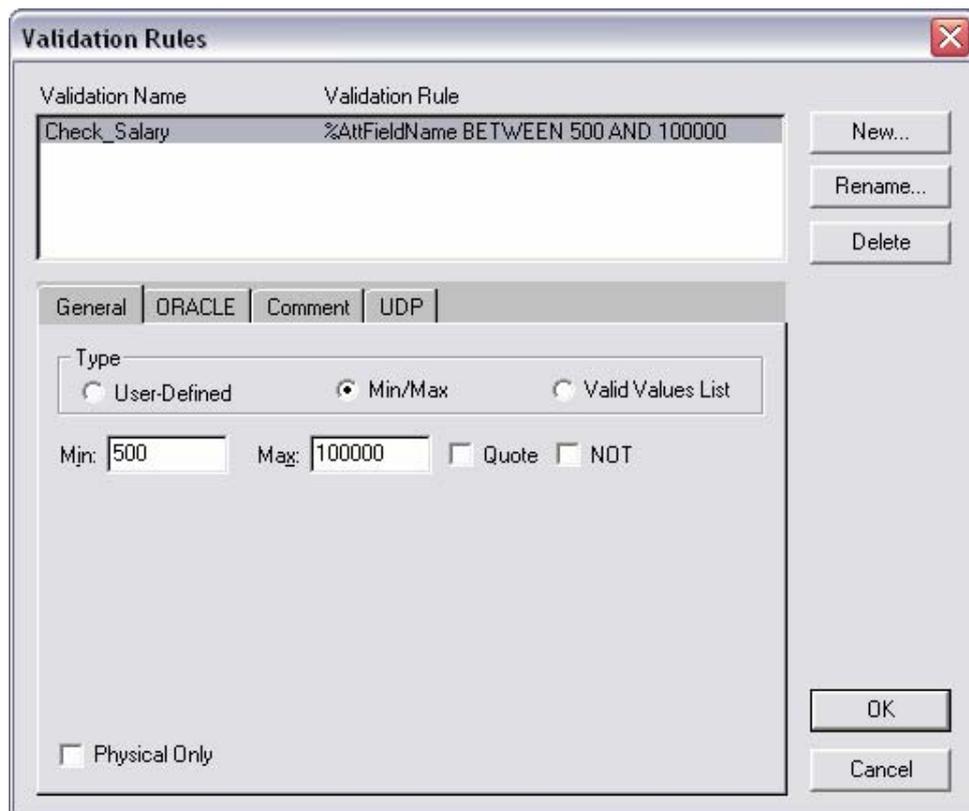


Рис. 2.16. Окно *Validation Rules*

2. Выберите пункт меню Model/Default Value.

3. В открывшемся редакторе Default/Initial Editor задайте значения, которые автоматически, по умолчанию, будут присваиваться колонке. Например, дате оформления заказа может быть присвоено значение по умолчанию «сегодняшнее число», т.е. автоматически задается, что все заказы оформляются в день ввода информации о них в БД.

2.3.4. Индексы

В БД данные обычно хранятся в том порядке, в котором их ввели в таблицу. Многие реляционные СУБД имеют страничную организацию, при которой таблица может храниться фрагментарно в разных областях диска, причем строки таблицы располагаются на страницах неупорядоченно [8]. Такой способ позволяет быстро вводить новые данные, но затрудняет поиск данных [4].

Чтобы решить проблему поиска, СУБД используют объекты, называемые *индексами*. *Индекс* содержит отсортированную по колонке или нескольким колонкам информацию и указывает на строки, в которых хранится конкретное значение колонки. Поскольку значения в *индексе* хранятся в определенном порядке, при поиске просматривать нужно значительно меньший объем данных, что существенно уменьшает время выполнения запроса. *Индекс* рекомендуется создавать для тех колонок, по которым часто производится поиск.

При генерации схемы физической БД ERwin автоматически создает *индекс* на основе *первичного ключа* каждой таблицы, а также на основе всех *альтернативных ключей* и внешних ключей, поскольку эти колонки наиболее часто используются для поиска данных. Можно отказаться от генерации *индексов* по умолчанию и создать собственные *индексы* [8]. Для увеличения эффективности поиска администратор БД должен анализировать часто выполняемые запросы и на основе анализа создавать собственные *индексы* [4].

1. Вызовите редактор Indexes.
2. Измените, имена индекса и их определения так, чтобы они принимали уникальные или дублирующие значения, или измените порядок сортировки данных.

2.3.5. Триггеры и хранимые процедуры

Триггеры и хранимые процедуры – это именованные блоки кода SQL, которые заранее откомпилированы и хранятся на сервере для того, чтобы быстро производить обработку запросов, валидацию данных и другие часто выполняемые функции. Хранение и выполнение кода на сервере позволяет давать код только один раз, а не в каждом приложении, работающем с БД. Это экономит время при написании и сопровождении программ. При этом гарантируется, что целостность данных и бизнес-правила поддерживаются независимо от того, какое именно клиентское приложение обращается к данным.

Триггеры и *хранимые процедуры* не требуется пересылать по сети из клиентского приложения, что значительно снижает сетевой трафик [4].

Хранимой процедурой называется именованный набор предварительно откомпилированных команд SQL, который может вызываться из клиентского приложения или из другой *хранимой процедуры*.

Триггером называется процедура, которая выполняется автоматически как реакция на событие.

Триггер ссылочной целостности – это особый вид *триггера*, используемый для поддержания целостности между двумя таблицами, которые связаны между собой. Если строка в одной таблице вставляется, изменяется или удаляется, то *триггер* ссылочной целостности сообщает СУБД, что нужно делать с теми строками в других таблицах, у которых значение внешнего ключа совпадает со значением *первичного ключа* вставленной строки (измененной или удаленной строки) [8].

Для генерации *триггеров* ERwin использует механизм шаблонов – специальных скриптов, использующих макрокоманды. При генерации кода *триггера* вместо макрокоманд подставляются имена таблиц, колонок, переменные и другие фрагменты кода, соответствующие синтаксису выбранной СУБД. Шаблоны *триггеров* ссылочной целостности, генерируемые ERwin по умолчанию, можно изменять.

Для создания и редактирования *хранимых процедур* ERwin располагает специальными редакторами, аналогичными редакторам, используемым для создания *триггеров*. В отличие от *триггера* *хранимая процедура* не выполняется в ответ на какое-то событие, а вызывается из другой программы, которая передает на сервер имя процедуры. *Хранимая процедура* более гибкая, чем *триггер*, поскольку может вызывать другие *хранимые процедуры*. Ей можно передавать параметры, и она может возвращать параметры, значения и сообщения.

1. Для таблицы из контекстно-зависимого меню выберите пункт Triggers.
2. В открывшемся окне осуществите операции по созданию и редактированию триггеров и процедур.
3. Окончательный вид физической модели базы данных показан на рис. 2.17.

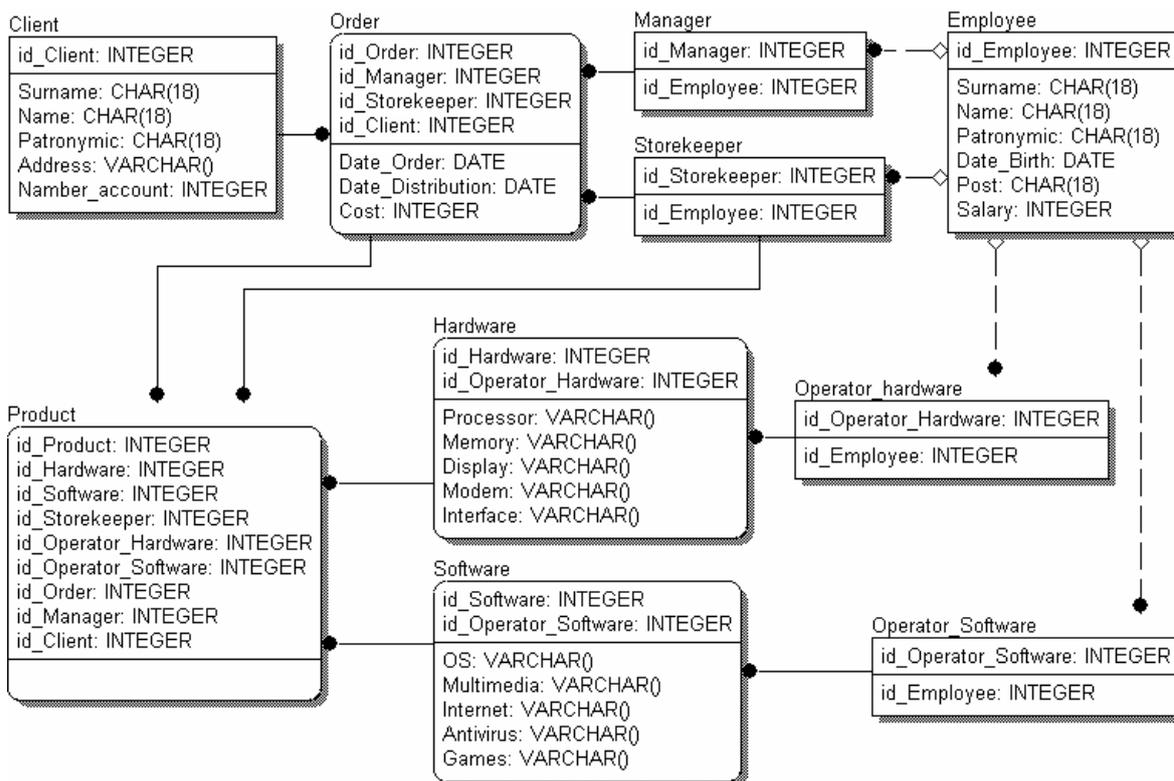


Рис. 2.17. Физическая модель базы данных компании

4. Для генерации отчетов в ERwin имеется простой и эффективный инструмент – Report Builder. Создайте с помощью него собственный отчет. Каждый отчет может быть настроен индивидуально, данные в нем могут быть отсортированы и отфильтрованы.

2.3.6. Перевод физической модели в SQL-скрипт для СУБД ORACLE

Кроме режима прямого и обратного проектирования ERwin поддерживает синхронизацию между логической моделью и системным каталогом СУБД на протяжении всего жизненного цикла создания БД.

1. Для генерации системного каталога БД следует выбрать пункт меню Tools/Forward Engineering/Schema Generation. В окне Schema Generation в закладке Options можно задать опции генерации объектов БД – триггеров, таблиц, представлений, колонок, индексов и т.д.

2. Кнопка Preview позволяет отобразить SQL-скрипт, создаваемый ERwin для генерации системного каталога СУБД.

3. Нажатие кнопки Generate приведет к запуску процесса генерации схемы. Возникает диалог связи с БД, устанавливается сеанс связи с сервером и начинает выполняться SQL-скрипт.

4. Кнопка Report сохраняет тот же скрипт в SQL текстовом файле. Эти команды можно в дальнейшем редактировать любым текстовым редактором и выполнять при помощи соответствующей утилиты сервера.

Вопросы

- 2.1. Укажите, к какому уровню детализации относится диаграмма сущность-связь.
- 2.2. Укажите, к какому уровню детализации относится полная атрибутивная модель.
- 2.3. Укажите, к какому уровню детализации относится модель данных, основанная на ключах.
- 2.4. Укажите, что позволяют осуществить диаграммы ERD.
- 2.5. Укажите, что задает правило валидации.
- 2.6. Укажите базовые понятия ERD-диаграммы.
- 2.7. Укажите, какая модель данных представляет данные в третьей нормальной форме.
- 2.8. Укажите, какие уровни отображения диаграммы имеет ERwin.

- 2.9. Укажите, какая модель данных включает описание всех сущностей и первичных ключей.
- 2.10. Основные этапы проектирования базы данных
- 2.11. Принципы построения логической модели данных.
- 2.12. Какие типы связей используются при построении модели «сущность-связь»?
- 2.13. Привести примеры идентифицирующих и неидентифицирующих связей?
- 2.14. Что такое мощность связи?
- 2.15. Какие объекты БД генерируются при проектировании физической схемы?
- 2.16. Отличия в идентификации объектов на логическом и физическом уровне.
- 2.17. Как можно осуществить конвертирование БД из одной СУБД в другую?
- 2.18. Что такое триггер? Какие элементы логической модели являются основополагающими для создания триггеров при прямом проектировании?

Глава 3. Визуальное моделирование информационных систем в среде IBM Rational Rose 2003

3.1. Унифицированный язык объектно-ориентированного моделирования UML

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML) – это стандартная нотация визуального моделирования программных систем, принятая консорциумом Object Managing Group (OMG) осенью 1997 г.

В настоящее время консорциум пользователей *UML Partners* включает в себя представителей таких грандов информационных технологий, как Rational Software, Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Unisys, IntelliCorp, Platinum Technology.

Существует достаточное количество инструментальных средств, поддерживающих с помощью *UML* жизненный цикл информационных систем, и, одновременно, *UML* является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

UML представляет собой *объектно-ориентированный* язык моделирования, обладающий следующими основными характеристиками [4]:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;
- содержит механизмы расширения и специализации базовых концепций языка.

UML включает внутренний набор средств моделирования, которые сейчас приняты во многих методах и средствах моделирования. Эти концепции необходимы в большинстве прикладных задач, хотя не каждая концепция необхо-

дима в каждой части каждого приложения. Пользователям языка предоставлены возможности:

- строить модели на основе средств ядра, без использования механизмов расширения для большинства типовых приложений;
- добавлять при необходимости новые элементы и условные обозначения, если они не входят в ядро, или специализировать компоненты, систему условных обозначений (нотацию) и ограничения для конкретных предметных областей.

3.1.1. Классы

Классы — это базовые элементы любой *объектно-ориентированной* системы. *Классы* представляют собой описание совокупностей однородных объектов с присущими им свойствами — атрибутами, операциями, отношениями и семантикой.

В рамках модели каждому *классу* присваивается уникальное имя, отличающее его от других *классов*. Если используется составное имя (в начале имени добавляется имя пакета, куда входит *класс*), то имя *класса* должно быть уникальным в пакете.

Атрибут — это свойство *класса*, которое может принимать множество значений. Множество допустимых значений атрибута образует домен. Атрибут имеет имя и отражает некоторое свойство моделируемой сущности, общее для всех объектов данного *класса*. *Класс* может иметь произвольное количество атрибутов.

Операция — реализация функции, которую можно запросить у любого объекта *класса*. Операция показывает, что можно сделать с объектом. Исполнение операции часто связано с обработкой и изменением значений атрибутов объекта, а также изменением состояния объекта [4].

На рис. 3.1 приведено графическое изображение *класса* «Заказ» в нотации *UML*.



Рис. 3.1. Изображение класса в *UML*

Синтаксис *UML* для свойств классов:

<признак видимости> <имя атрибута> : <тип данных> = <значение по умолчанию>

<признак видимости> <имя операции> <(список аргументов)>

В языке *UML* определены три уровня видимости [4]:

- **public** (общий) — любой внешний *класс*, который «видит» данный, может пользоваться его общими свойствами. Обозначаются знаком «+» перед именем атрибута или операции;
- **protected** (защищенный) — только любой потомок данного *класса* может пользоваться его защищенными свойствами. Обозначаются знаком «#»;
- **private** (закрытый) — только данный *класс* может пользоваться этими свойствами. Обозначаются символом «-» .

Область действия свойства указывает, будет ли оно проявлять себя по-разному в каждом экземпляре *класса*, или одно и то же значение свойства будет совместно использоваться всеми экземплярами:

- **instance** (экземпляр) — у каждого экземпляра *класса* есть собственное значение данного свойства;

- classifier (классификатор) — все экземпляры совместно используют общее значение данного свойства (выделяется на диаграммах подчеркиванием).

Возможное количество экземпляров *класса* называется его кратностью. В *UML* можно определять следующие разновидности *классов* [4]:

- не содержащие ни одного экземпляра — тогда *класс* становится служебным (Abstract);
- содержащие ровно один экземпляр (Singleton);
- содержащие заданное число экземпляров;
- содержащие произвольное число экземпляров.

Принципиальное назначение *классов* характеризуют стереотипы. Это, фактически, классификация объектов на высоком уровне, позволяющая определить некоторые основные свойства. Механизм стереотипов является также средством расширения словаря *UML* за счет создания на основе существующих блоков языка новых, специфичных для решения конкретной проблемы.

3.1.2. Этапы проектирования ИС с применением UML

UML обеспечивает поддержку всех этапов жизненного цикла ИС и предоставляет для этих целей ряд графических средств – диаграмм [6].

На этапе создания *концептуальной модели*:

- для описания бизнес-деятельности используются *модели бизнес-прецедентов* и диаграммы видов деятельности;
- для описания бизнес-объектов – *модели бизнес-объектов* и диаграммы последовательностей.

На этапе создания логической модели ИС:

- описание требований к системе задается в виде модели и описания системных прецедентов;

- предварительное проектирование осуществляется с использованием диаграмм классов, диаграмм последовательностей и диаграмм состояний.

На этапе создания физической модели:

- детальное проектирование выполняется с использованием диаграмм классов, диаграмм компонентов, диаграмм развертывания.

Ниже приводятся определения и описывается назначение перечисленных диаграмм и моделей применительно к задачам проектирования ИС (в скобках приведены альтернативные названия диаграмм, используемые в современной литературе) [4].

- Диаграммы прецедентов (диаграммы вариантов использования, use case diagrams) – это обобщенная модель функционирования системы в окружающей среде.

- Диаграммы видов деятельности (диаграммы деятельностей, activity diagrams) – модель бизнес-процесса или поведения системы в рамках прецедента.

- Диаграммы взаимодействия (interaction diagrams) – модель процесса обмена сообщениями между объектами, представляется в виде диаграмм последовательностей (sequence diagrams) или кооперативных диаграмм (collaboration diagrams).

- Диаграммы состояний (statechart diagrams) – модель динамического поведения системы и ее компонентов при переходе из одного состояния в другое.

- Диаграммы классов (class diagrams) – логическая модель базовой структуры системы, отражает статическую структуру системы и связи между ее элементами.

- Диаграммы базы данных (database diagrams) — модель структуры базы данных, отображает таблицы, столбцы, ограничения и т.п.

- Диаграммы компонентов (component diagrams) – модель иерархии подсистем, отражает физическое размещение баз данных, приложений и интерфейсов ИС.
- Диаграммы развертывания (диаграммы размещения, deployment diagrams) – модель физической архитектуры системы, отображает аппаратную конфигурацию ИС.

На рис. 3.2 показаны отношения между различными видами диаграмм UML. Указатели стрелок можно интерпретировать как отношение «является источником входных данных для...» (например, диаграмма прецедентов является источником данных для диаграмм видов деятельности и последовательности). Приведенная схема является наглядной иллюстрацией итеративного характера разработки моделей с использованием UML [4].



Рис. 3.2. Взаимосвязи между диаграммами UML

Ниже приводятся описания последовательных этапов проектирования ИС с использованием UML:

- **Модель бизнес-прецедентов** описывает бизнес-процессы с точки зрения внешнего пользователя, т.е. отражает взгляд на деятельность организации извне.
- Следующим этапом проектирования ИС является разработка *модели бизнес-объектов*, которая показывает выполнение бизнес-процессов органи-

зации ее внутренними исполнителями. Основными компонентами *моделей бизнес-объектов* являются внешние и внутренние исполнители, а также бизнес-сущности, отображающие все, что используют внутренние исполнители для реализации бизнес-процессов.

- Затем на основе информации, выявленной на этапах бизнес-моделирования, выполняется разработка *концептуальной модели данных*, которые будут использоваться в разрабатываемой системе.

- На этапе формирования требований, основой разработки является *модель системных прецедентов*, отражающая выполнение конкретных обязанностей внутренними и внешними исполнителями с использованием ИС.

- На этапе анализа требований и предварительного проектирования системы основным инструментом являются диаграммы классов системы, которые строятся на основе разработанной *модели системных прецедентов*. Одновременно на этом этапе уточняются диаграммы последовательностей выполнения отдельных прецедентов, что приводит к изменениям в составе объектов и диаграммах классов.

- На этапе разработки *моделей базы данных и приложений* осуществляется отображение элементов полученных ранее моделей классов в элементы моделей базы данных и приложений.

- На этапе проектирования физической реализации системы, модели баз данных и приложений дополняются обозначениями их размещения на технических средствах разрабатываемой системы.

Таким образом, при проектировании сложной ИС она разделяется на части, и каждая из них затем исследуется и создается отдельно. В настоящее время используются два различных способа такого разбиения ИС на подсистемы: структурное (или функциональное) разбиение и объектная (компонентная) декомпозиция.

Если при проектировании ИС разбивается на объекты, то для ее визуального моделирования следует использовать UML. При выборе подхода к разработке ИС следует учитывать, что визуальные модели все более широко используются в существующих технологиях управления проектированием систем, сложность, масштабы и функциональность которых постоянно возрастают. Они хорошо приспособлены для решения таких часто возникающих при создании систем задач как: физическое перераспределение вычислений и данных, обеспечение параллелизма вычислений, репликация БД, обеспечение безопасности доступа к ИС, оптимизация балансировки нагрузки ИС, устойчивость к сбоям и т.п. Визуализированные средствами UML модели ИС позволяют наладить плодотворное взаимодействие между заказчиками, пользователями и командой разработчиков. Они обеспечивают ясность представления выбранных архитектурных решений и позволяют понять разрабатываемую систему во всей ее полноте.

3.2. Общая характеристика и особенности рабочего интерфейса 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. Rational Rose 2003 аккумулирует 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.

В CASE-средстве IBM Rational Rose 2003 реализованы общепринятые стандарты на *рабочий интерфейс программы*, аналогично известным средам визуального программирования [5].

Рабочий интерфейс программы IBM Rational Rose 2003 состоит из различных элементов, основными из которых являются (рис. 3.3):

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

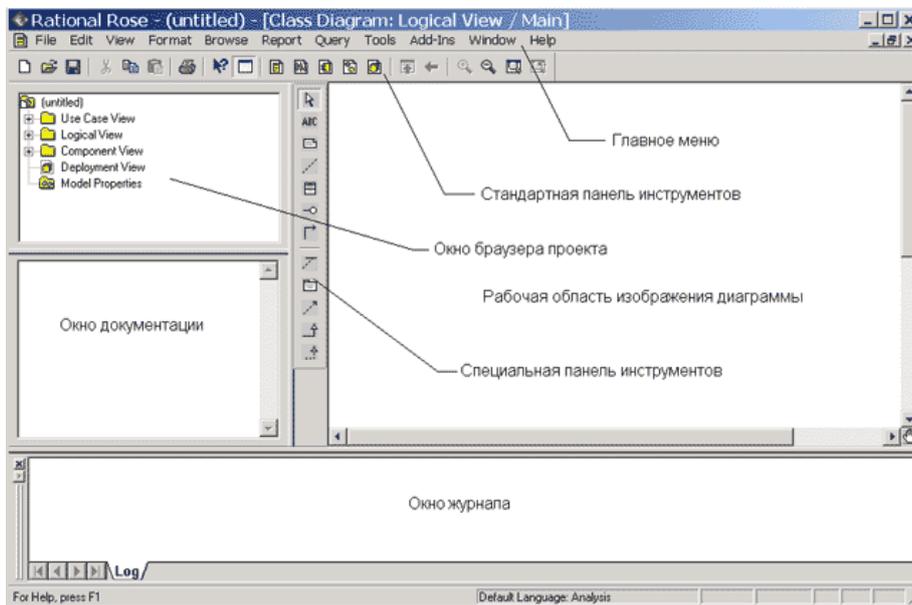


Рис. 3.3. Общий вид рабочего интерфейса CASE-средства IBM Rational Rose 2003

Главное меню программы IBM Rational Rose 2003 выполнено в общепринятом стандарте и имеет следующий вид (рис. 3.4).



Рис. 3.4. Внешний вид главного меню программы

Стандартная панель инструментов располагается ниже строки главного меню и имеет следующий вид (рис. 3.5).



Рис. 3.5. Внешний вид стандартной панели инструментов

Пользователь может настроить внешний вид этой панели по своему усмотрению. Для этого необходимо выполнить операцию *главного меню*: **Tools/Options**, открыть вкладку **Toolbars** появившегося диалогового окна и нажать кнопку **Standard**. В дополнительно открытом окне можно переносить требуемые кнопки из левого списка в правый список, а ненужные кнопки - из правого списка в левый. Данным способом можно показать или скрыть различные кнопки инструментов, а также изменить их размер.

Назначение операций главного меню очень подробно описано в литературе [5].

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. Браузер проекта позволяет также организовывать элементы модели в пакеты и перемещать элементы между различными представлениями модели.

Иерархическое представление структуры каждого разрабатываемого проекта организовано в форме следующих 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.

Специальная панель инструментов располагается между *окном браузера* и *окном диаграммы* в средней части рабочего интерфейса. По умолчанию предлагается панель инструментов для построения диаграммы классов модели (рис. 3.6).



Рис. 3.6. Внешний вид специальной панели инструментов для диаграммы классов

Программа IBM Rational Rose 2003 позволяет настраивать состав кнопок данной *панели*, добавляя или удаляя отдельные кнопки, соответствующие тем или иным инструментам. Назначение отдельных кнопок различных панелей инструментов подробно рассмотрено в литературе [5]. Названия кнопок данной *панели* всегда можно узнать из всплывающих подсказок, появляющихся после задержки указателя мыши над соответствующей кнопкой.

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**.

Окно диаграммы является основной графической областью программы IBM Rational Rose 2003, в которой визуализируются различные представления модели проекта. По умолчанию *окно диаграммы* располагается в правой части рабочего интерфейса, однако его расположение и размеры также можно изменить. При разработке нового проекта, если не был использован мастер проектов, *окно диаграммы* представляет собой чистую область, не содержащую никаких элементов модели (рис. 3.3). По мере разработки отдельных диаграмм в *окне диаграммы* будут располагаться соответствующие графические элементы модели.

Название диаграммы, которая является активной и располагается в данном окне, указывается в строке заголовка программы IBM Rational Rose 2003, если она развернута на всю область диаграммы, иначе название диаграммы указывается в строке заголовка *окна диаграммы*. Одновременно в графической области диаграмм могут присутствовать несколько окон диаграмм, переключение между которыми можно осуществить выбором нужного представления на стандартной панели инструментов, а также с помощью выделения требуемой диаграммы в браузере проекта или с помощью операций главного меню **Window**. При активизации отдельного вида диаграммы изменяется внешний вид *специальной панели инструментов*, которая настраивается под конкретный вид диаграммы [5].

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**, после чего *окно документации* появится ниже *окна браузера проекта*. *Окно документации* предназначено для документирования элементов разрабатываемой модели. В него можно записывать различную текстовую информацию, в том числе и на

русском языке. Эта информация при генерации программного кода преобразуется в комментарии и никак не влияет на логику выполнения программного кода.

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. Если *окно журнала* отсутствует на экране или наоборот необходимо его убрать, то можно воспользоваться операцией главного меню **View/Log**, для чего следует выставить или снять отметку в соответствующей строке вложенного меню для данной операции.

3.3. Разработка диаграммы вариантов использования и редактирование свойств ее элементов

3.3.1. Особенности разработки диаграмм вариантов использования

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 [5]. В качестве проекта далее будет рассматриваться модель функционирования мобильного телефона. Достоинством этого проекта является то, что он не требует специального описания предметной области, поскольку предполагает интуитивное знакомство читателей с особенностями функционирования телефона. При этом разрабатываемая модель функционирования телефона 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 CPmodel.mdl.

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**;
- с помощью операции главного меню **Browse/Use Case Diagram**.

При этом появляется новое окно с чистым рабочим листом диаграммы вариантов использования и специальная панель инструментов, содержащая кнопки с изображением графических элементов, необходимых для разработки диаграммы вариантов использования. Назначение отдельных кнопок данной панели можно узнать из всплывающих подсказок или в литературе [5].

3.3.2. Добавление актера на диаграмму вариантов использования и редактирование его свойств

Для добавления актера на диаграмму *варианта использования* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы актера на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение актера с маркерами изменения его геометрических размеров и предложенным программой именем по умолчанию NewClass. Для разрабатываемой модели телефона предложенное программой имя актера следует изменить на Пользователь (рис. 3.7).

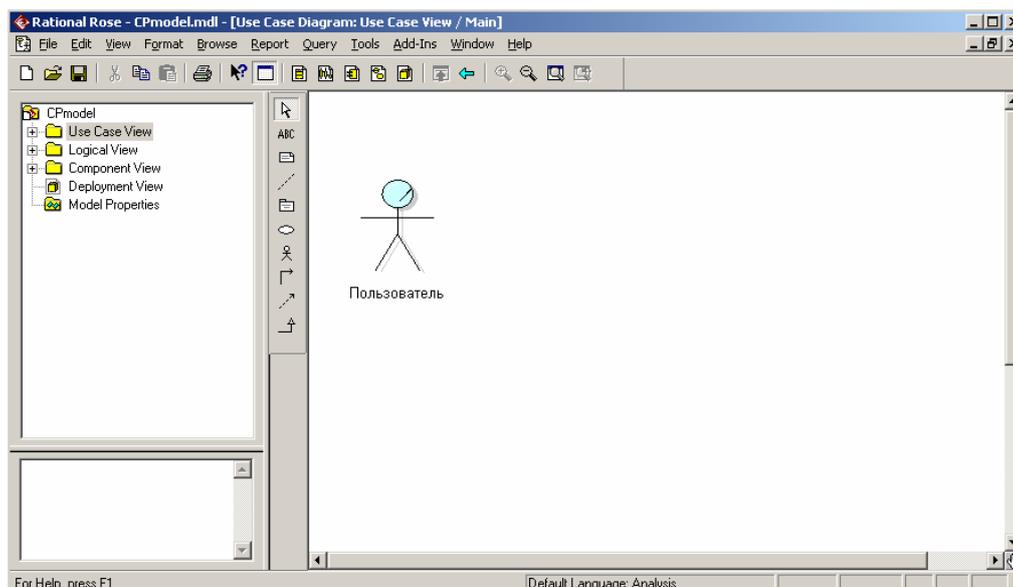


Рис. 3.7. Диаграмма вариантов использования после добавления на нее актера

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**. В этом случае появляется дополнительное диалоговое окно со специальными вкладками, в поля ввода которых можно занести всю информацию по данному элементу. Для добавленного актера Пользователь окно спецификации свойств выглядит следующим образом (рис. 3.8).

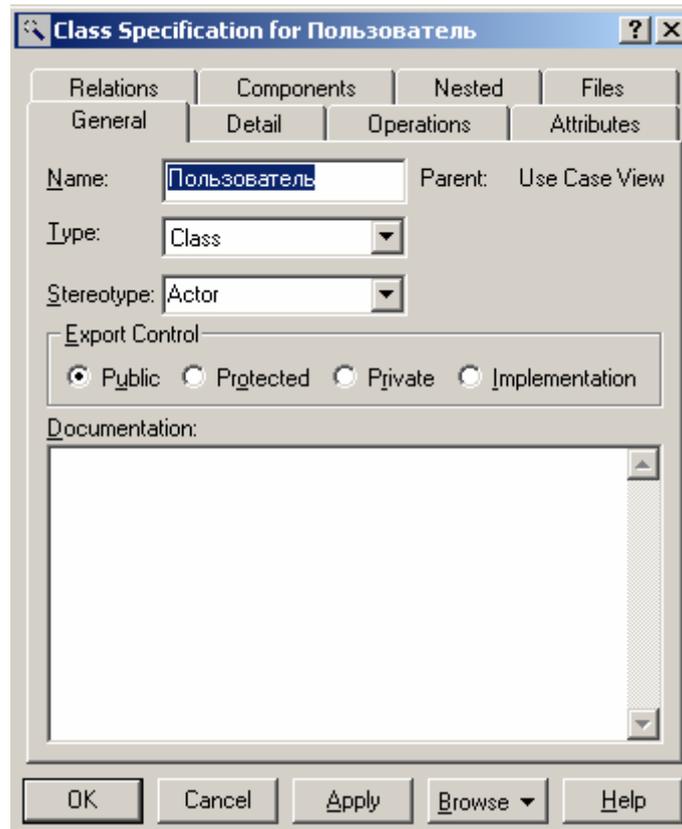


Рис. 3.8. Диалоговое окно спецификации свойств актера Пользователь

Для актера Пользователь 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**. Для добавления текста документации в секцию **Documentation** следует ввести текст: «Любое физическое лицо, пользующееся услугами мобильного телефона» и нажать кнопку **Apply** или **OK**. После изменения данных свойств актера Пользователь окно спецификации свойств будет выглядеть следующим образом (рис. 3.9).

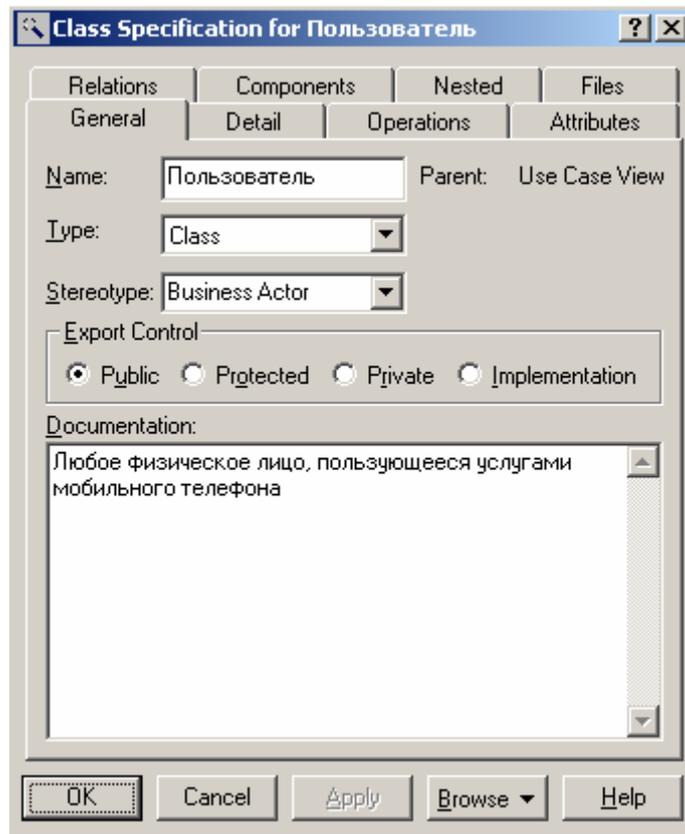


Рис. 3.9. Диалоговое окно спецификации свойств после изменения стереотипа и добавления текста документации для актера Пользователь

3.3.3. Добавление и редактирование варианта использования

Для добавления *варианта использования* на диаграмму нужно с помощью левой кнопки мыши нажать кнопку с изображением *варианта использования* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте диаграммы. На диаграмме появится изображение *варианта использования* с маркерами изменения его геометрических размеров и предложенным программой именем по умолчанию NewUseCase [5]. Для разрабатываемой модели телефона предложенное программой имя *варианта использования* следует изменить на «Исходящее соединение» (рис. 3.10).

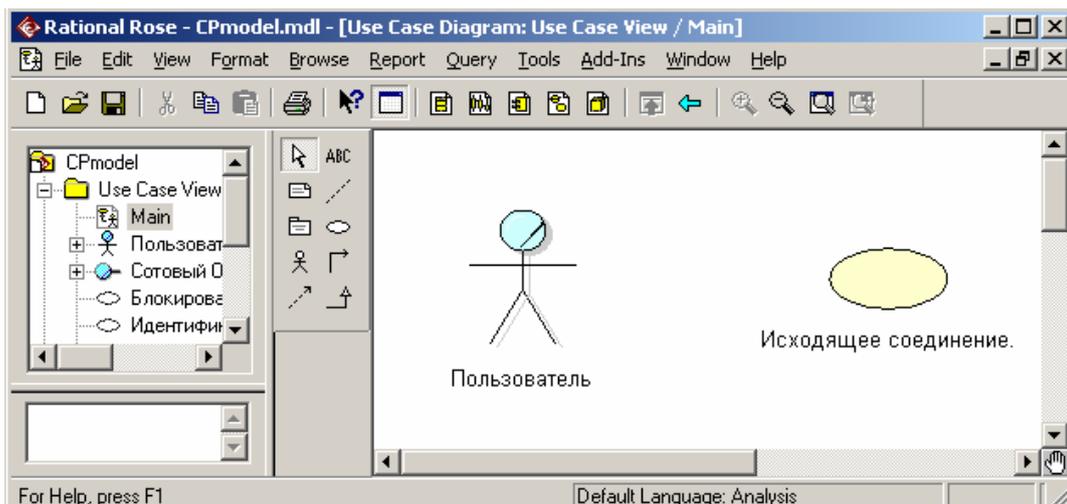


Рис. 3.10. Диаграмма вариантов использования после добавления на нее варианта использования

Для уточнения свойств данного *варианта использования* следует открыть диалоговое окно спецификации его свойств, например, с помощью двойного щелчка левой кнопкой мыши на изображении этого элемента на диаграмме. Для изменения *стереотипа* во вложенном списке **Stereotype** нужно выбрать строку **Business Use Case**. Для добавления текста документации в секцию **Documentation** следует ввести текст: «Основной вариант использования для разрабатываемой модели мобильного телефона» и нажать кнопку **Apply** или **OK**.

3.3.4. Добавление ассоциации

Для добавления *ассоциации* между актером и вариантом использования на диаграмму нужно с помощью левой кнопки мыши нажать на специальной панели инструментов кнопку с изображением пиктограммы направленной *ассоциации*, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении актера на диаграмме и отпустить ее на изображении *варианта использования*. В результате этих действий на диаграмме появится изображение *ассоциации*, соединяющей актера с вариантом использования (рис. 3.11).

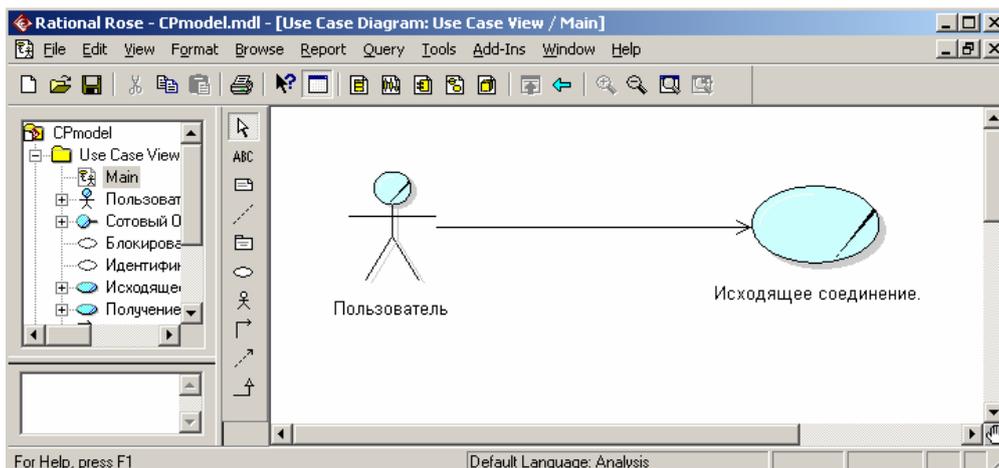


Рис. 3.11. Диаграмма вариантов использования после добавления на нее направленной ассоциации

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** на вкладке **Role A Detail**.

3.3.5. Добавление отношения зависимости и редактирование его свойств

Для добавления отношения зависимости между двумя вариантами использования на диаграмму необходимо предварительно рассмотренным выше способом добавить второй вариант использования с именем Идентификация Пользователя. 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 Исходящее соединение и отпустить ее на изображении варианта использования Идентификация Пользователя. В результате этих действий на диаграмме появится изображение отношения зависимости, которое соединяет два выбранных варианта использования [5].

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

После задания для данного отношения *зависимости стереотипа* <<include>> текст этого *стереотипа* в угловых скобках появится рядом с изображением пунктирной линии зависимости, связывающей соответствующие варианты использования (рис. 3.12). С целью лучшей визуализации диаграммы текстовую область *стереотипа* можно переместить в нужное место диаграммы.

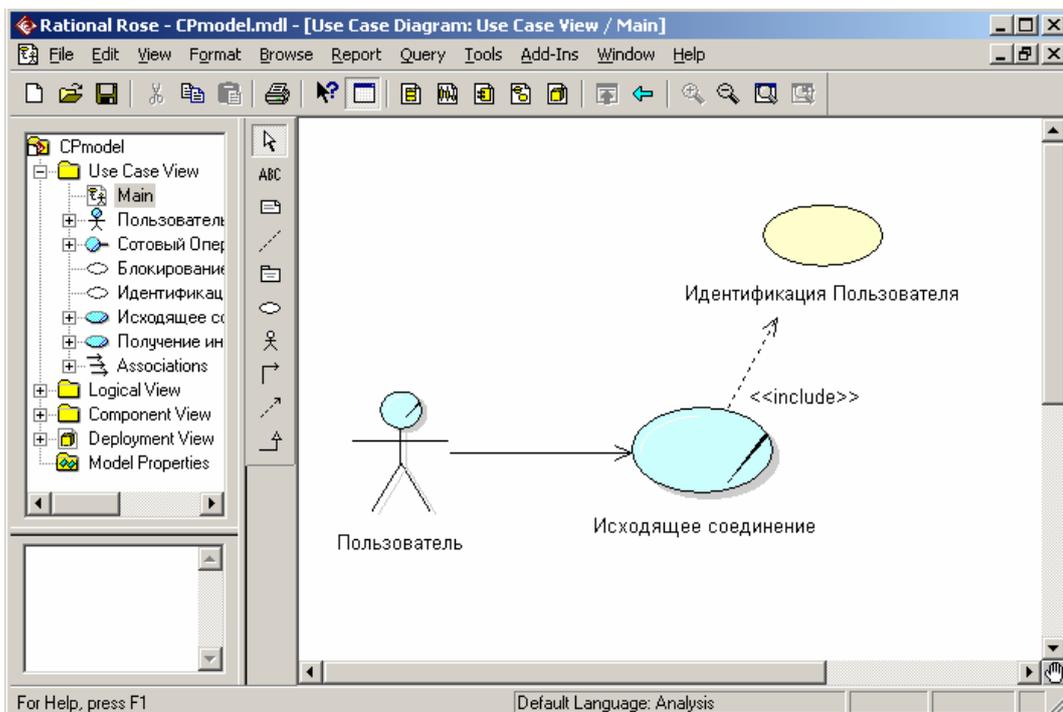


Рис. 3.12. Диаграмма вариантов использования после добавления на нее отношения зависимости

Аналогичным образом могут быть добавлены на диаграмму вариантов использования отношения зависимости со стереотипом <<extend>>, которые

применяются для моделирования исключений при выполнении отдельных *вариантов использования*.

3.3.6. Окончательное построение диаграммы вариантов использования

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

1. Добавить актера с именем Сотовый Оператор, для которого выбрать стереотип **Service**, означающий, что мобильный телефон использует некоторые услуги Сотового Оператора в качестве *сервиса*.

2. Добавить *вариант использования* Получение информации о состоянии счета, для которого выбрать *стереотип Business Use Case*.

3. Добавить *вариант использования* Блокирование SIM (Subscriber Identity Module) Карты Телефона.

4. Добавить направленную *ассоциацию* от бизнес-актера Пользователь к *варианту использования* Получение информации о состоянии счета.

5. Добавить направленную *ассоциацию* от *варианта использования* Исходящее соединение к *сервису* Сотовый Оператор.

6. Добавить направленную *ассоциацию* от *варианта использования* Получение информации о состоянии счета к *сервису* Сотовый Оператор.

7. Добавить отношение зависимости со стереотипом <<include>>, направленное от *варианта использования* Получение информации о состоянии счета к *варианту использования* Идентификация Пользователя.

8. Добавить отношение зависимости со стереотипом <<extend>>, направленное от *варианта использования* Блокирование SIM Карты Телефона к *варианту использования* Идентификация Пользователя.

Построенная таким образом диаграмма вариантов использования будет иметь следующий вид (рис. 3.13). Отношение зависимости со стереотипом <<extend>> на данной диаграмме означает следующее. Вариант использования Блокирование SIM Карты Телефона будет выполняться только в том случае, если в результате Идентификации Пользователя будет установлено, что соответствующая SIM карта утрачена ее владельцем или признана недействительной.

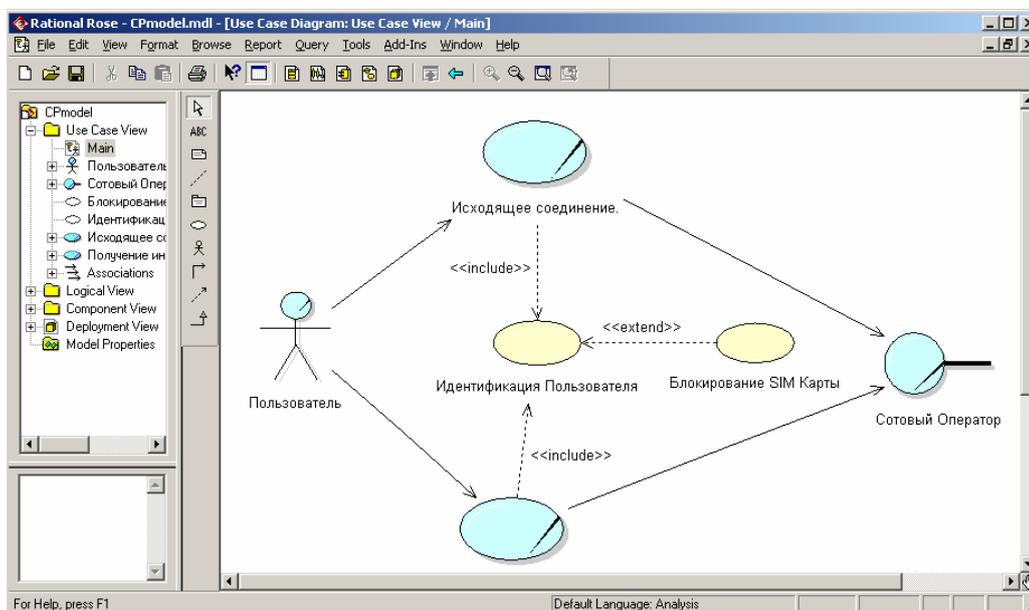


Рис. 3.13. Окончательный вид диаграммы вариантов использования для разрабатываемой модели мобильного телефона

Следует отметить, 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** или комбинацией клавиш **Ctrl+D**.

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

3.4. Разработка диаграммы классов и редактирование их свойств

3.4.1. Особенности разработки диаграмм классов

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 [5]:

- 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**;
- выполнить операцию главного меню: **Browse/Class Diagram**.

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*. Назначение отдельных кнопок панели можно узнать из всплывающих подсказок или в литературе [5].

3.4.2. Добавление класса на диаграмму классов и редактирование его свойств

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

Продолжая разработку модели мобильного телефона 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* CP, а имя добавленного на диаграмму *класса* - на Транзакция Телефона (рис. 3.14).

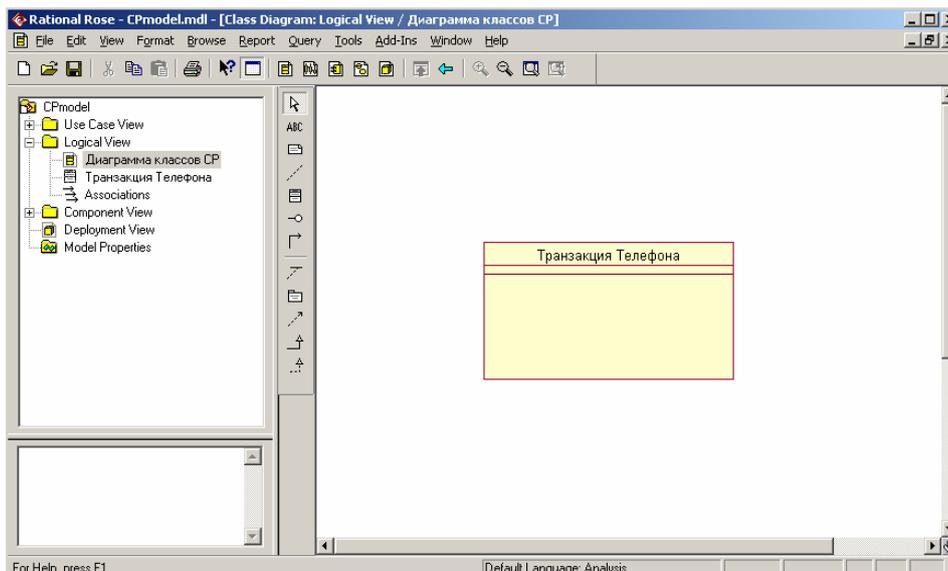


Рис. 3.14. Диаграмма классов модели телефона после добавления на нее класса Транзакция Телефона

Поскольку разрабатываемая модель мобильного телефона 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 записываются на русском языке с пробелами и записываются символами кириллицы. В последующем по мере выполнения проекта и реализации модели на некотором языке программирования, имена соответствующих классов, атрибутов и операций должны быть преобразованы в символы латиницы. При этом имена этих элементов модели должны быть записаны без пробелов. В контексте управляемой моделью архитектуры первую модель еще называют независимой от платформы реализации, а вторую - зависимой от платформы реализации.

Для класса Транзакция Телефона 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. 3.15) и на вкладке **General** выбрать из вложенного списка **Stereotype** стереотип **entity**.

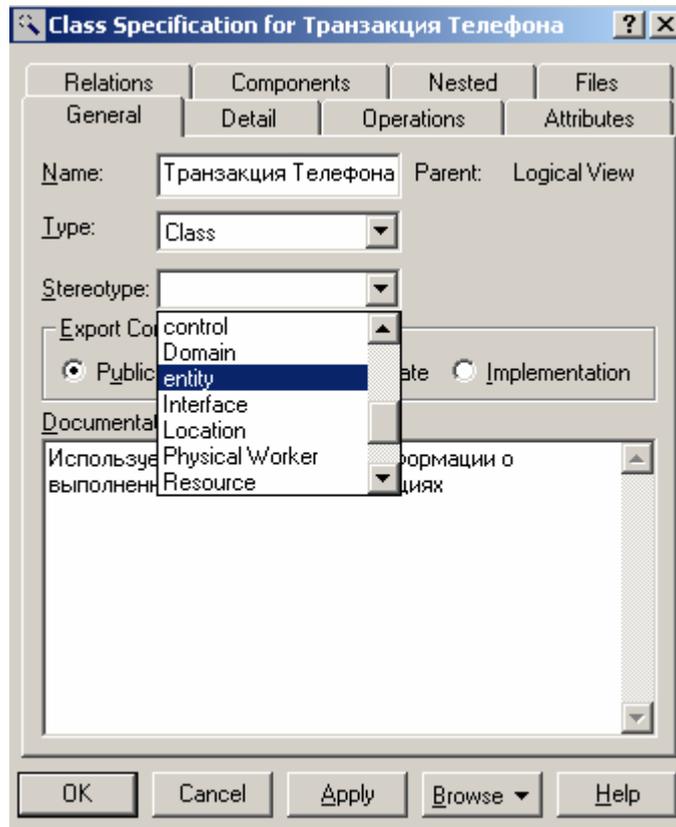


Рис. 3.15. Диалоговое окно спецификации свойств класса *Транзакция Телефона* при выборе из вложенного списка стереотипа *entity*

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 телефоном транзакциях" и нажать кнопку **Apply** или **OK**, чтобы сохранить результаты редактирования свойств выбранного *класса*. После назначения стереотипа классу *Транзакция Телефона* текст данного стереотипа в угловых скобках появится выше имени данного *класса* (рис. 3.16).

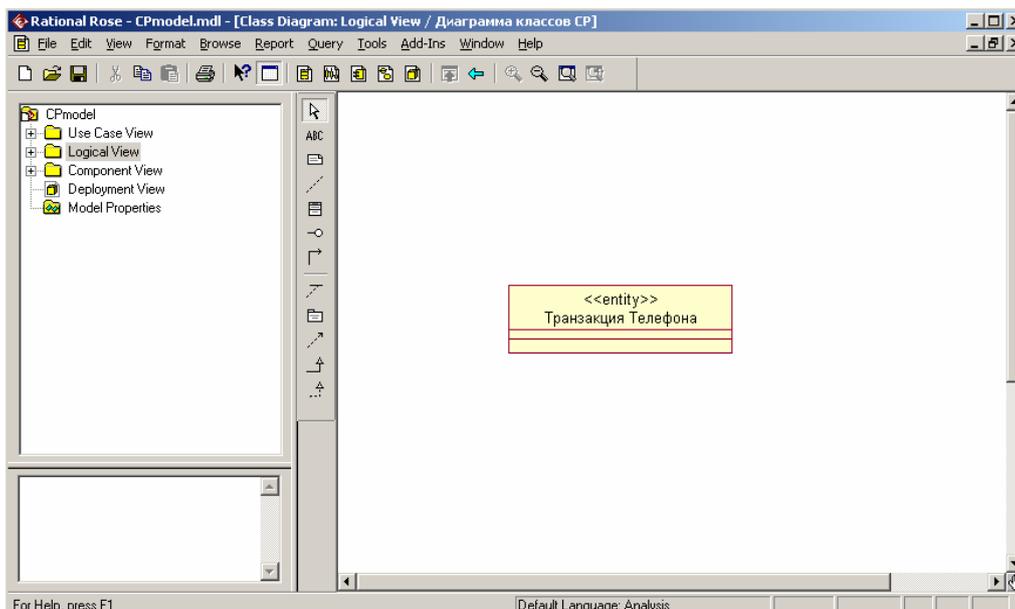


Рис. 3.16. Диаграмма классов модели мобильного телефона после выбора стереотипа для класса Транзакция Телефона

Для отдельного класса можно уточнить также и другие его свойства, доступные для редактирования на вкладке **Detail** окна спецификации свойств этого класса. Например, на этой вкладке с помощью вложенного списка **Multiplicity** можно задать количество объектов или экземпляров данного класса, для чего следует выбрать строку с буквой *n*. Данное значение означает, что у класса Транзакция телефона может быть любое конечное число экземпляров (рис. 3.17). Поле ввода с именем **Space** служит для указания объема абсолютной или относительной памяти, которая требуется, по оценке разработчика, для реализации каждого объекта данного класса. Применительно к рассматриваемой модели это поле можно оставить пустым.

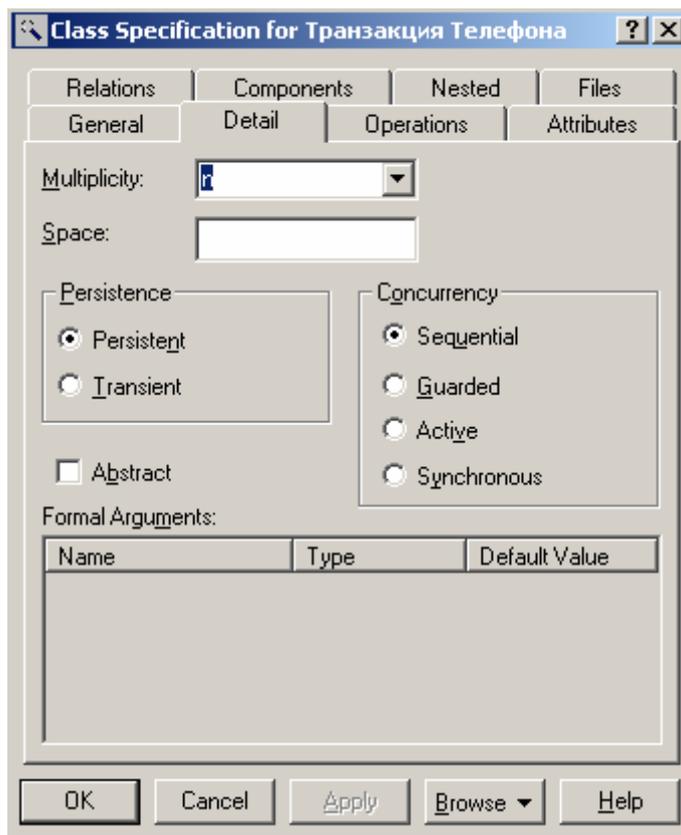


Рис. 3.17. Диалоговое окно спецификации свойств класса Транзакция Телефона, открытое на вкладке Detail

Далее можно задать устойчивость классов в группе выбора **Persistence**. При этом выбор свойства **Persistent** означает, что информация об объектах данного класса должна быть сохранена в системе. Выбор свойства **Transient** 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** можно специфицировать условия на возможность реализации объектов данного класса в параллельных потоках управления. Для выбора могут быть использованы следующие свойства [5]:

- **Sequential** - 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 после-

довательно. В то же время при наличии нескольких потоков управления стабильное поведение объектов *класса* не гарантируется.

- **Guarded** - 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** - означает, что *класс* должен иметь свой собственный поток управления.

- **Synchronous** - 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**. Применительно к рассматриваемой модели для *класса* Транзакция телефона 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** или нажать комбинацию клавиш: **Ctrl+S**.

3.4.3. 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**. В качестве примера можно представить изображение *класса* Транзакция Телефона в форме специальной графической пиктограммы стереотипа. С этой целью следует выполнить операцию контекстного меню: **Options/Stereotype Display/Icon**. Соответствующее графическое изображение стереотипа <<entity>> для *класса* Транзакция Телефона в форме пиктограммы будет иметь следующий вид (рис. 3.18, слева).

Для сравнения можно выбрать изображение *класса* Транзакция Телефона в форме декоративного графического стереотипа. С этой целью необходимо выполнить операцию контекстного меню: **Options/Stereotype Display/Decoration**. Соответствующее графическое изображение стереотипа <<entity>> для *класса* Транзакция Телефона в форме декорации будет иметь следующий вид (рис. 3.18, справа).

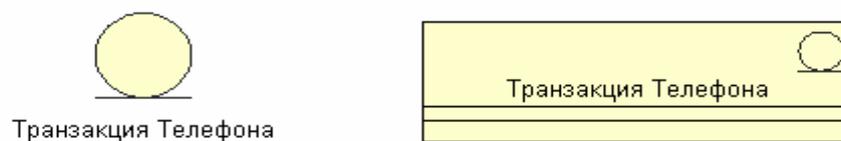


Рис. 3.18. Графические способы изображения стереотипа <<entity>> для класса Транзакция Телефона

Изменить изображение стереотипов одновременно для нескольких *классов* диаграммы можно с помощью одной из вложенных операций главного меню: **Format/Stereotype Display**. 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** или с помощью комбинации клавиш **Ctrl+A**.

Продолжая разработку модели телефона, добавим на диаграмму второй *класс* с именем Контроллер Телефона, для которого в окне спецификации свойств выберем стереотип **control**, а в качестве документации введем текст: "Реализует логику функционирования телефона". При этом атрибуты и операции у данного *класса* будут отсутствовать.

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

Далее следует добавить *класс* с именем IКонтроллер Оператора, для которого выбрать стереотип **Interface**, означающий, что телефон пользуется услугами Оператора 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 SIM карты и IКонтроллер Оператора будет иметь следующий вид (рис. 3.19).

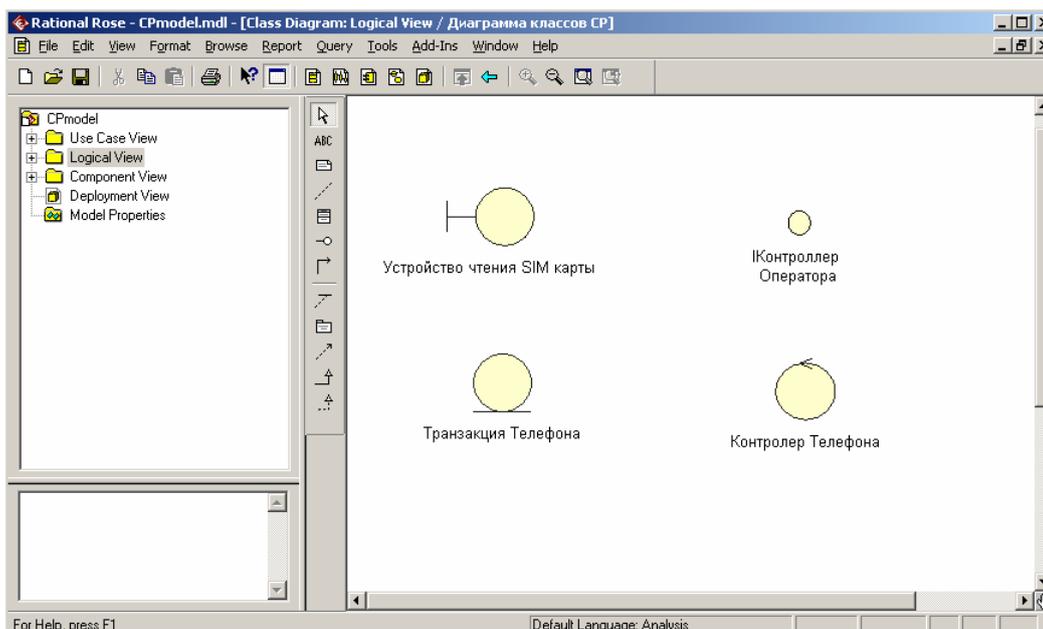


Рис. 3.19. Фрагмент диаграммы классов модели телефона после добавления на нее классов *Устройство чтения SIM карты* и *Контролер Телефона*

3.4.4. 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 [5]:

- С помощью *операции* контекстного меню **New Attribute** 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/Attribut** 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**, 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. 3.20).

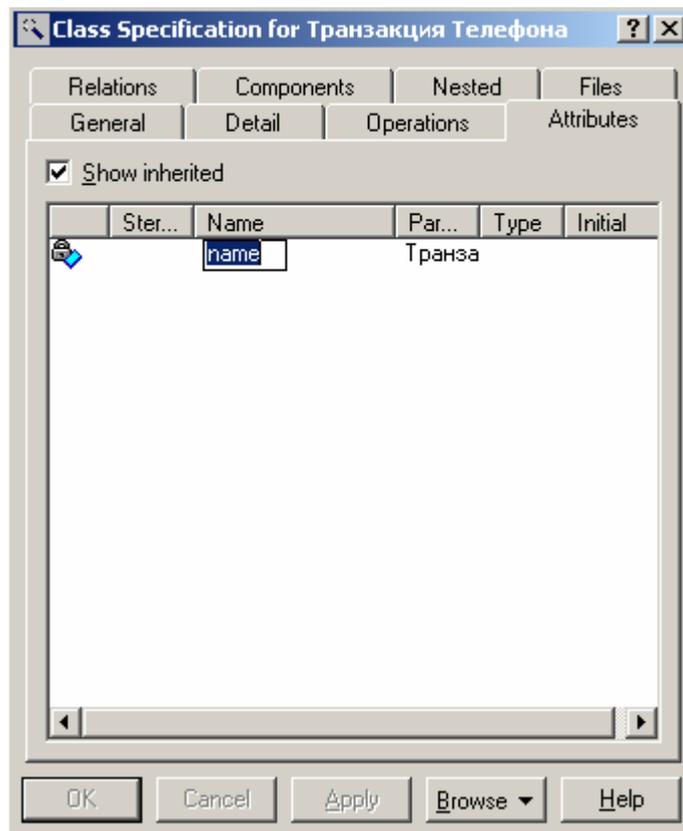


Рис. 3.20. Диалоговое окно спецификации свойств класса после добавления нового атрибута

Для рассматриваемой модели телефона имя добавленного *атрибута* следует изменить на номер телефона. Напомним, что имена *атрибутов* и *операций* классов должны начинаться со строчной буквы. Видимость *атрибутов* на диаграмме классов изображается в форме специальных пиктограмм или ук-

рашений. Используемые пиктограммы видимости изображаются перед именем соответствующего *атрибута* и имеют следующий смысл (табл. 3.1) [5].

Таблица 3.1. Пиктограммы видимости атрибутов классов

Графическое изображение	Текстовый аналог	Назначение пиктограммы
	Public	Общедоступный или открытый. В нотации языка UML такому <i>атрибуту</i> соответствует знак «+»
	Protected	Защищенный. В нотации языка UML такому <i>атрибуту</i> соответствует знак «#»
	Private	Закрытый. В нотации языка UML такому <i>атрибуту</i> соответствует знак «-»
	Implementation	Реализация. В нотации языка UML такому <i>атрибуту</i> 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* номер телефона в качестве *типа его допустимых значений* из вложенного списка **Type** следует выбрать тип **Integer**, а для задания *квантора видимости* следует выбрать в группе **Export Control** *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 считывает значение этого *атрибута* с SIM карты» и нажать кнопку **Apply** или **OK**. Соответствующее окно спецификации свойств *атрибута* идентификационный номер после редактирования его общих свойств будет иметь следующий вид (рис. 3.21).

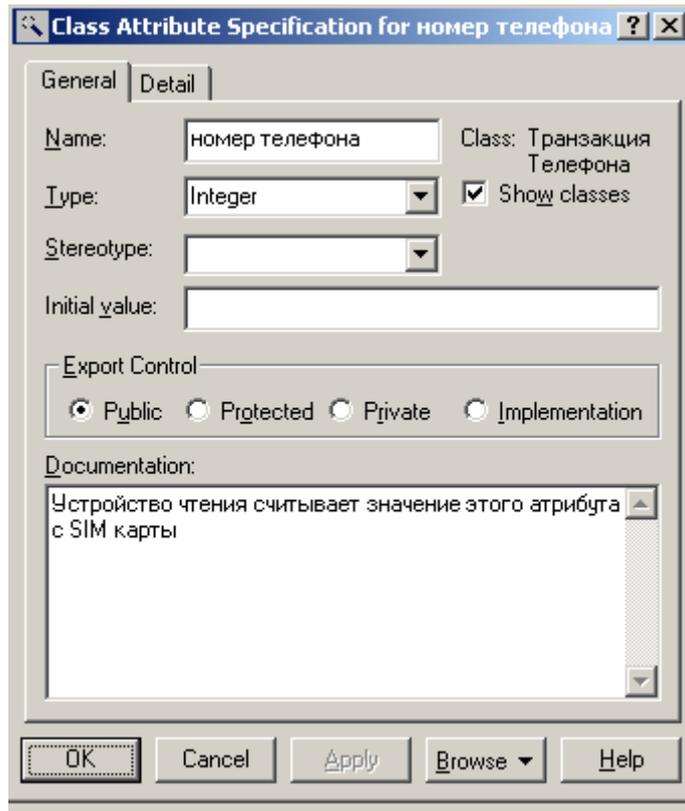


Рис. 3.21. Диалоговое окно спецификации свойств атрибута номер телефона после его редактирования

Для отдельного атрибута можно также определить дополнительные свойства, доступные для редактирования на вкладке **Detail** диалогового окна спецификации свойств выбранного атрибута (рис. 3.22).



Рис. 3.22. Диалоговое окно спецификации свойств атрибута номер телефона, открытое на вкладке *Detail*

На вкладке **Detail** в группе выбора **Containment** можно специфицировать условия хранения *атрибута* у объектов выбранного класса. Для выбора могут быть использованы следующие свойства [5]:

- **By value** - 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** - означает, что значение *атрибута* хранится вне адресного пространства, выделенного для объекта данного класса, но у объектов класса имеется указатель на этот атрибут.

- **Unspecified** - 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 то же 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.

3.4.5. Добавление и редактирование операций классов

Функционирование телефона основано на выполнении отдельными его устройствами тех или иных действий. В модели структуры телефона 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 [5]:

- С помощью *операции* контекстного меню **New Operation** 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** 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**, 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 следующий смысл (табл. 3.2) [5].

Таблица 3.2. Пиктограммы видимости операций классов

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

В контексте рассматриваемой модели телефона в качестве имени первой *операции* для класса Транзакция Телефона следует задать: создать новое соединение. 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 имени *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 новое соединение() в качестве *квантора видимости* следует выбрать из вложенного списка *квантор public*. Соответствующее окно спецификации

свойств *операции* создать новое соединение() после редактирования ее свойств будет иметь следующий вид (рис. 3.23).

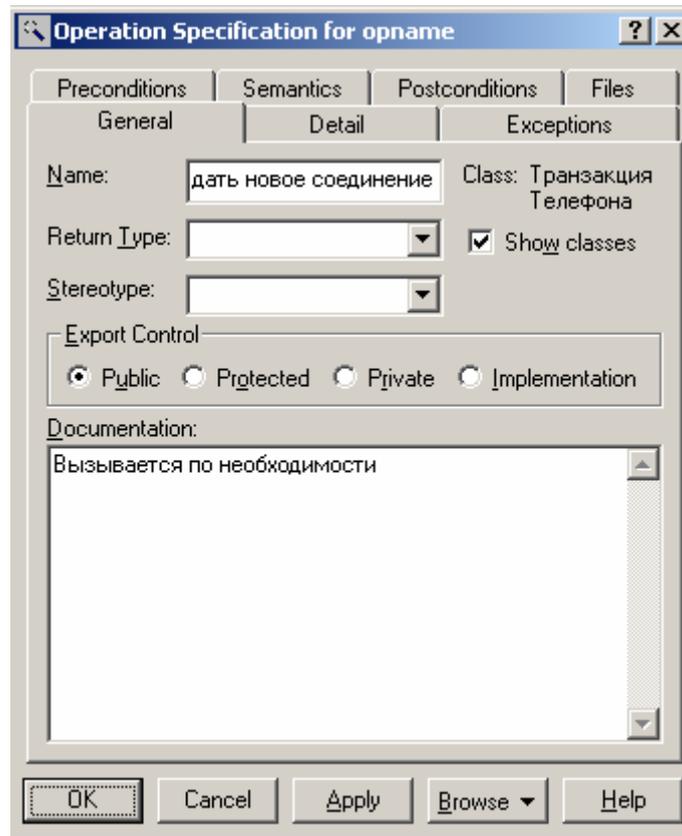


Рис. 3.23. Диалоговое окно спецификации свойств операции создать новое соединение()

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** диалогового окна спецификации свойств выбранной *операции* (рис. 3.24).

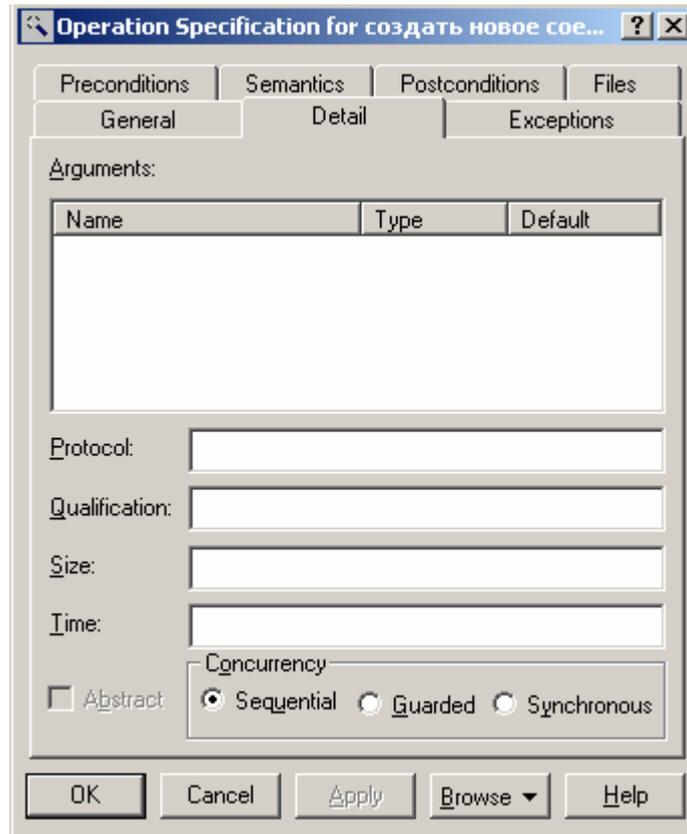


Рис. 3.24. Диалоговое окно спецификации свойств операции *создать новое соединение()*, открытое на вкладке *Detail*

На вкладке **Detail** в многостраничном поле **Arguments** можно определить аргументы редактируемой операции. Для этого следует выполнить операцию контекстного меню **Insert**. 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** 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** 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** и **Time** можно специфицировать предполагаемый объем памяти и время, необходимое для выполнения *операции*. Соответствующая информация попадает в генерируемый код в форме комментария.

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

- **Sequential** - 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** - 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** - 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 новое соединение() следует выбрать свойство **Sequential**, а поля всех других свойств оставить пустыми.

3.4.6. Спецификация атрибутов и операций для класса Транзакция Телефона

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

1. Идентификационный номер с *квантором видимости public*. В качестве типа этого *атрибута* следует выбрать тип **Integer**.

2. Пользовательский ПИН-код с *квантором видимости public*. В качестве типа этого *атрибута* следует выбрать тип **Integer**, а в секцию документации *атрибута* ввести поясняющий текст: «Значение этого *атрибута* вводится пользователем с клавиатуры телефона».

3. Введенный номер или код операции с *квантором видимости public*. В качестве типа этого *атрибута* следует выбрать тип **Currency**, а в секцию документации *атрибута* ввести поясняющий текст: «Значение этого *атрибута* вводится пользователем с клавиатуры телефона».

4. Проверить правильность ПИН-кода () с *квантором видимости public*. В качестве *типа возвращаемого результата* для этой *операции* следует выбрать тип **Boolean**, а в секцию ее документации ввести поясняющий текст: «Вызывается после того, как пользователь ввел значение ПИН-кода с клавиатуры телефона».

5. Завершить соединение () с *квантором видимости public*. В секцию ее документации ввести поясняющий текст: «Вызывается после завершения всех действий телефона по обслуживанию пользователя».

Соответствующий фрагмент диаграммы классов после добавления и спецификации *атрибутов* и *операций* для класса Транзакция Телефона будет иметь следующий вид (рис. 3.25).

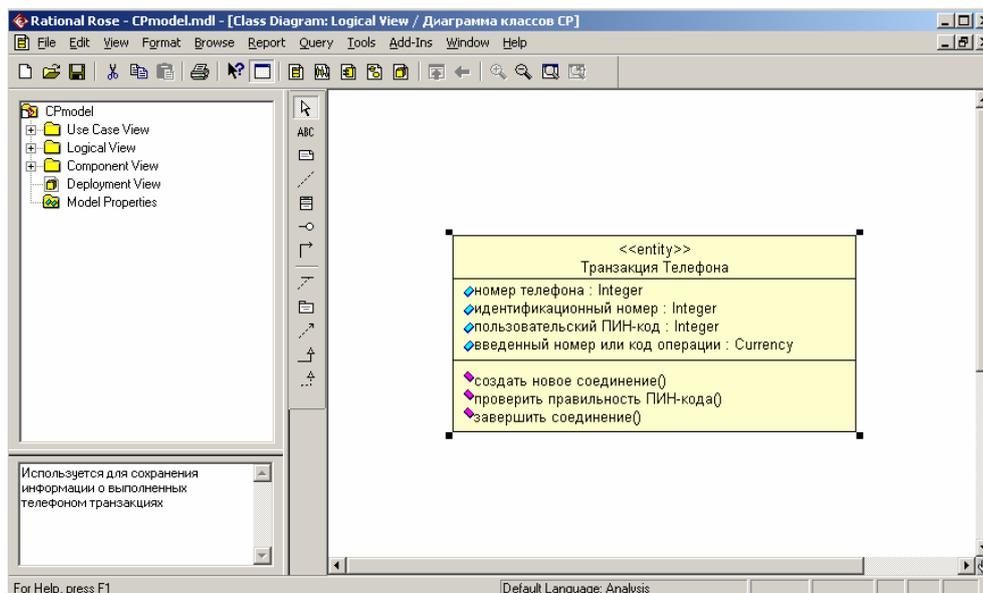


Рис. 3.25. Фрагмент диаграммы классов модели телефона после добавления атрибутов и операций для класса Транзакция телефона

3.5. 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 [5].

3.5.1. 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 диа-

грамме 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 телефона, добавим на нее описанным способом направленную *ассоциацию* между классом Контроллер Телефона и классом Транзакция Телефона (рис. 3.26).

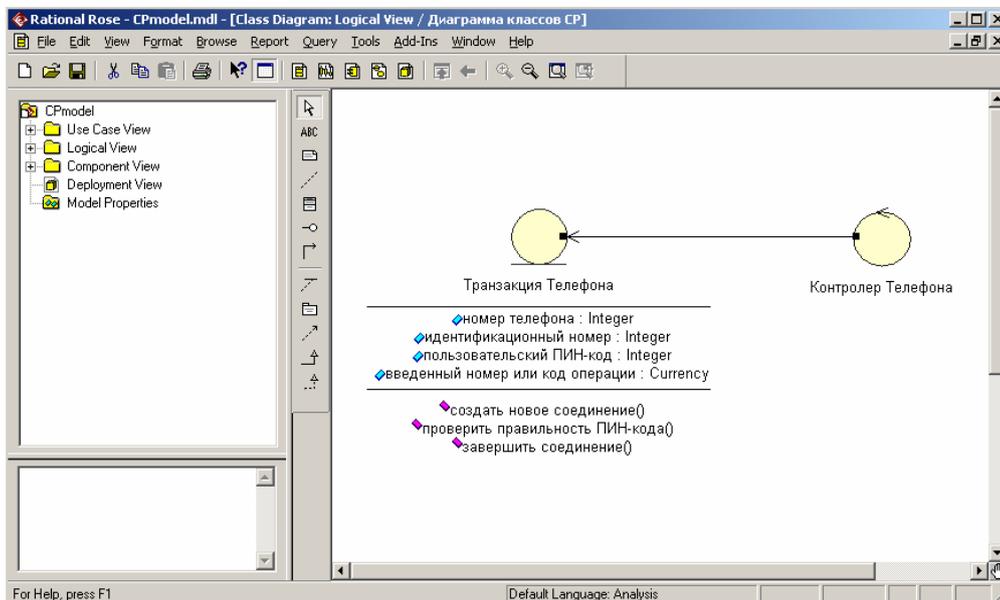


Рис. 3.26. Фрагмент диаграммы классов модели телефона после добавления на неё направленной ассоциации

Изменим имя для данной *ассоциации*, предложенное средой по умолчанию. Это можно выполнить с помощью окна спецификации свойств *ассоциации* **Association Specification** после выделения линии *ассоциации* на диаграмме классов или в браузере проекта и двойного щелчка на ней левой кнопки мыши (рис. 3.27).

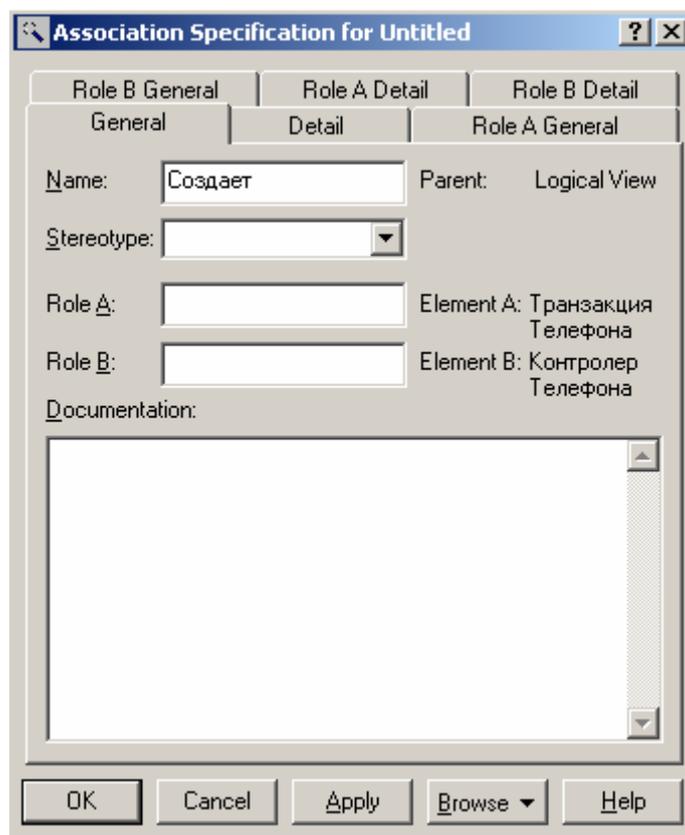


Рис. 3.27. Диалоговое окно спецификации свойств ассоциации

Для задания имени *ассоциации* следует на вкладке **General** в поле ввода **Name** ввести текст 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 Телефона, 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 Телефона 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 Телефона может быть связан с одним или несколькими объектами класса Транзакция Телефона.

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**.

3.5.2. 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**.

В качестве примера изменим тип, созданной ранее *ассоциации* и сделаем ее агрегацией. Содержательно это будет означать, что класс Контроллер Телефона будет включать в себя в качестве составной части класс Транзакция

Телефона; при этом уничтожение любого объекта класса Контроллер Телефона не должно привести к уничтожению ассоциированных с ним объектов класса Транзакция Телефона. С этой целью на вкладке **Role B Detail** деталей конца *ассоциации* класса Контроллер Телефона следует выставить отметку в строке выбора **Aggregate** (рис. 3.28).

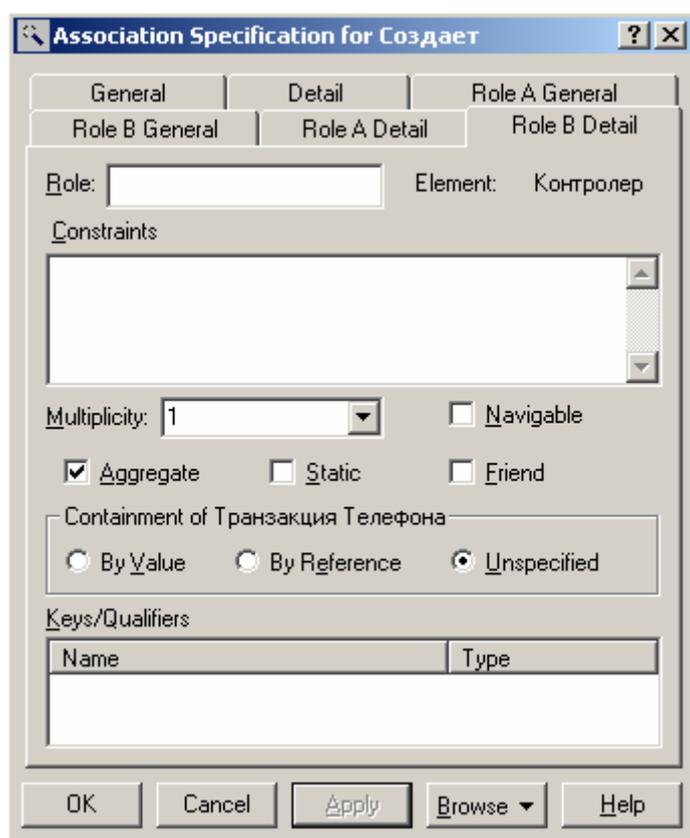


Рис. 3.28. Диалоговое окно спецификации свойств ассоциации

Соответствующий фрагмент диаграммы классов после изменения *ассоциации* между классами Контроллер Телефона и Транзакция Телефона на отношение *агрегации* будет иметь следующий вид (рис. 3.29).

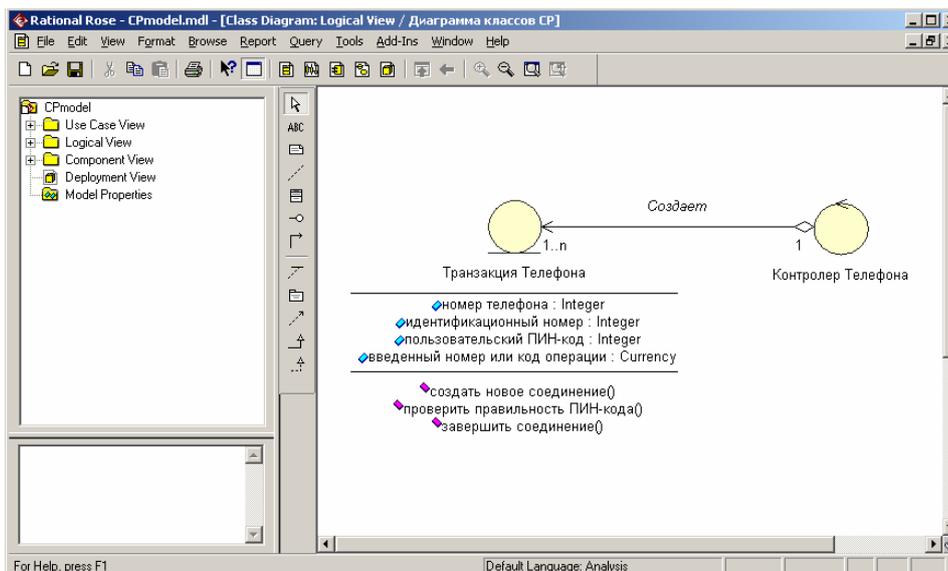


Рис. 3.29. Фрагмент диаграммы классов модели телефона после добавления на нее отношения агрегации

Для изображения отношения *композиции* можно также вначале изобразить обычную *ассоциацию*, после чего, открыв окно ее свойств на вкладке деталей соответствующего конца *ассоциации*, выставить отметку в строке выбора **Aggregate** и в секции **Containment** выбрать опцию **By Value**. По умолчанию эта опция не определена, т.е. выставлена отметка опции **Unspecified**.

3.5.3. 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 телефона, добавим на нее описанным способом направленную *ассоциацию* между классом Контроллер Телефона и дополнительно созданным абстрактным классом Контроллер (рис. 3.30). Последний класс может быть предназначен для спецификации системных атрибутов и операций, необходимых при выполнении соответствующей программы. Напомним, что на абстрактный характер класса указывает написание курсивом его имени, а для спецификации данного свойства класса необходимо на вкладке **Detail** окна спецификации свойств класса Контроллер выставить отметку в строке выбора **Abstract**.

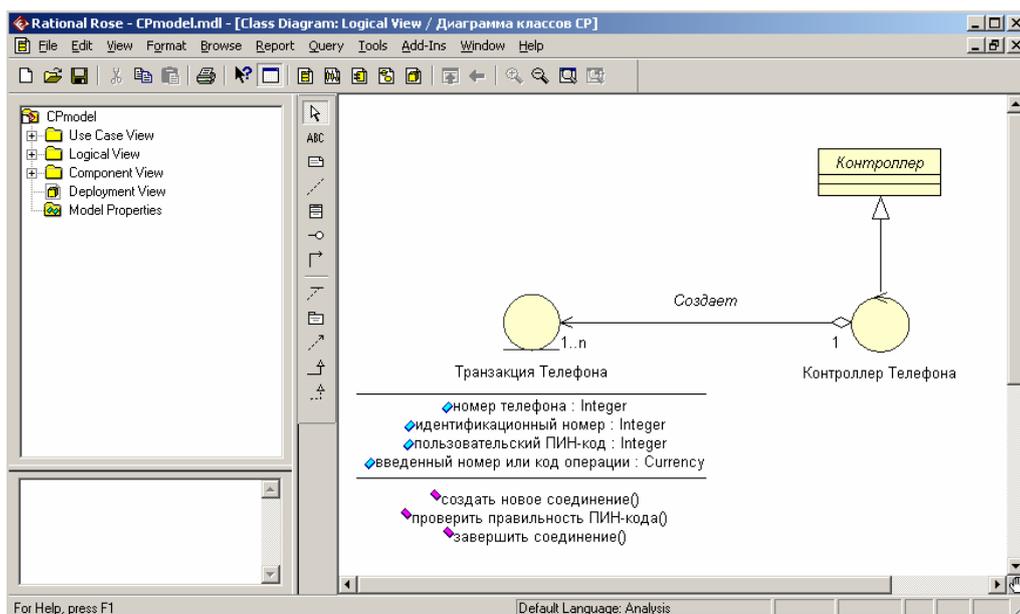


Рис. 3.30. Диаграмма классов модели телефона после добавления на неё отношения обобщения

Изменить имя отношения *обобщения*, предложенное средой по умолчанию можно с помощью окна спецификации свойств *обобщения*. Доступ к диалоговому окну спецификации свойств отношения *обобщения* **Generalize Specification** можно получить после выделения линии *обобщения* на диа-

грамме классов или в браузере проекта и двойного щелчка на ней левой кнопки мыши (рис. 3.31).

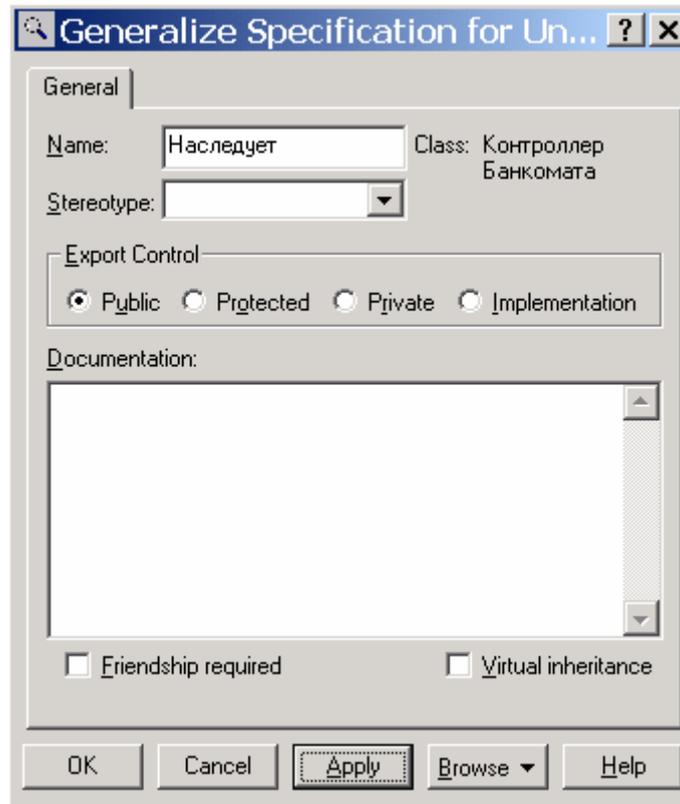


Рис. 3.31. Диалоговое окно спецификации свойств отношения обобщения

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

3.5.4. Окончательное построение диаграммы классов модели телефона

Для окончательного построения диаграммы классов рассматриваемой модели телефона 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 `IKонтроллер Оператора` добавить операцию: проверить идентификационный номер телефона (идентификационный номер: `Integer`) с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Boolean**, а в качестве целочисленного аргумента задать идентификационный номер. Для задания аргумента необходимо перейти на вкладку **Detail** 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 `IKонтроллер Оператора` добавить операцию: открыть счет клиента (идентификационный номер: `Integer`) с квантором видимости **public**. В качестве целочисленного аргумента этой операции следует задать идентификационный номер.

3. Для класса `IKонтроллер Оператора` добавить операцию: проверить баланс клиента (идентификационный номер: `Integer`) с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Boolean**. В качестве целочисленного аргумента этой операции следует задать идентификационный номер.

4. Для класса `IKонтроллер Оператора` добавить операцию: уменьшить счет клиента (идентификационный номер: `Integer`) с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Boolean**. В качестве целочисленного аргумента этой операции следует задать идентификационный номер.

5. Для класса `Устройство чтения SIM карты` добавить операцию: прочитать идентификационный номер() с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Integer**.

6. Для класса Устройство чтения SIM карты добавить операцию: прочитать ПИН-код () с квантором видимости **public**. В качестве типа возвращаемого результата для этой операции следует выбрать тип **Integer**.

7. Для класса Устройство чтения SIM карты добавить операцию: заблокировать SIM карту () с квантором видимости **public**. В секцию документации данной операции следует ввести поясняющий текст: «Вызывается после того, как установлен факт утраты SIM карты владельцем».

8. Добавить класс с именем Экран Телефона, 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 телефоне».

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

10. Для класса Экран Телефона добавить операции: показать баланс счета (), показать картинку(), показать видео ролик() с кванторами видимости **public**.

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

12. Для класса Клавиатура Телефона 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».

13. Для класса Клавиатура Телефона добавить операции: указать тип соединения (), ввести номер (), ввести текст

sms () с кванторами видимости **public**. В качестве типа возвращаемого результата для этих операций следует выбрать тип **Boolean**.

14. Добавить класс с именем Периферийное устройство, для которого выбрать стереотип **boundary**. В секцию документации данного класса следует ввести поясняющий текст: «Устанавливается на телефоне».

15. Добавить класс с именем Камера телефона, для которого выбрать стереотип **boundary**. В секцию документации данного класса следует ввести поясняющий текст: «Устанавливается на телефоне».

16. Для класса Камера телефона добавить операции: видео съемка () и фото съемка () с кванторами видимости **public**. В секцию документации данных операций следует ввести поясняющий текст: «Вызывается по дополнительному запросу клиента».

17. Добавить направленную *ассоциацию* от класса Контроллер Телефона к классу Устройство чтения SIM карты. В качестве *кратности* концов этой *ассоциации* установить значение 1.

18. Добавить направленную *ассоциацию* от класса Контроллер Телефона к классу Камера Телефона. В качестве *кратности* концов этой *ассоциации* установить значение 1.

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

20. Добавить направленную *ассоциацию* от класса Контроллер Телефона к классу Периферийное устройство. В качестве *кратности* концов этой *ассоциации* установить значение 1.

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

22. Добавить направленную ассоциацию от класса Контроллер Телефона 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.

Построенная в результате указанных действий диаграмма классов будет иметь следующий вид (рис. 3.32).

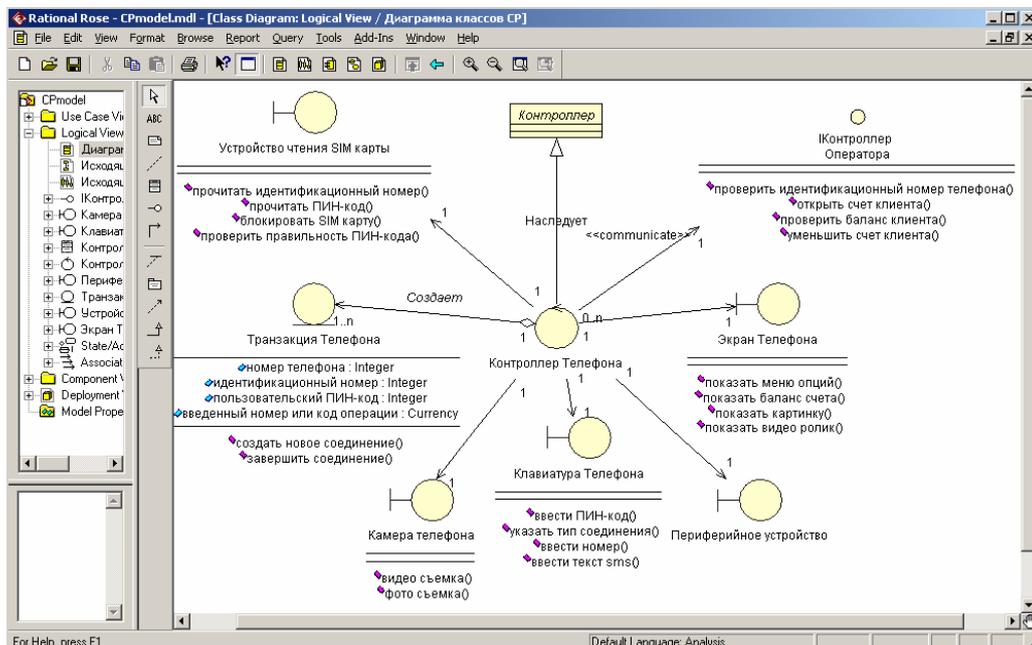


Рис. 3.32. Окончательный вид диаграммы классов для разрабатываемой модели телефона

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**.

3.6. Разработка диаграммы кооперации и редактирование свойств ее элементов

3.6.1. Особенности разработки диаграмм кооперации в среде IBM Rational Rose 2003

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 [5]:

- 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** и выбрать для построения новую диаграмму кооперации.
- Выполнить операцию контекстного меню: **New/Collaboration Diagram** 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. Назначение отдельных кнопок панели можно узнать из всплывающих подсказок или в литературе [5].

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 Исходящий вызов применительно к разрабатываемому проекту системы управления телефоном. 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: Исходящий вызов.

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>.

3.6.2. 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. 3.33).

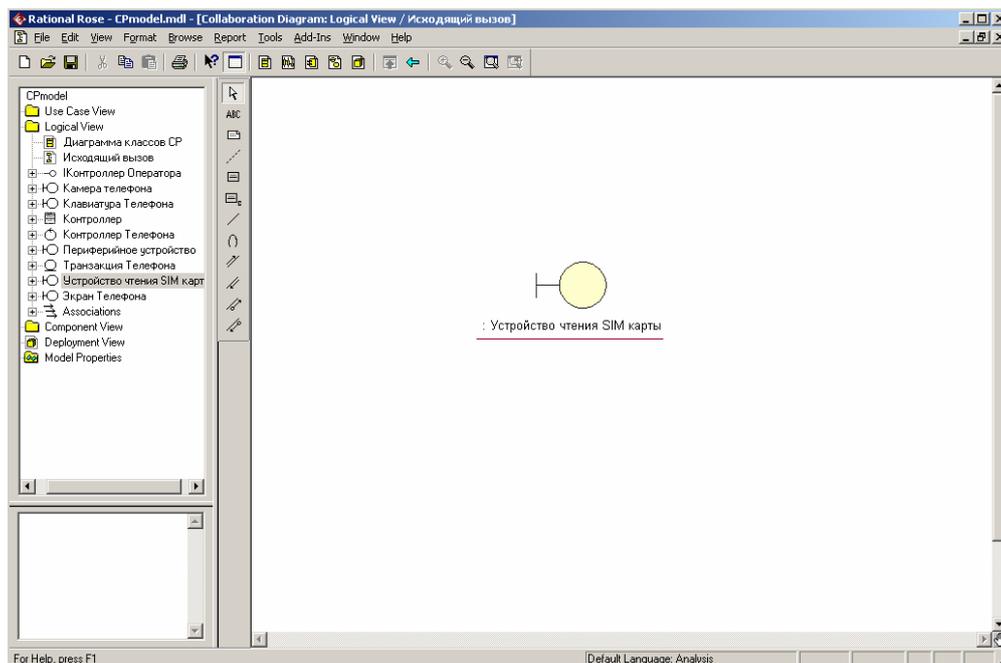


Рис. 3.33. Диаграмма кооперации после добавления на нее анонимного объекта класса Устройство чтения SIM карты

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. 3.34).

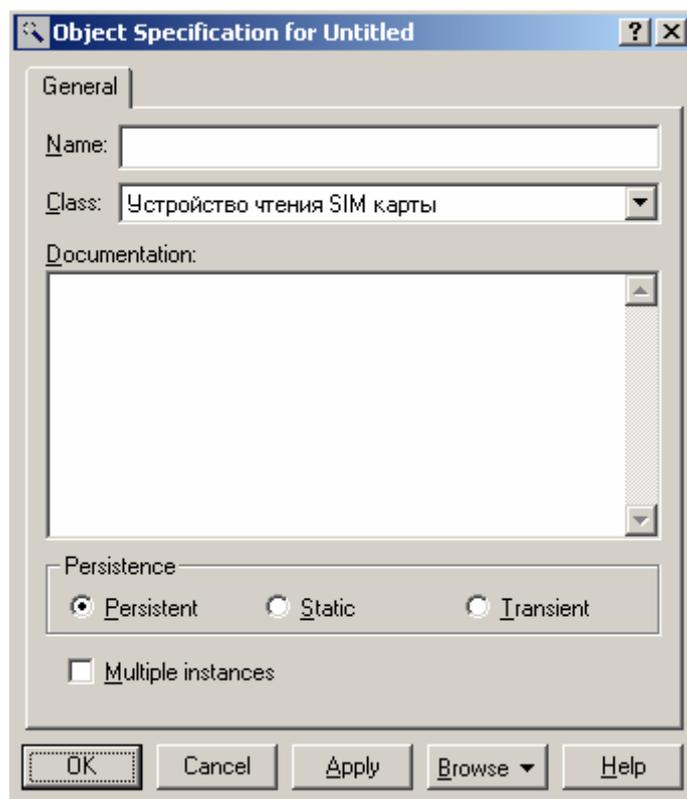


Рис. 3.34. Диалоговое окно спецификации свойств объекта класса Устройство чтения SIM карты

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* означает, что информация об *объектах* данного класса должна быть сохранена в системе некоторым подходящим способом. *Свойство Static* означает, что соответствующий *объект* сохраняется в памяти компьютера в течение всего времени работы программного приложения. *Свойство Transient* соответствующий *объект* хранится в памяти компьютера в течение короткого времени, необходимого только для выполнения его операций.

Применительно к рассматриваемой для *объекта* класса Устройство чтения SIM карты модели следует выбрать свойство **Persistent**.

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

3.6.3. Добавление связи и редактирование ее свойств

Для добавления *связи* между предварительно размещенными на диаграмме *объектами* нужно с помощью левой кнопки мыши нажать кнопку с изображением *связи* на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении одного *объекта* на диаграмме и отпустить ее на изображении другого *объекта*. В результате этих действий на диаграмме появится изображение *связи*, например, соединяющей *объект* класса Пользователь (актера) с объектом класса Устройство чтения SIM карты (рис. 3.35). Поскольку кнопка с изображением актера отсутствует на специальной панели инструментов диаграммы кооперации, соответствующий *объект* следует предварительно поместить на диаграмму способом перетаскивания пиктограммы актера из браузера проекта.

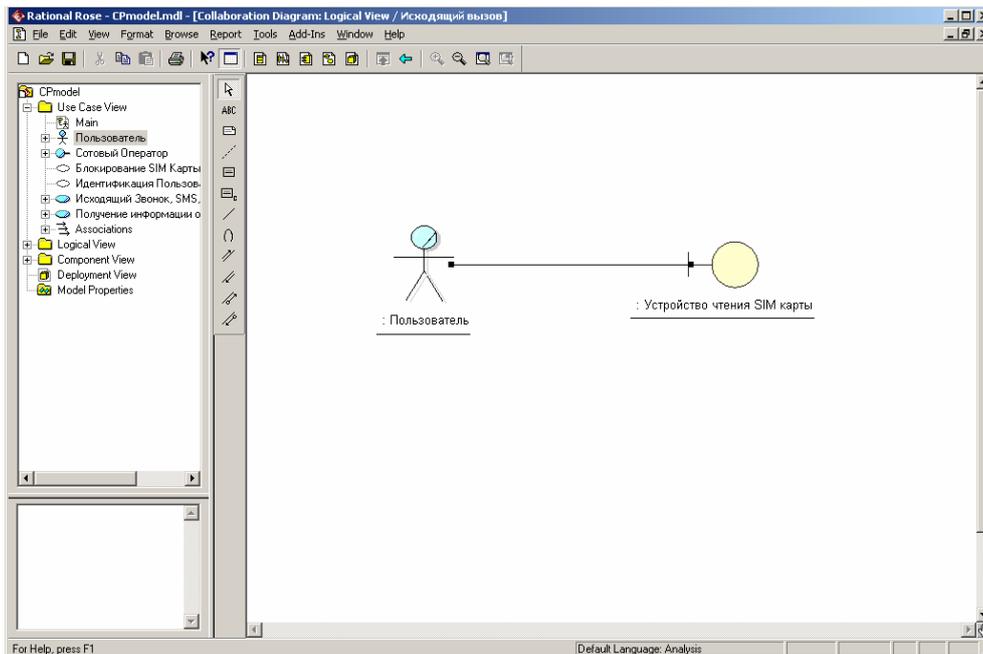


Рис. 3.35. Диаграмма кооперации после добавления связи между объектом класса Пользователь и объектом класса Устройство чтения SIM карты

По умолчанию каждая добавляемая связь считается анонимной. При необходимости можно задать имя связи с помощью диалогового окна спецификации свойств данной связи (рис. 3.36).

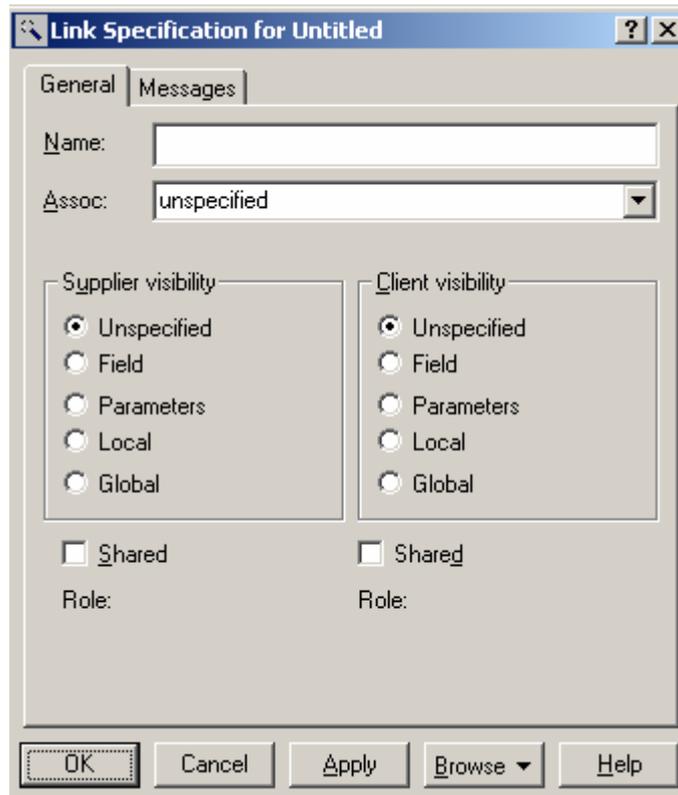


Рис. 3.36. Диалоговое окно редактирования свойств связи

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

3.6.4. 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* [5].

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**. После этого следует выполнить операцию контекстного меню **Insert To**, в результате чего появляется вложенный список с предложением выбрать одну из операций целевого класса для спецификации имени *сообщения* (рис. 3.37).

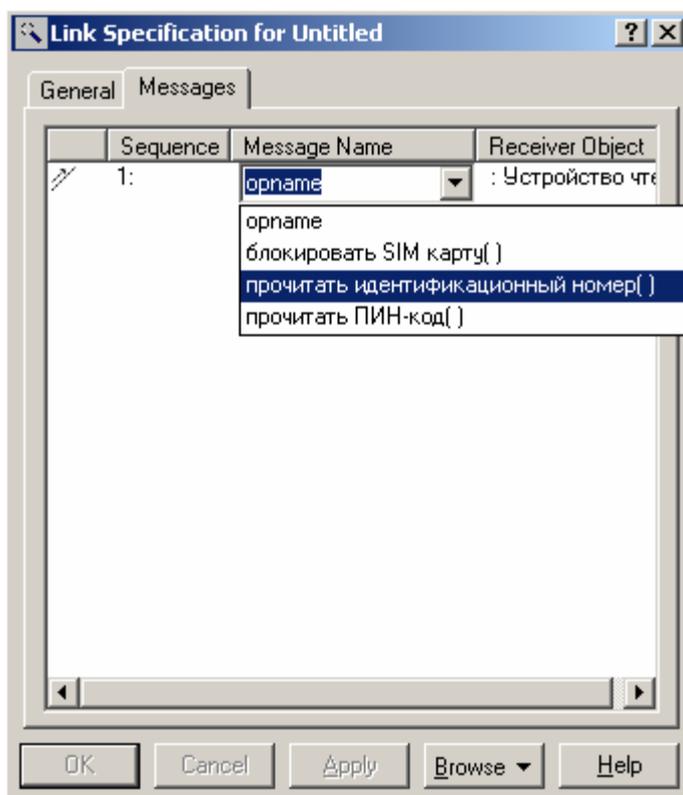


Рис. 3.37. Диалоговое окно добавления сообщения для выбранной связи

Для рассматриваемой модели телефона для первого *сообщения* следует выбрать операцию *прочитать идентификационный номер()*. 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. 3.38).

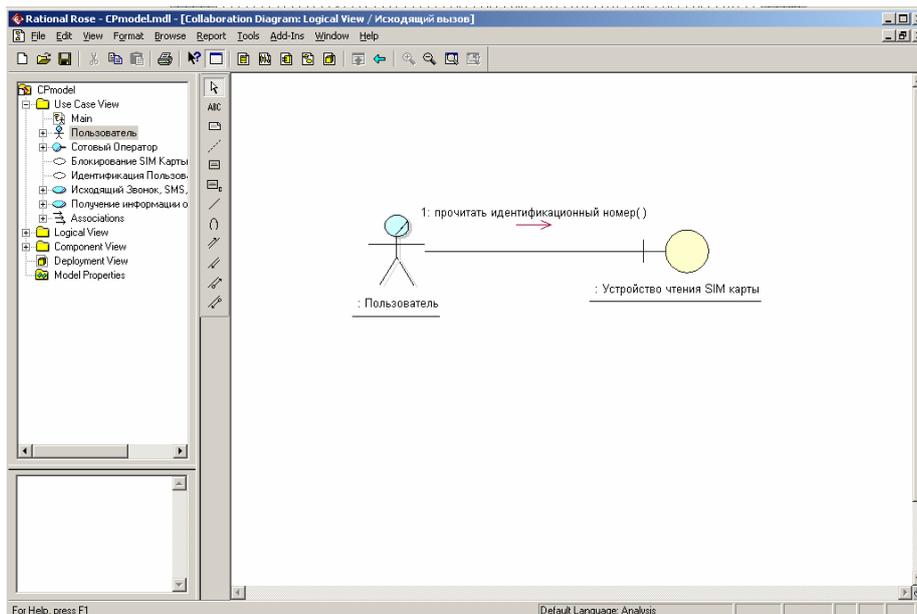


Рис. 3.38. Диаграмма кооперации после добавления связи между объектом класса Пользователь и объектом класса Устройство чтения SIM карты

Кроме имени сообщения можно также задать *стереотип* синхронизации и частоту передачи. Для этой цели следует воспользоваться диалоговым окном спецификации свойств *сообщений* (рис. 3.39), которое можно открыть двойным щелчком на имени *сообщения* в списке рассматриваемой вкладки **Messages** окна спецификации свойств *связи*.

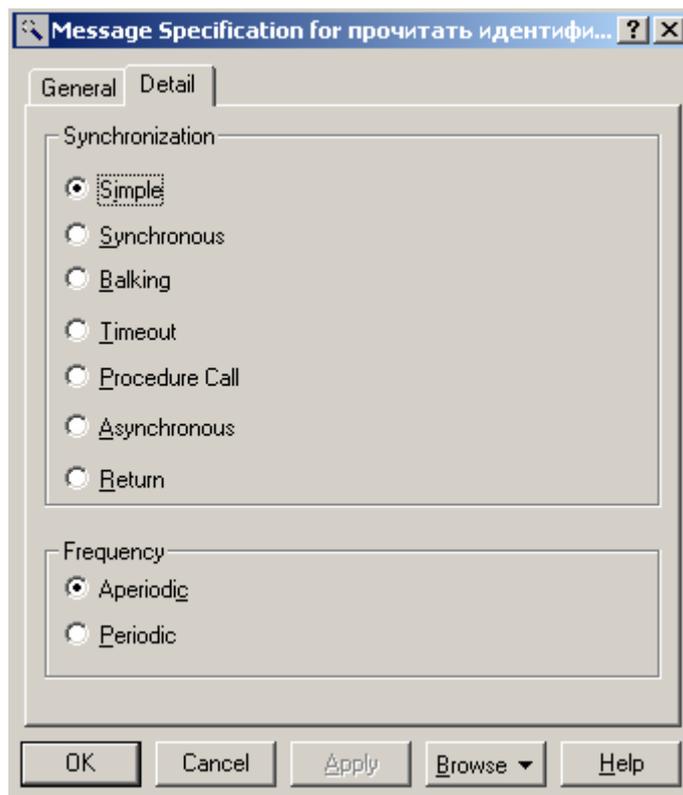


Рис. 3.39. Диалоговое окно спецификации свойств сообщения

Группа свойств **Synchronization** 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. 3.3) [5].

Таблица 3.3. 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		После передачи данного сообщения клиент

(Синхронное)		ожидает ответа от объекта-приемника о результате выполнения соответствующей операции
Balking (С отказом)		После передачи данного <i>сообщения</i> объект-приемник отказывает клиенту в выполнении соответствующей операции, если он занят выполнением других операций
Timeout (С ожиданием)		После передачи данного <i>сообщения</i> объект-приемник может поместить данное <i>сообщение</i> в очередь с ограниченным временем ожидания, если он занят выполнением других операций
Procedure Call (Вызов процедуры)		Клиент посылает данное <i>сообщение</i> объекту-приемнику и, чтобы продолжить свою работу ожидает, пока вся дальнейшая вложенная последовательность <i>сообщений</i> не будет обработана приемником
Asynchronous (Асинхронное)		Клиент посылает данное <i>сообщение</i> и продолжает свою работу, не ожидая подтверждения от объекта-приемника о получении этого <i>сообщения</i> . При этом соответствующая операция может быть как выполнена, так и не выполнена
Return (Возврат)		Данное <i>сообщение</i> посылается клиенту после окончания выполнения вызова процедуры

Группа свойств **Frequency** предназначена для указания периодического характера передачи *сообщения*. При изменении этого *свойства* графическое изображение стрелки соответствующего *сообщения* не изменяется. *Свойство Aperiodic* 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* означает, что *сообщение* регулярно посылается клиентом через определенные промежутки времени.

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

3.6.5. Окончательное построение диаграммы кооперации для модели телефона

Для завершения построения диаграммы кооперации рассматриваемого примера следует описанным выше способом добавить оставшиеся *объекты, связи и сообщения*. С этой целью следует выполнить следующие действия:

1. Добавить *объекты* классов с именами: Контроллер Телефона, Транзакция Телефона, Клавиатура Телефона, Экран Телефона и IКонтроллер Оператора.

2. Добавить *связи*, соединяющие *объекты* классов с именами: Контроллер Телефона с Устройством чтения SIM карты, Контроллер Телефона с Транзакцией Телефона, Контроллер Телефона с Клавиатурой Телефона, Контроллер Телефона с Экраном Телефона и Контроллер Телефона с IИнтерфейсом Банка.

3. Добавить *сообщение*: проверить идентификационный номер (Integer) , направленное от *объекта* класса Контроллер Телефона к *объекту* класса IКонтроллер Оператора.

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

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

6. Добавить *сообщение*: проверить правильность ПИН-кода (), направленное от *объекта* класса Контроллер Телефона к *объекту* класса Устройство чтения SIM карты.

7. Добавить *сообщение*: создать новое соединение (), направленное от *объекта* класса Контроллер Телефона к *объекту* класса Транзакция Телефона.

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

9. Добавить *сообщение*: указать тип соединения (), направленное от *объекта* класса-актера Пользователь к *объекту* класса Клавиатура Телефона.

10. Добавить *сообщение*: показать баланс счета () направленное от *объекта* класса Контроллер Телефона к *объекту* класса Экран Телефона.

11. Добавить *сообщение*: указать тип соединения (), направленное от *объекта* класса-актера Пользователь к *объекту* класса Клавиатура Телефона.

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

13. Добавить *сообщение*: завершить соединение (), направленное от *объекта* класса Контроллер Телефона к *объекту* класса Транзакция Телефона.

Диаграмма кооперации, описывающая реализацию типичного хода событий варианта использования Исходящий вызов для проекта системы управления телефоном, показана на рис. 3.40.

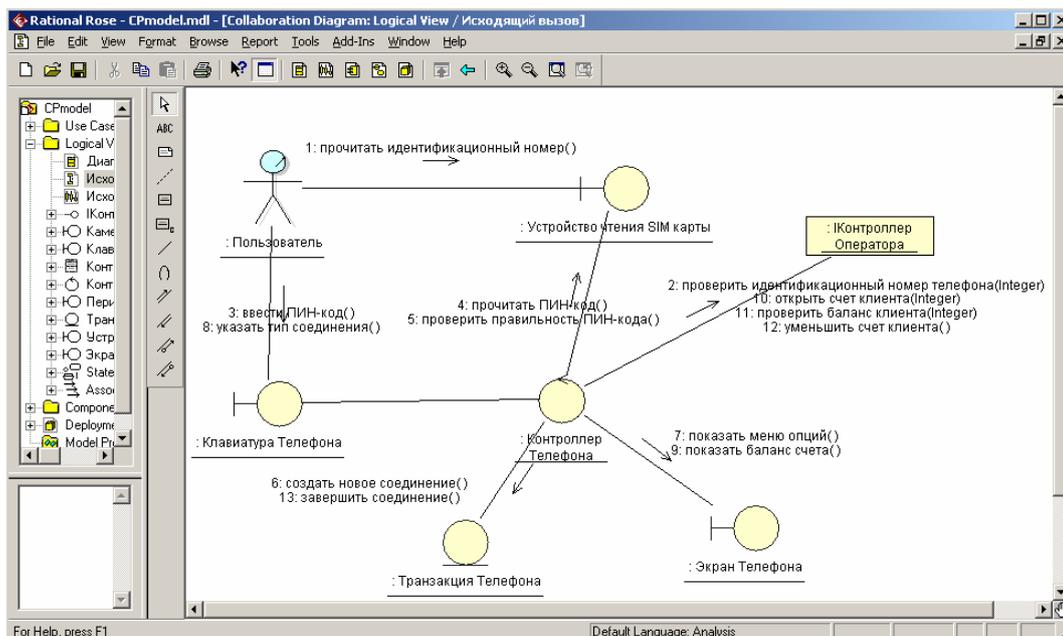


Рис. 3.40. Окончательный вариант диаграммы кооперации, описывающий типичный ход событий варианта использования Исходящий вызов

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

3.7. Разработка диаграммы последовательности и редактирование свойств ее элементов

3.7.1. Особенности разработки диаграммы последовательности

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 [5]:

- Щелкнуть на кнопке с изображением диаграммы взаимодействия на стандартной панели инструментов и выбрать для построения диаграмму последовательности.
- Выполнить операцию главного меню: **Browse/Interaction Diagram** и выбрать для построения новую диаграмму последовательности.
- Выполнить операцию контекстного меню: **New/Sequence Diagram** 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. Назначение отдельных кнопок панели можно узнать из всплывающих подсказок или в литературе [5].

3.7.2. 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 геометриче-

ских размеров и вертикальной пунктирной линией, означающей *линию жизни* этого объекта (рис. 3.41).

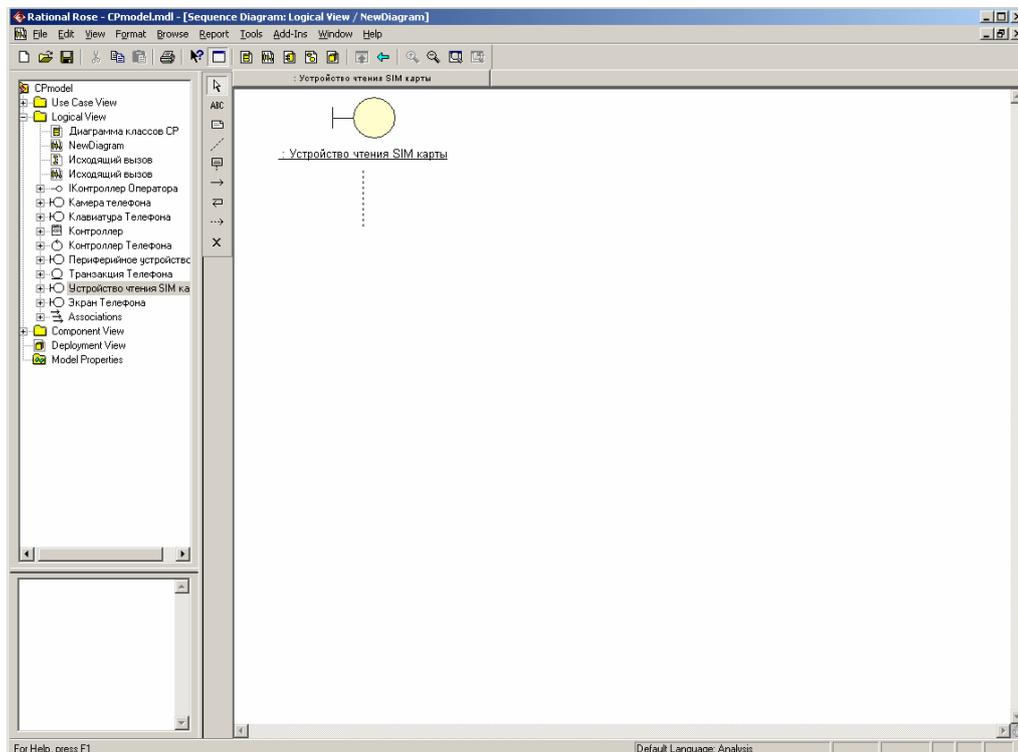


Рис. 3.41. Диаграмма последовательности после добавления анонимного объекта класса Устройство чтения SIM карты

Так же как и для диаграммы кооперации, для диаграммы последовательности каждый добавляемый объект по умолчанию считается анонимным. При необходимости можно задать собственное имя объекта.

3.7.3. 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 Пользователь объекту класса Устройство чтения SIM карты. 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. 3.42).

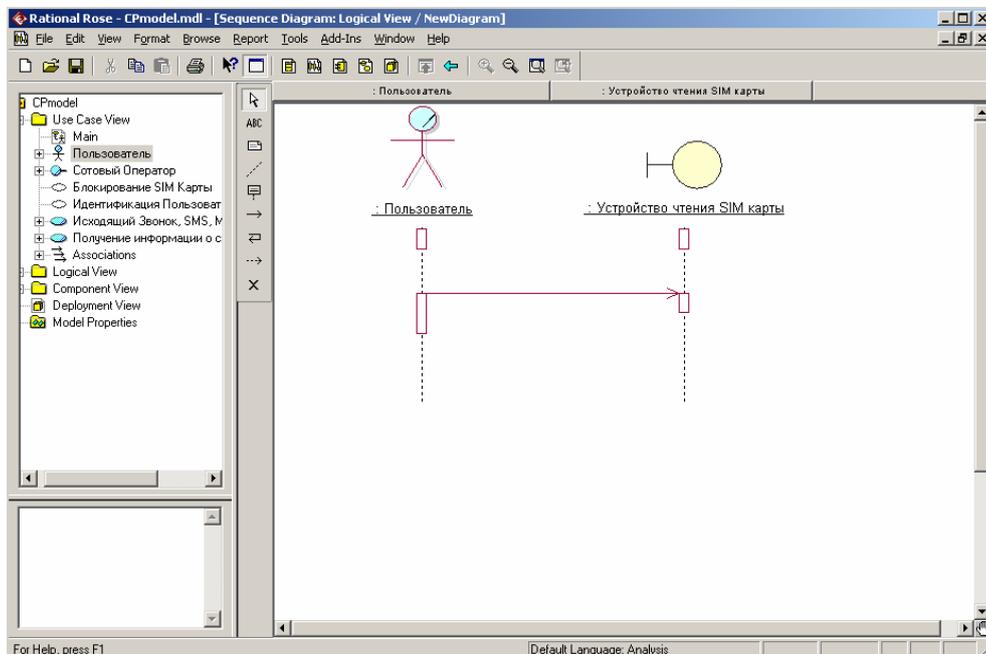


Рис. 3.42. Диаграмма последовательности после добавления сообщения от экземпляра актера Пользователь к объекту класса Устройства чтения SIM карты

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** из выпадающего списка операций соответствующего класса-приемника (рис. 3.43).

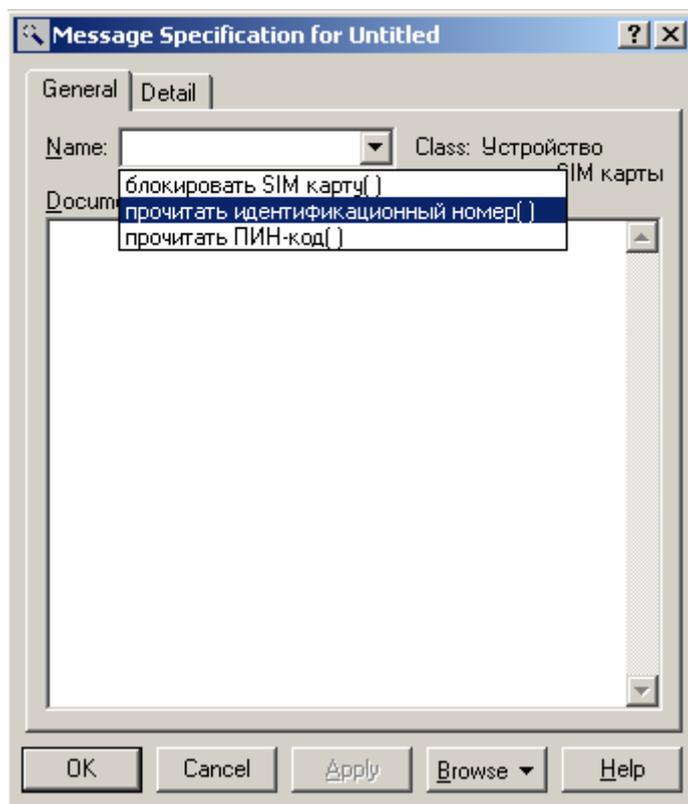


Рис. 3.43. Диалоговое окно спецификации свойств сообщения

Имя сообщения можно выбрать также из контекстного меню сообщения, в котором перечислены все операции класса-приемника данного сообщения (рис. 3.44). 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.

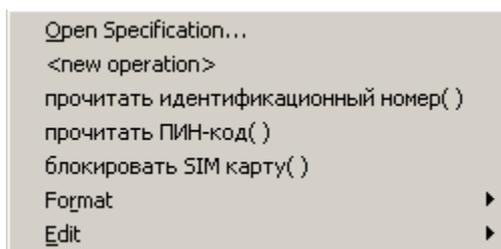


Рис. 3.44. Контекстное меню сообщения на диаграмме последовательности

Для рассматриваемой модели телефона в качестве имени первого *сообщения* следует выбрать операцию прочитать идентификационный номер(). После выбора операции для данного *сообщения* следует нажать кнопку **Apply** или **ОК**, в результате чего имя *сообщения* будет изображено на диаграмме последовательности рядом с линией *сообщения* (рис. 3.45).

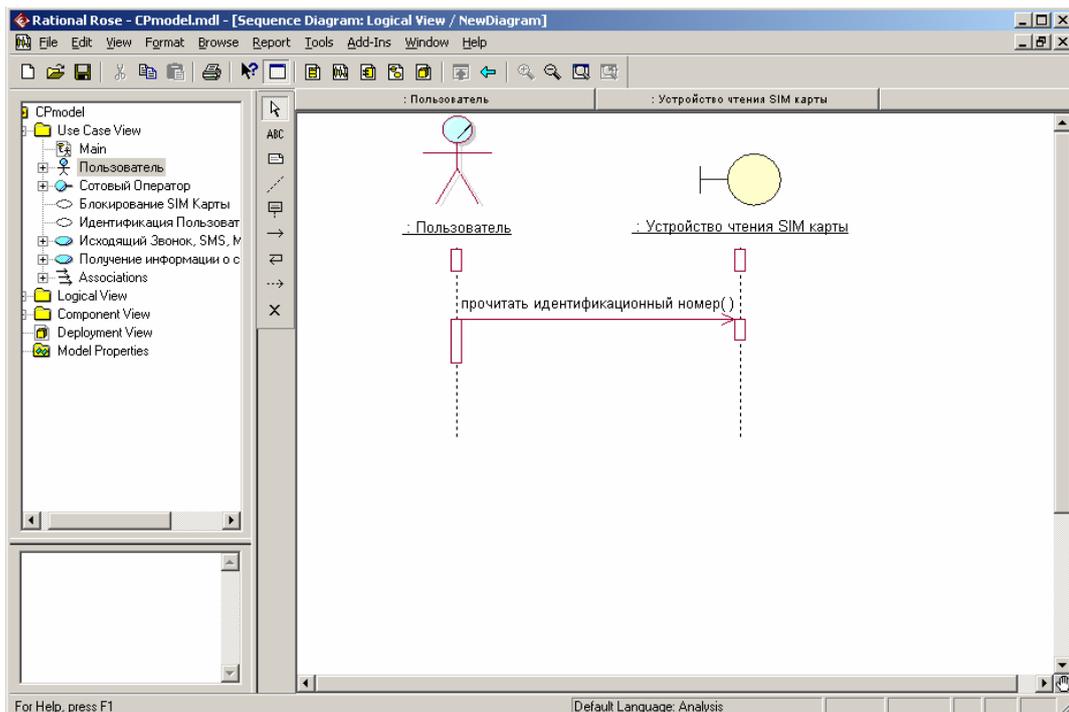


Рис. 3.45. Диаграмма последовательности после добавления сообщения от экземпляра актера Пользователь к объекту класса Устройство чтения SIM карты

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**. 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**, открыть вкладку **Diagram** и выставить отметку выбора строки **Sequence numbering** в группе свойств **Display** (рис. 3.46).

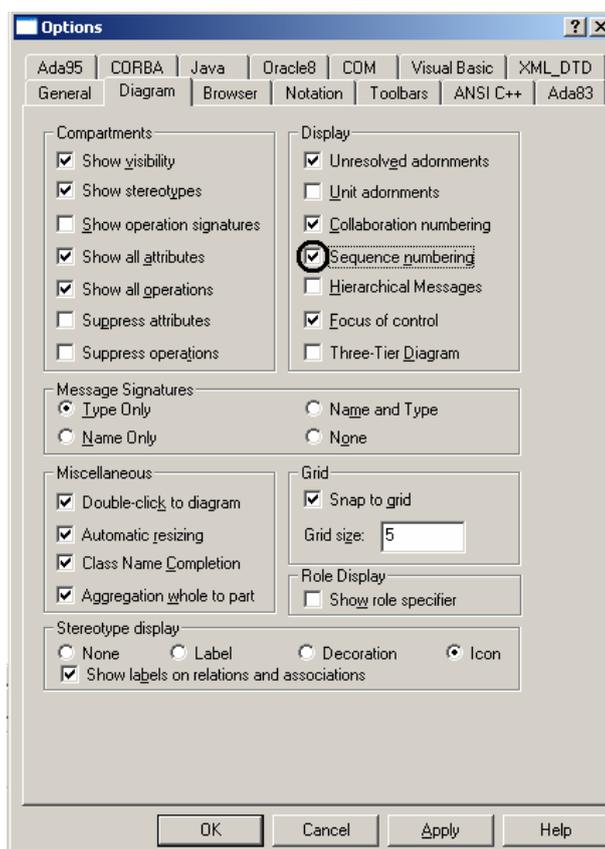


Рис. 3.46. Диалоговое окно спецификации свойств модели

Это же окно спецификации свойств модели можно открыть с помощью операции главного меню **View/Toolbars/Configure**.

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

3.7.4. Окончательное построение диаграммы последовательности модели телефона

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

1. Добавить *объекты* классов с именами: Контроллер Телефона, Транзакция Телефона, Клавиатура Телефона, Экран Телефона и IКонтроллер Оператора.

2. Добавить *сообщение*: проверить идентификационный номер (Integer), направленное от *объекта* класса Контроллер Телефона к *объекту* класса IКонтроллер Оператора.

3. Добавить *сообщение*: ввести ПИН-код (), направленное от *объекта* класса-актера Пользователь к *объекту* класса Клавиатура Телефона.

4. Добавить *сообщение*: прочитать ПИН-код (), направленное от *объекта* класса Контроллер Телефона к *объекту* класса Устройство чтения SIM карты.

5. Добавить *сообщение*: создать новое соединение (), направленное от *объекта* класса Контроллер Телефона к изображению *объекта* класса Транзакция Телефона. При этом изображение *объекта* класса Транзакция Телефона 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 Телефона к *объекту* класса Транзакция Телефона.

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

8. Добавить *сообщение*: указать тип соединения (), направленное от *объекта* класса-актера Пользователь к *объекту* класса Клавиатура Телефона.

9. Добавить *сообщение*: показать баланс счета (), направленное от *объекта* класса Контроллер Телефона к *объекту* класса Экран Телефона.

10. Последовательно добавить 3 *сообщения*: открыть счет клиента (Integer), проверить баланс клиента (Integer) и уменьшить счет клиента (), направленные от *объекта* класса Контроллер Телефона к *объекту* класса IКонтроллер Оператора.

11. Добавить *сообщение*: завершить соединение (), направленное от *объекта* класса Контроллер Телефона к *объекту* класса Транзакция Телефона.

12. После добавления *сообщения* завершить соединение () поместить на *линию жизни объекта* класса Транзакция Телефона символ уничтожения этого *объекта*.

Фрагмент диаграммы последовательности, описывающая реализацию типичного хода событий варианта показан на рис. 3.47.

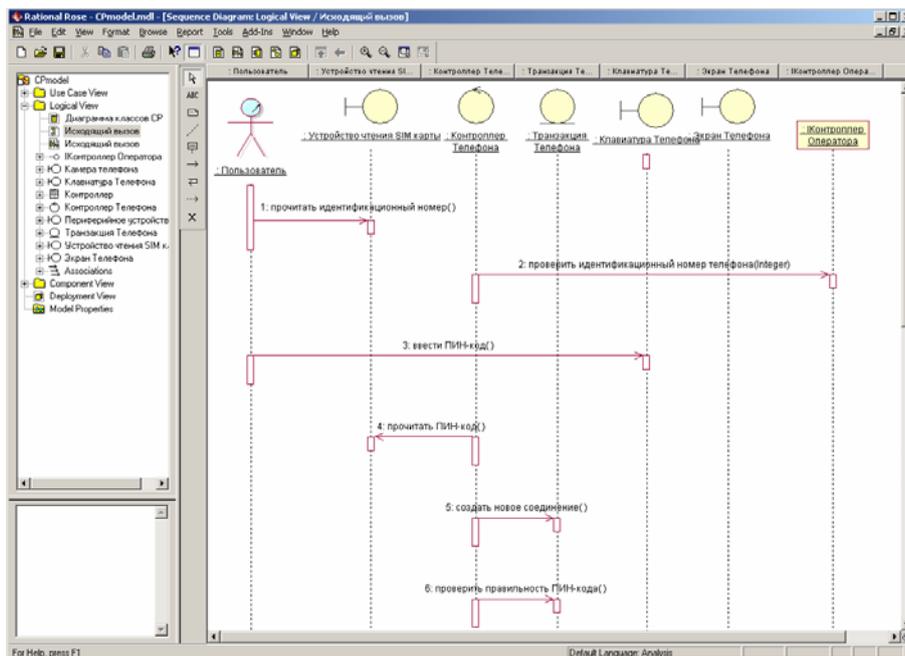


Рис. 3.47. Фрагмент окончательного вида диаграммы последовательности, описывающей типичный ход событий варианта использования *Исходящий вызов*

Если необходимо изменить порядок следования *сообщений*, то из двух диаграмм взаимодействия данное действие удобнее выполнить на диаграмме последовательности, чем на диаграмме кооперации. В этом случае достаточно нажать левую кнопку мыши на стрелке соответствующего *сообщения* и, не отпуская ее, перетащить вертикально вверх или вниз данное *сообщение*. Дополнительно можно добавить потоки данных и определить устойчивость *объектов* на основе активизации соответствующих спецификаций.

3.8. Разработка диаграммы состояний и редактирование свойств ее элементов

3.8.1. Особенности разработки диаграммы состояний

Переходя к рассмотрению диаграммы *состояний*, следует отметить, что в среде IBM Rational Rose 2003 этот тип диаграмм может относиться к отдель-

ному 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 [5]:

- 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**.

- 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**.

- Выполнить операцию главного меню **Browse/State Machine Diagram** после чего следует выбрать представление и тип разрабатываемой диаграммы.

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы *состояний* и специальная панель инструментов, содержащая кнопки с изображением графических элементов модели, необходимых для разработки диаграммы *состояний*. Назначение отдельных кнопок панели можно узнать из всплывающих подсказок или в литературе [5].

Продолжая разработку проекта по моделированию системы управления телефоном, можно приступить к разработке новой диаграммы *состояний*. С этой целью для диаграммы *состояний* модели телефона зададим имя Диа-

грамма *состояний* CP, а в секцию ее документации введем текст «*Диаграмма состояний* описывает конечный автомат телефона».

3.8.2. Добавление состояния на диаграмму состояний и редактирование его свойств

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

Для диаграммы *состояний* модели телефона в качестве имени первого добавленного *состояния* изменим предложенное программой по умолчанию имя NewState на Ожидание SIM карты (рис. 3.48). Задать имя *состояния* можно либо непосредственно при добавлении нового *состояния* на диаграмму *состояний*, либо открыв окно спецификации свойств нового *состояния*.

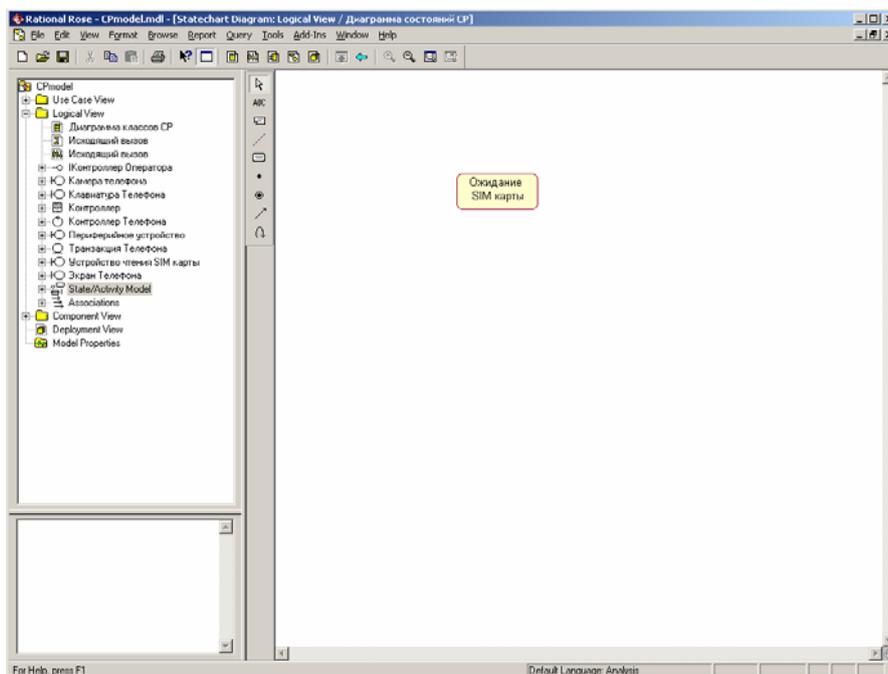


Рис. 3.48. Диаграмма состояний после добавления на нее состояния SIM карты

Для добавленного *состояния* можно открыть диалоговое окно его свойств двойным щелчком левой кнопкой мыши на изображении этого элемента на диаграмме. В этом случае активизируется диалоговое окно со специальными вкладками, в поля которых можно занести всю информацию по данному *состоянию* (рис. 3.49).

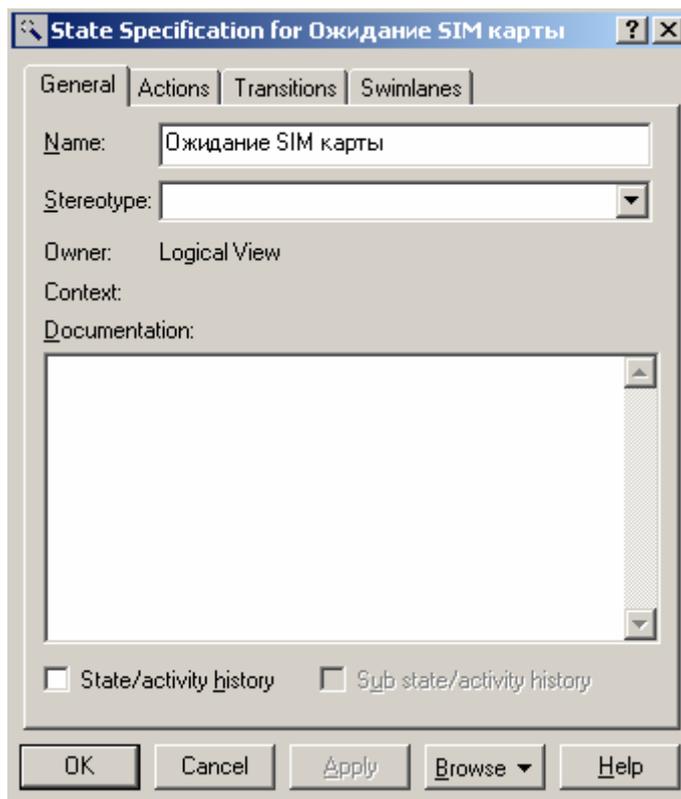


Рис. 3.49. Диалоговое окно спецификации свойств состояния

При необходимости в диалоговом окне спецификации свойств выбранного *состояния* можно задать вложенное *историческое состояние*. Для этого следует выставить отметку у свойства **State/activity history** и нажать кнопку **Apply**. В результате внутри исходного *состояния* появится вложенное *историческое состояние* (рис. 3.50, слева).



Рис. 3.50. Добавление вложенного исторического состояния (слева) и состояния глубокой истории (справа) для состояния *Ожидание SIM карты*

Чтобы обычное *историческое состояние* превратить в *состояние* глубокой истории, следует дополнительно выставить отметку у свойства **Sub state/activity history**, которое становится доступным для редактирования по-

сле выбора первого свойства, и нажать кнопку **Apply**. В результате внутри исходного *состояния* появится вложенное *состояние* глубокой истории (рис. 3.50, справа).

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. 3.51).

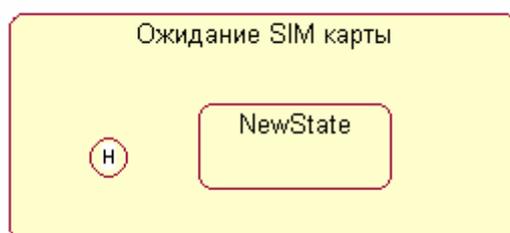


Рис. 3.51. Превращение состояния *Ожидание SIM карты* в *композитное состояние*

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

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** и **Actions**. На вкладке **Transitions** можно определять и редактировать *переходы*, которые входят и выходят из рассматриваемого состояния. Последняя вкладка **Swimlanes** служит для спецификации дорожек, которые, в контексте языка UML, определяются для диаграммы деятельности [5].

3.8.3. 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 телефоном, добавим на диаграмму *состояний* начальное *состояние* (**Start State**) и соединим его *переходом* с *состоянием* Ожидание SIM карты (рис. 3.52).

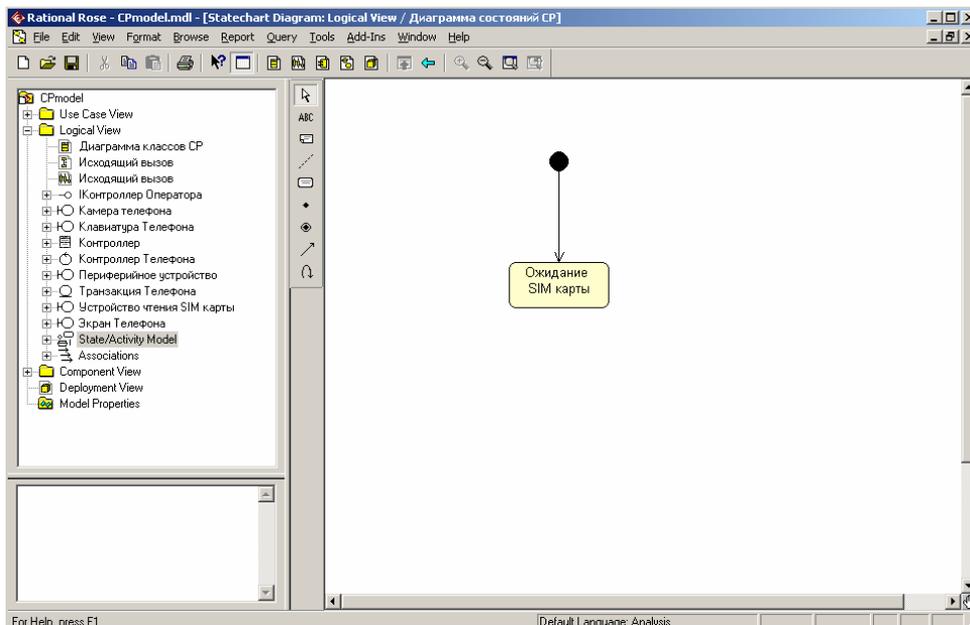


Рис. 3.52. Диаграмма состояний после добавления на нее перехода из начального состояния в состояние Ожидание SIM карты

После добавления *перехода* на диаграмму *состояний* можно открыть диалоговое окно его свойств и специфицировать дополнительные свойства, доступные на соответствующих вкладках (рис. 3.53). Следует обратить внимание на две первые строки вкладки **Detail**, которые представляются наиболее важ-

ными из свойств *перехода*. Первое поле ввода **Guard Condition** служит для задания *сторожевого условия*, которое определяет правило срабатывания соответствующего *перехода*. Во втором поле ввода **Action** можно специфицировать *действие*, которое происходит при срабатывании *перехода* до того, как моделируемая система попадет в целевое *состояние*.

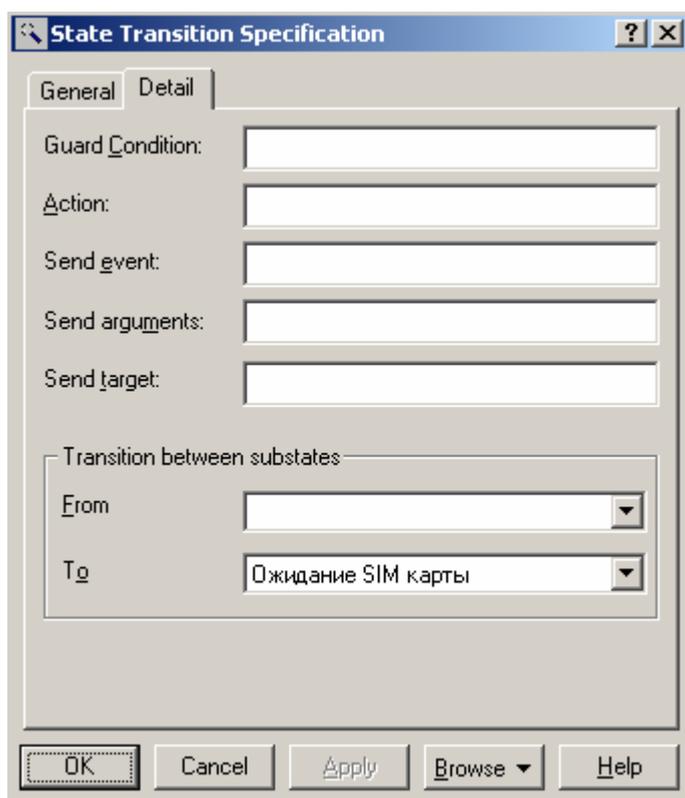


Рис. 3.53. Диалоговое окно спецификации свойств перехода, открытое на вкладке *Detail*

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

3.8.4. Окончательное построение диаграммы состояний модели телефона

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, Режим ожидания, Обработка исходящего вызова, Обработка запроса на получение информации о балансе, Вывод ошибки на экран, Осуществление вызова, Вывод на экран, Завершение соединения и *финальное состояние*.

2. Добавить *переход*: SIM карта вставлена, направленный от *состояния* Ожидание SIM карты 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 со сторожевым условием*: [ПИН-код верный], направленный от *состояния* Проверка ПИН-кода к *состоянию* Режим ожидания

5. Добавить *переход со сторожевым условием*: [ПИН-код неверный], направленный от *состояния* Проверка ПИН-кода к *состоянию* Ожидание ввода ПИН-кода.

6. Добавить *переход*: исходящий вызов, направленный от *состояния* Режим ожидания к *состоянию* Обработка исходящего вызова.

7. Добавить *переход*: баланс счета, направленный от *состояния* Режим ожидания к *состоянию* Обработка запроса на получение информации о балансе.

8. Добавить *переход со сторожевым условием*: [положительный баланс], направленный от *состояния* Обработка исходящего вызова к *состоянию* Осуществление вызова.

9. Добавить *переход со сторожевым условием*: [отрицательный баланс] с *действием на переходе* сообщение, направленный от *состояния* Обработка исходящего вызова к *состоянию* Завершение соединения. Для задания *действия на данном переходе* следует ввести текст сообщение в поле ввода **Action** на вкладке **Detail** окна спецификации свойств данного *перехода* (рис. 3.54).

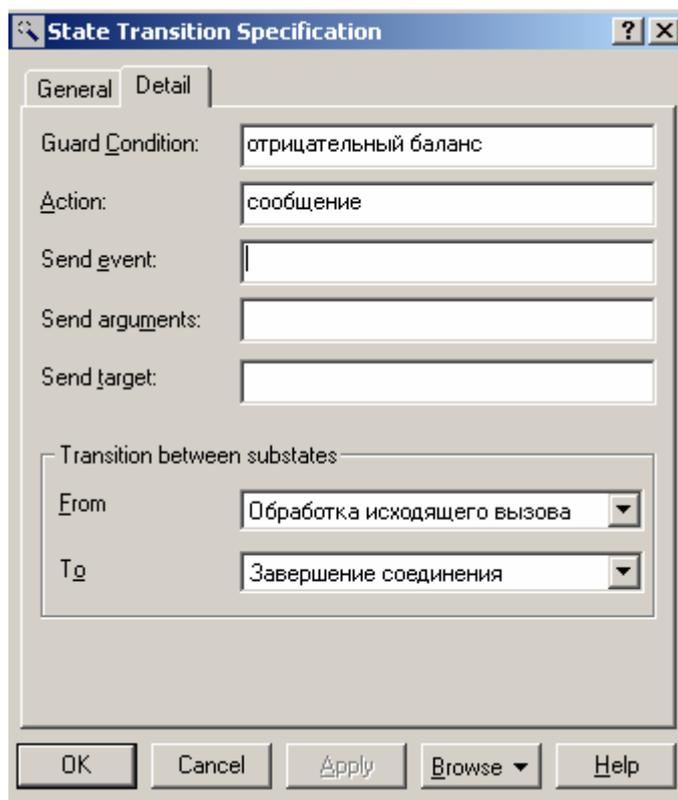


Рис. 3.54. Диалоговое окно спецификации свойств перехода отрицательный баланс при задании действия на переходе

10. Добавить *переход*: информация о вызове, направленный от *состояния* Осуществление вызова к *состоянию* Вывод информации на экран.

11. Добавить *переход*: вызов отклонен, направленный от *состояния* Осуществление вызова к *состоянию* Вывод ошибки на экран.

12. Добавить *переход*: информация сформирована, направленный от *состояния* Обработка запроса на получение информации о балансе к *состоянию* Вывод информации на экран.

13. Добавить *переход*: вызов закончен, направленный от *состояния* Вывод информации на экран к *состоянию* Завершение соединения.

14. Добавить *переход*, направленный от *состояния* Завершение соединения к финальному *состоянию*.

Диаграмма *состояний* для рассматриваемой модели телефона будет иметь следующий вид (рис. 3.55).

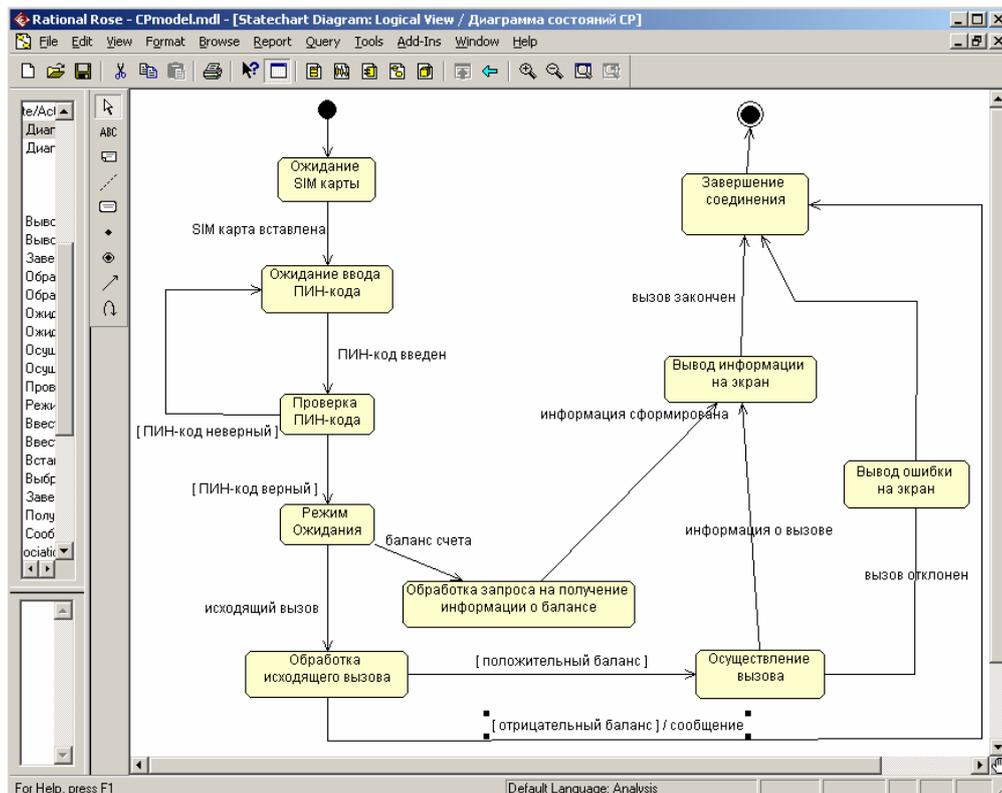


Рис. 3.55. Окончательный вид диаграммы состояний для моделирования поведения телефона

Следует заметить, что в разрабатываемой модели диаграмма *состояний* является единственной и описывает поведение системы управления телефоном.

ном 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.

3.9. Разработка диаграммы деятельности и редактирование свойств ее элементов

3.9.1. Особенности разработки диаграммы деятельности

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 активировать.

Начать построение диаграммы *деятельности* для выбранного элемента модели или моделируемой системы в целом можно одним из следующих способов [5]:

- 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**.

- 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**.

- Выполнить операцию главного меню: **Browse/State Machine Diagram**, после следует чего выбрать представление и тип разрабатываемой диаграммы - диаграмма *деятельности*.

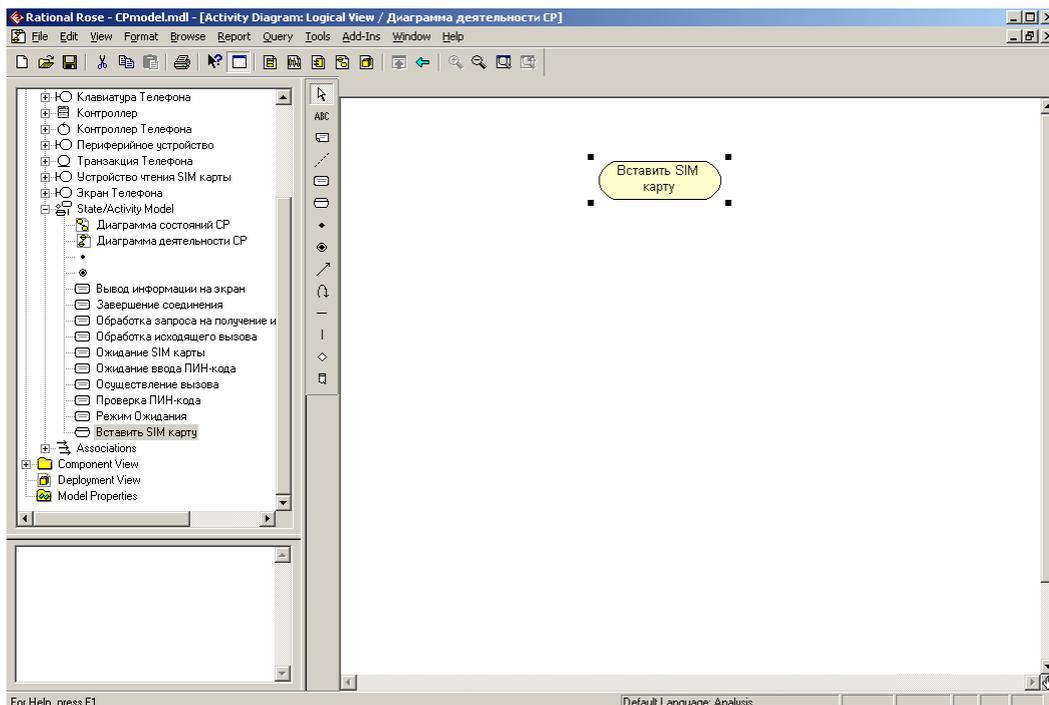
В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы *деятельности* и специальная панель инструментов, содержащая кнопки с изображением графических элементов, необходимых для разработки диаграммы *деятельности*. Назначение отдельных кнопок панели можно узнать из всплывающих подсказок или в литературе [5]. Кнопки с пиктограммами объекта и потока объектов по умолчанию на панели инструментов отсутствуют, при необходимости их можно добавить на специальную панель диаграммы *деятельности* стандартным способом.

Для разрабатываемого проекта системы управления телефоном диаграмма *деятельности* описывает последовательность действий клиента при использовании телефона. 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**. Продолжая разработку проекта по моделированию системы управления телефоном, можно приступить к разработке новой диаграммы *деятельности*. С этой целью для диаграммы *деятельности* модели телефона зададим имя Диаграмма *деятельности* CP, а в секцию ее документации введем текст «Диаграмма *деятельности* описывает последовательность действий клиента при использовании телефона».

3.9.2. Добавление деятельности на диаграмму деятельности и редактирование ее свойств

Для добавления *деятельности* на диаграмму *деятельности* нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы *деятельности* на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение *деятельности* с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить. Добавить *деятельность* на диаграмму можно также с помощью операции главного меню: **Tools/Create/Activity** или с помощью операции контекстного меню: **New/Activity**, предварительно выделив диаграмму *деятельности* в браузере проекта [5].

В результате этих действий на диаграмме появится изображение *деятельности* с именем **NewActivity**, предложенное программой по умолчанию. Начиная построение диаграммы *деятельности* модели телефона, для первой добавленной *деятельности* зададим имя Вставить карточку (рис. 3.56).



*Рис. 3.56. Диаграмма деятельности после добавления на нее деятельности
Вставить SIM*

После добавления *деятельности* на диаграмму *деятельности* можно открыть диалоговое окно спецификации ее свойств и определить дополнительные свойства *деятельности*, доступные на соответствующих вкладках (рис. 3.57).

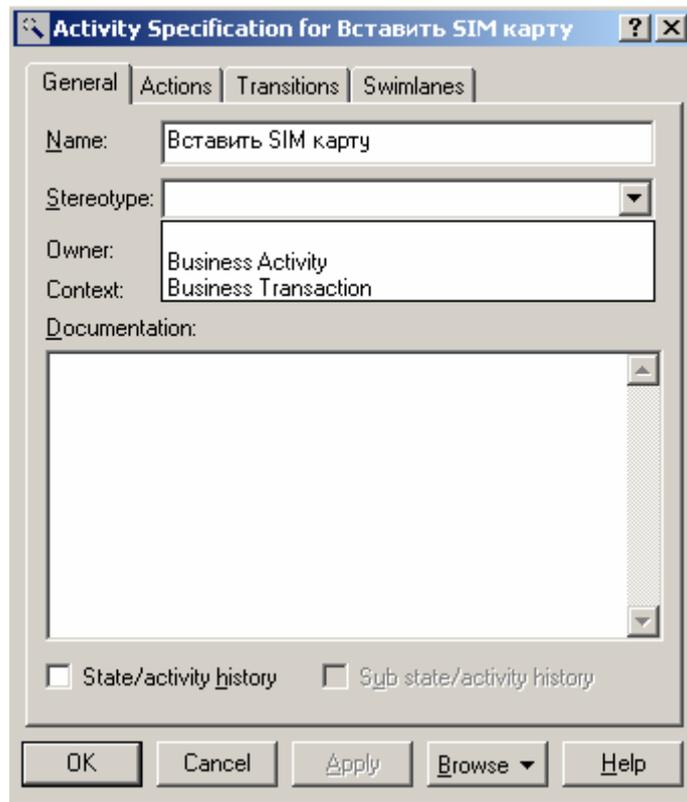


Рис. 3.57. Диалоговое окно спецификации свойств деятельности

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

Хотя программа IBM Rational Rose 2003 позволяет определить свойства *деятельности*, доступные на вкладке **Actions**, 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.

3.9.3. 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. 3.58), которое можно открыть по двойному щелчку левой кнопкой мыши на изображении *перехода*.

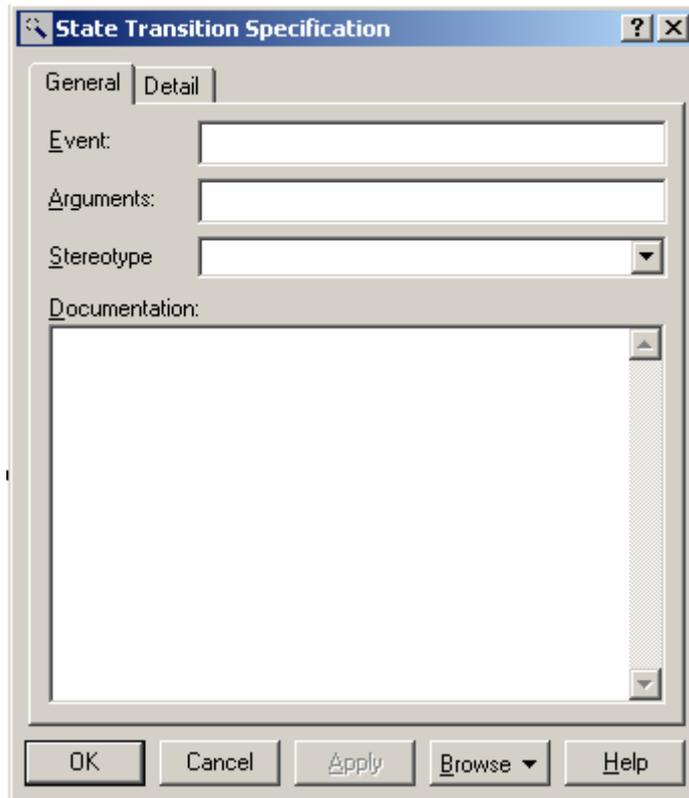


Рис. 3.58. Диалоговое окно спецификации свойств перехода

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** для всех *переходов* должно оставаться пустым. Но все *переходы*, выходящие из *символов ветвления (решения)*, должны иметь *сторожевые условия*, которые специфицируются на вкладке **Detail** диалогового окна спецификации свойств *перехода*.

3.9.4. Окончательное построение диаграммы деятельности модели телефона

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 соединения, Ввести номер, Получить информацию о состоянии счета (точное время, номер телефона и т.д.), Сообщение об отрицательном балансе, Осуществление разговора, Завершить соединение и финальное состояние.

2. Добавить *символы ветвления (решения)*, расположив их между *действиями* с именами: Ввести ПИН-код и Выбрать тип соединения, Выбрать тип соединения и Ввести номер, Ввести номер и Сообщение об отрицательном балансе, Сообщение об отрицательном балансе и Завершить соединение. 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 *действию* Выбрать тип соединения. Для задания *сторожевого условия* данного *перехода* следует ввести текст ПИН-код верный в поле ввода **Guard Condition** на вкладке **Detail** окна спецификации свойств данного *перехода* (рис. 3.59). При этом текст *сторожевого условия* следует вводить без скобок.

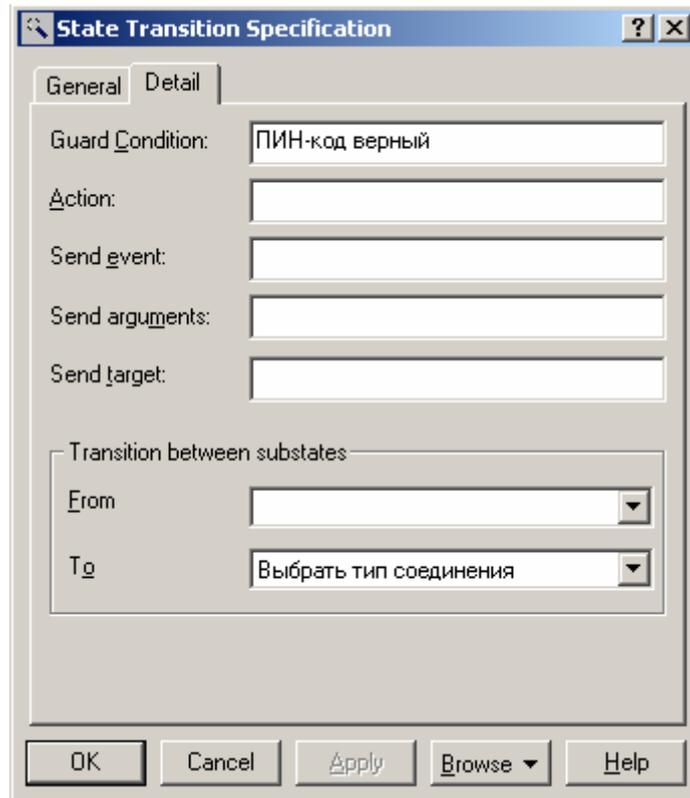


Рис. 3.59. Диалоговое окно спецификации свойств перехода при задании сторожевого условия

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

5. Добавить *переход* со *сторожевым условием*: [ПИН-код неверный], направленный от *символа решения* к символу соединения.

6. Добавить *переход*, направленный от *деятельности* *Выбрать тип соединения* к *символу решения*.

7. Добавить *переход* со *сторожевым условием*: [выбор совершения исходящего соединения], направленный от *символа решения* к *деятельности* *Ввести номер*.

8. Добавить *переход* со *сторожевым условием*: [выбор отправления короткого USSD сообщения], направленный от *символа решения* к *дея-*

тельности Получить информацию о состоянии счета (точное время, номер телефона и т.д.).

9. Добавить *переход*, направленный от *деятельности* Ввести номер к *символу решения*.

10. Добавить *переход* со *сторожевым условием*: [отрицательный баланс], направленный от *символа решения* к *деятельности* Сообщение об отрицательном балансе.

11. Добавить *переход* со *сторожевым условием*: [положительный баланс], направленный от *символа решения* к *деятельности* Осуществление разговора.

12. Добавить *переход*, направленный от *деятельности* Сообщение об отрицательном балансе к *символу соединения*.

13. Добавить *переход*, направленный от *деятельности* Осуществление разговора к *символу соединения*.

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

15. Добавить *переход*, направленный от *символа соединения* к *Завершить соединение*.

16. Добавить *переход*, направленный от *деятельности* Завершить соединение к *финальному состоянию*.

Построенная таким образом диаграмма *деятельности* будет иметь следующий вид (рис. 3.60).

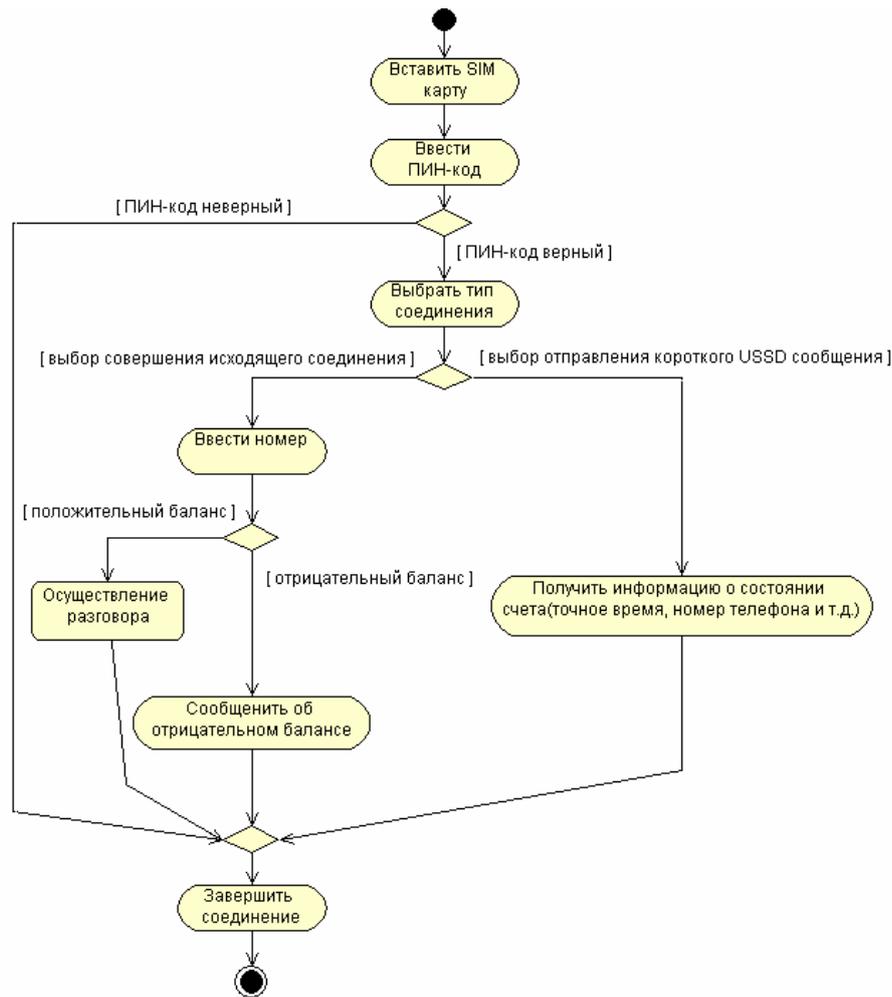


Рис. 3.60. Окончательный вид диаграммы деятельности для модели телефона

Следует заметить, что в разрабатываемой модели диаграмма *деятельности* не описывает ситуацию блокирования SIM карты 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.

3.10. Разработка диаграммы компонентов и редактирование свойств ее элементов

3.10.1. Особенности разработки диаграммы компонентов

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**.

Активизация диаграммы *компонентов* может быть выполнена одним из следующих способов [5]:

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

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*.

Программа IBM Rational Rose 2003 предлагает целый ряд собственных *стереотипов*. Графическое изображение этих *стереотипов* и их краткая характеристика приводятся в литературе [5]. При этом каждому из *компонен-*

тов, как правило, соответствует отдельный файл исходной сборки программного приложения.

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

3.10.2. 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 [5].

В результате этих действий на диаграмме появится изображение *компонента* с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить. Продолжая разработку модели системы управления телефоном, построим для нее диаграмму *компонентов*. С этой целью изменим имя диаграммы, предложенное по умолчанию Main, на Диаграмма *компонентов* CP, а для первого добавленного *компонента* зададим имя MainCP.exe (рис. 3.61).

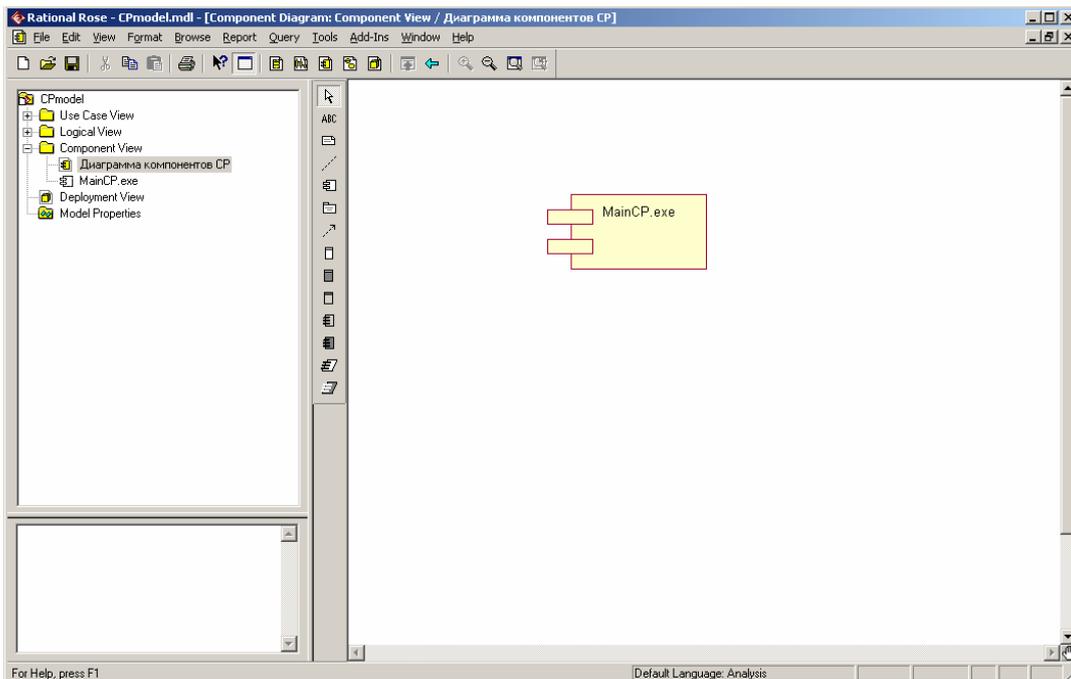


Рис. 3.61. Диаграмма компонентов после добавления компонента *MainCP.exe*

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. 3.62).

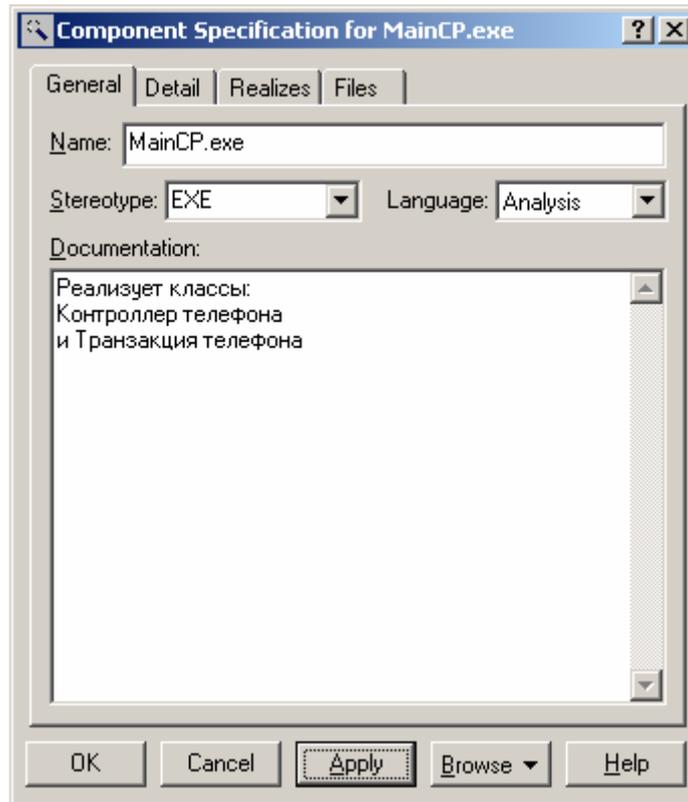


Рис. 3.62. Диалоговое окно спецификации свойств компонента MainCP.exe

В частности, для компонента MainCP.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** содержатся все классы, включая и актеров, которые на данный момент присутствуют в модели (рис. 3.63). Следует заметить, что классы будут показаны в этом окне только при выбранном свойстве **Show all classes**.

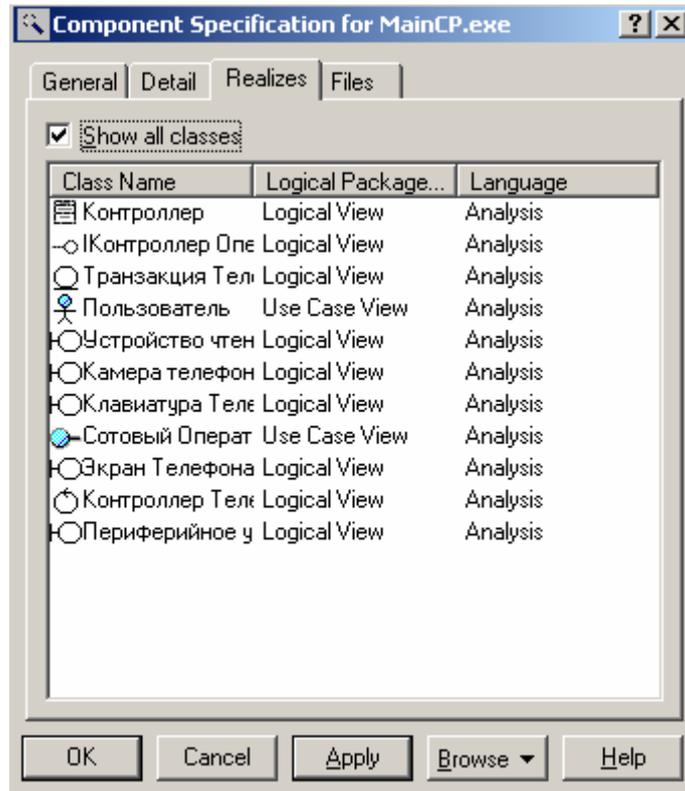


Рис. 3.63. Диалоговое окно спецификации свойств компонента MainCP.exe, открытое на вкладке Realizes

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.

3.10.3. Добавление отношения зависимости и редактирование его свойств

Добавление отношения зависимости на диаграмму *компонентов* аналогично добавлению соответствующего отношения на диаграмму вариантов использования. Продолжая разработку модели телефона, на диаграмму *компонентов* предварительно следует добавить второй *компонент* с именем MainOp, 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* [5].

Применительно к диаграмме *компонентов* модели телефона рассмотренным способом следует добавить отношение зависимости от *компонента* с именем MainCP.exe к *компоненту* с именем MainOp. В дополнение к этому для наглядности можно указать в форме примечаний те классы модели, которые предполагается реализовать в данных *компонентах* (рис. 3.64).

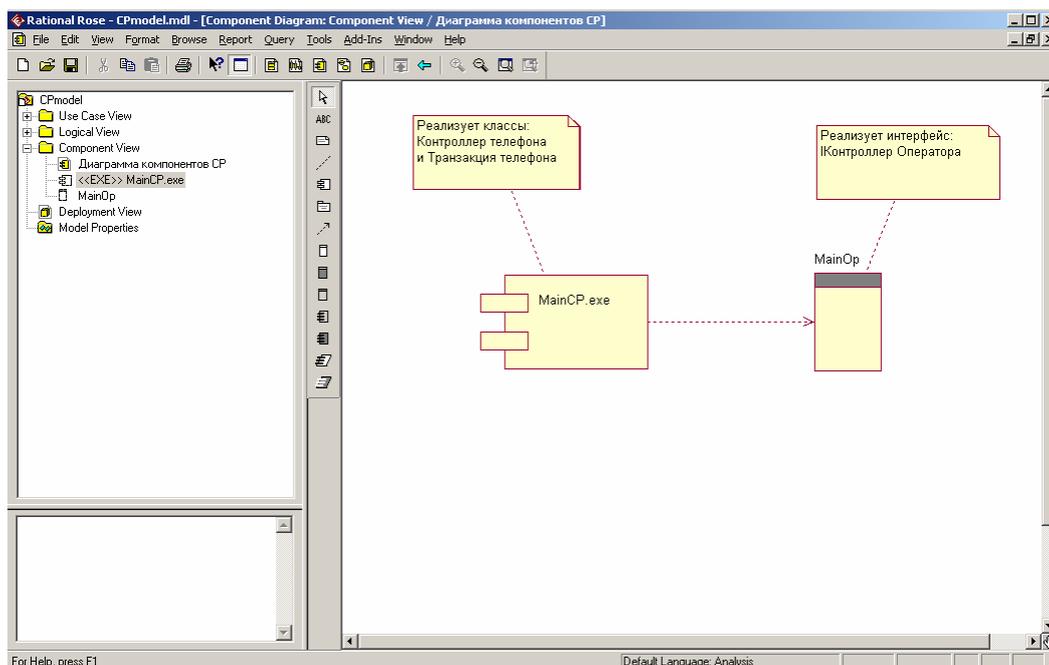


Рис. 3.64. Диаграмма компонентов после добавления отношения зависимости между компонентами *MainCP.exe* и *MainOp*

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.

3.10.4. Окончательное построение диаграммы компонентов модели телефона

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 Телефона, для которого задать стереотип **Task Specification**.

2. Добавить *компоненты* с именами: Устройство чтения SIM карты, Клавиатура Телефона, Камера Телефона, Экран Телефона, Периферийное устройство, для которых задать стереотип **Task Body**.

3. Добавить *зависимость от компонента* с именем MainSR.exe к *компоненту* с именем Устройства Телефона.

4. Добавить *зависимость от компонента* с именем Устройство чтения SIM карты к *компоненту* с именем Устройства Телефона.

5. Добавить *зависимость от компонента* с именем Клавиатура Телефона к *компоненту* с именем Устройства Телефона.

6. Добавить *зависимость от компонента* с именем Камера Телефона к *компоненту* с именем Устройства Телефона.

7. Добавить *зависимость от компонента* с именем Экран Телефона к *компоненту* с именем Устройства Телефона.

8. Добавить *зависимость от компонента* с именем Периферийное устройство к *компоненту* с именем Устройства Телефона.

Построенная таким образом диаграмма *компонентов* будет иметь следующий вид (рис. 3.65).

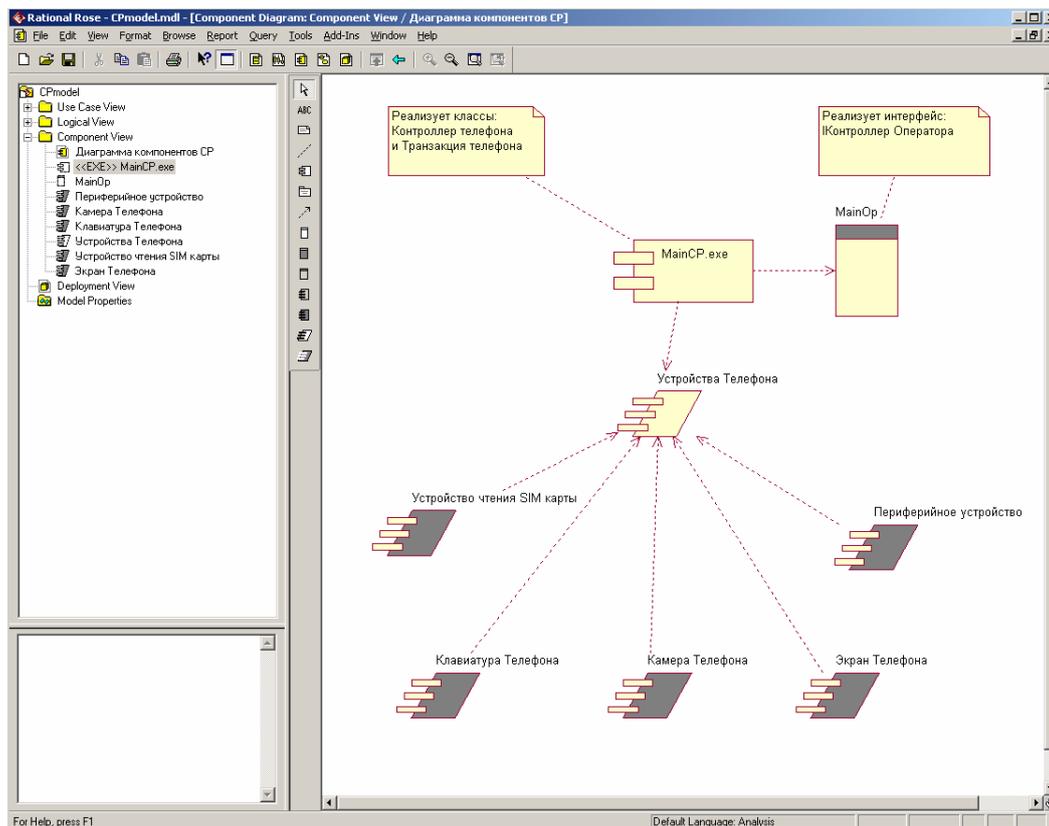


Рис. 3.65. Окончательный вид диаграммы компонентов разрабатываемой модели управления

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.

3.11. Разработка диаграммы развертывания и редактирование свойств ее элементов

3.11.1. Особенности разработки диаграммы развертывания

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 [5]:

- 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**.

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.

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

3.11.2. 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 телефоном, построим для нее диаграмму развертывания. С этой целью в качестве первого *узла* выберем тип *процессор* и зададим ему имя Телефон №1, для которого в форме примечания укажем *помеченное значение*: {адрес абонента = Ленинский проспект, д.5}. Это *значение* служит для спецификации конкретного адреса одного из телефонов системы (рис. 3.66).

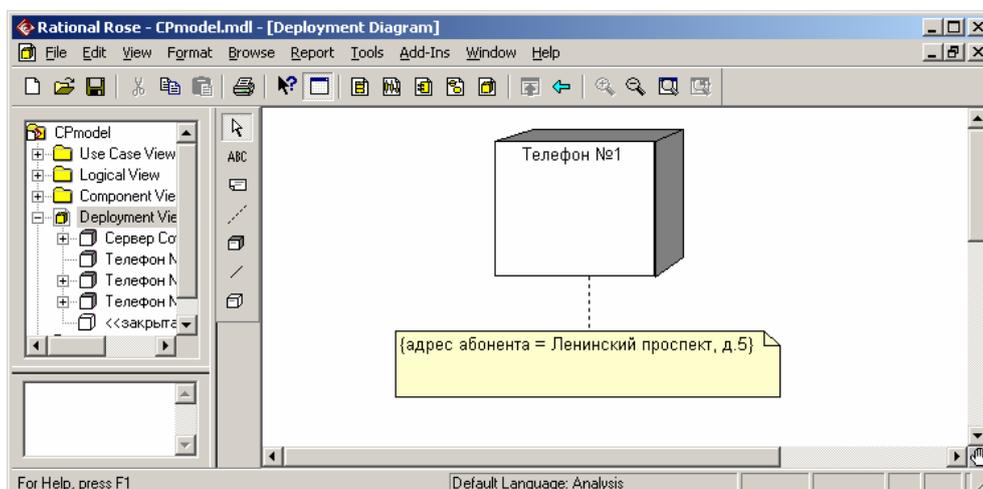


Рис. 3.66. Диаграмма развертывания после добавления узла Телефон № 1

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. 3.67).

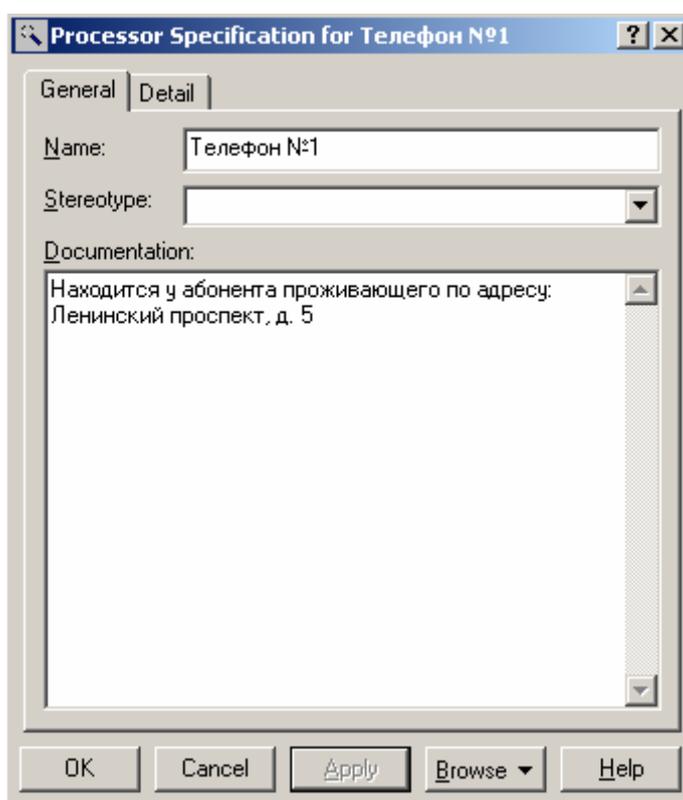


Рис. 3.67. Диалоговое окно спецификации свойств узла Телефон № 1

При этом на вкладке **General** можно только изменить имя *процессора*, ввести текст стереотипа, предложенный самим разработчиком, и текст документации, поясняющий особенности физического размещения данного компонента. На вкладке **Detail** окна спецификации свойств *процессора* можно определить его характеристики, выбрать процессы и вариант *планирования* его работы (рис. 3.68).

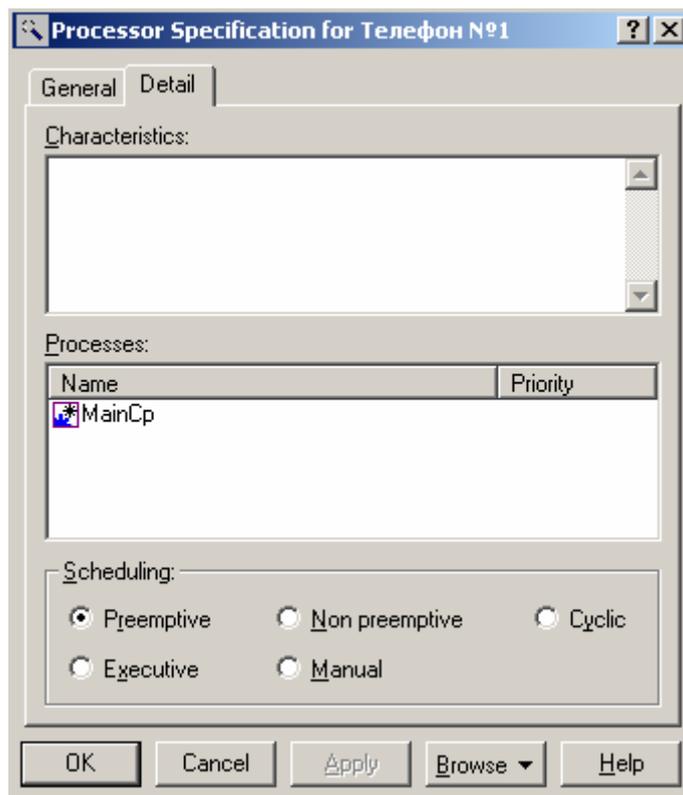


Рис. 3.68. Диалоговое окно спецификации свойств узла Телефон № 1, открытое на вкладке *Detail*

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** можно задать некоторый процесс, который предполагается реализовать на данном *процессоре*. С этой целью необходимо выполнить операцию контекстного меню **Insert** 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** [5]:

- **Preemptive** - определяет процедуру *планирования*, при которой процесс с большим приоритетом будет иметь преимущество при использовании ресурсов *процессора* по сравнению с менее приоритетными процессами.
- **Non preemptive** - 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** - 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** - определяет процедуру *планирования*, для которой существует некоторый алгоритм, предназначенный для управления отдельными процессами.
- **Manual** - 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**. Для отображения информации о процедуре *планирования отдельных процессов* на выбранном *процессоре* следует выполнить операцию контекстного меню **Show Scheduling**.

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

Для *устройства* набор редактируемых свойств меньше, поэтому для него с помощью соответствующего окна спецификации свойств можно определить: имя, стереотип, документацию и характеристику (рис. 3.69). Этот факт согласуется с определением *устройства* как нересурсоемкого узла, на котором отсутствует *процессор*.

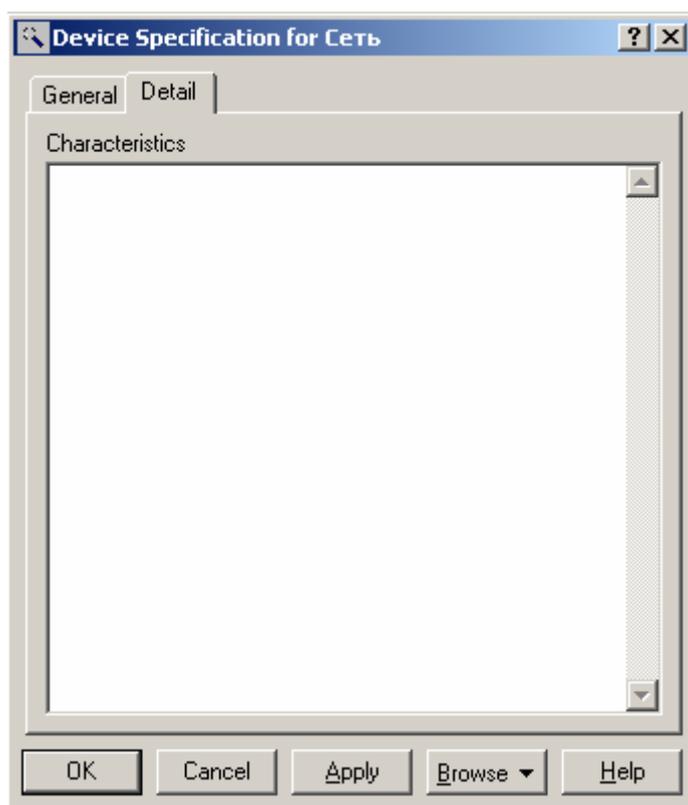


Рис. 3.69. Диалоговое окно спецификации свойств устройства *Сеть*, открытое на вкладке *Detail*

3.11.3. 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 изо-

бражении 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 телефона одним из рассмотренных способов следует добавить *соединение* для узлов с именами Телефон №1 и Сеть (рис. 3.70).

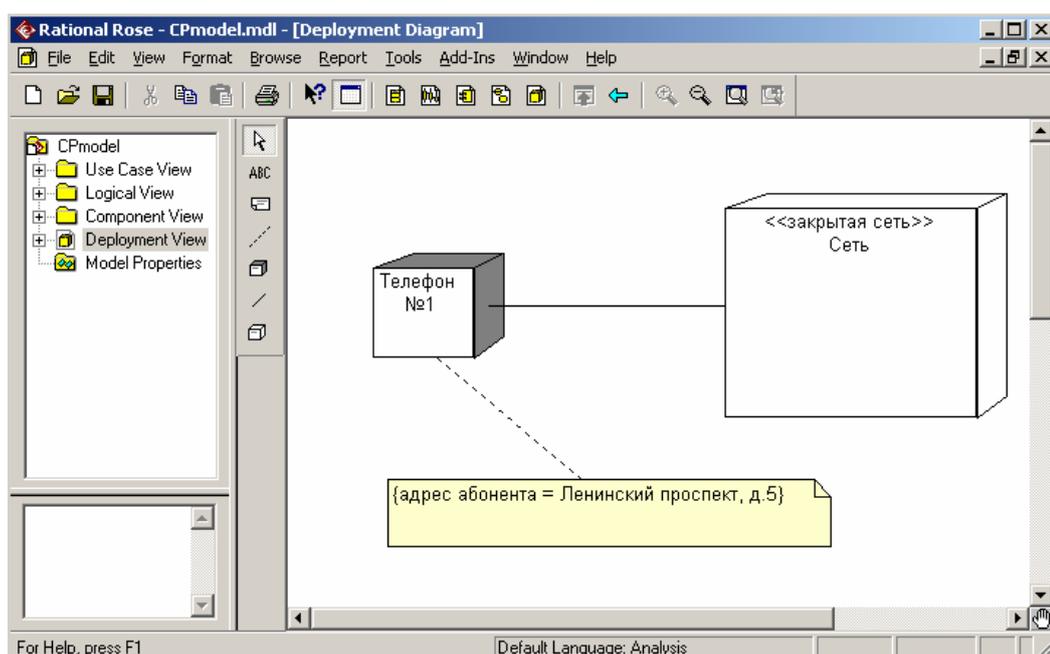


Рис. 3.70. Диаграмма развертывания после добавления соединения между узлами Телефон №1 и Сеть

Для соединения набор редактируемых свойств аналогичен набору свойств устройства, поэтому для него с помощью соответствующего окна спецификации свойств можно определить только имя, стереотип, документацию и характеристику.

3.11.4. Окончательное построение диаграммы развертывания модели телефона

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: Телефон №2, для которого задать *помеченное значение* в форме примечания: {адрес абонента = ул. академика Королева, д.12}, а на вкладке свойств **Detail** определить новый процесс и выбрать для него имя MainCP из вложенного списка.

2. Добавить *процессор* с именем: Телефон №3, для которого задать *помеченное значение* в форме примечания: {адрес абонента = ул. Тверская, д.33}, а на вкладке свойств **Detail** определить новый процесс и выбрать для него имя MainCP из вложенного списка.

3. Добавить *процессор* с именем: Сервер Сотового Оператора, для которого на вкладке свойств **Detail** определить новый процесс с именем MainOp.

4. Добавить *соединение* для узлов с именами Телефон №2 и Сеть.

5. Добавить *соединение* для узлов с именами Телефон №3 и Сеть.

6. Добавить *соединение* для узлов с именами Сервер Сотового Оператора и Сеть.

Построенная таким образом диаграмма развертывания будет иметь следующий вид (рис. 3.71), 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.

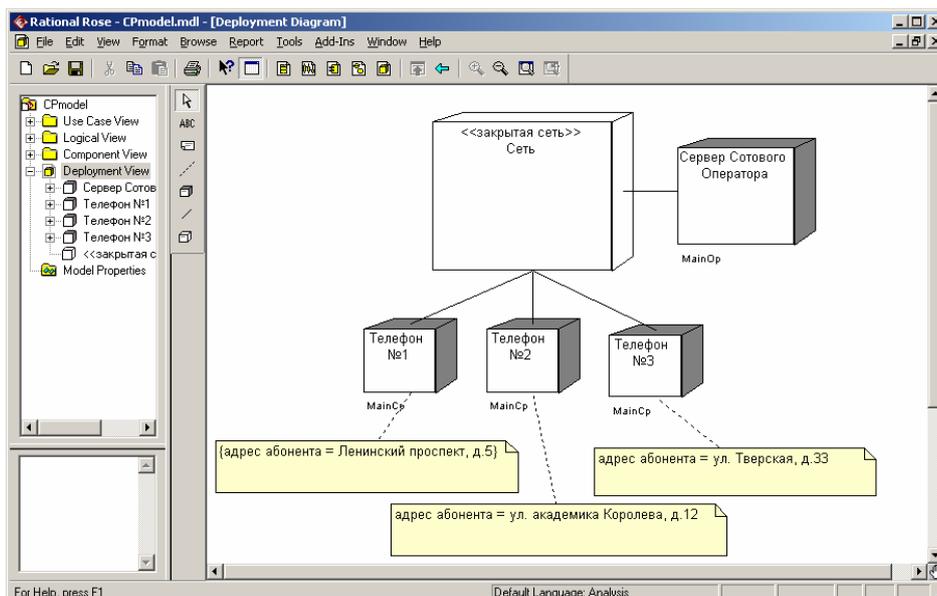


Рис. 3.71. Окончательный вид диаграммы развертывания разрабатываемой модели управления телефоном

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. После построения диаграммы развертывания разработка визуальной модели системы управления телефоном в нотации UML может считаться завершённой.

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**, 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.

3.12. Особенности генерации программного кода в среде IBM Rational Rose 2003

3.12.1. 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 [5]:

- Проверка модели на отсутствие ошибок.
- Создание *компонентов* для реализации *классов*.
- Отображение *классов* на *компоненты*.
- Выбор *языка программирования* для генерации текста программного кода.
- Установка свойств генерации программного кода.
- Выбор *класса, компонента* или пакета.
- 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 целесооб-

разно 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*.

3.12.2. 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 [5].

Для проверки модели следует выполнить операцию главного меню: **Tools/Check Model**. 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. 3.72).



Рис. 3.72. Вид журнала при отсутствии ошибок по результатам проверки модели

3.12.3. 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

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

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*.

3.12.4. 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**, в результате чего будет вызвано диалоговое окно настройки параметров модели. Далее на вкладке **Notation** в строке **Default Language** 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**, 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** на вкладке **General** окна спецификации свойств *компонента*, для чего из вложенного списка следует выбрать язык *ANSI C++*.

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**. 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*. При этом соответствующий *класс* должен быть выделен в браузере проекта.

При генерации программного кода на языке *ANSI C++* для модели телефона значения свойств, предлагаемых средой IBM Rational Rose 2003 по умолчанию, первоначально можно оставить без изменения [5].

3.12.5. 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 телефоном для генерации программного кода на языке *ANSI C++* выберем *компонент* с именем MainCP.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**. В результате этого будет открыто диалоговое окно с предложением выбора *классов* для генерации программного кода на выбранном языке программирования. После выбора соответствующих *классов* и нажатия кнопки ОК программа IBM Rational Rose 2003 выполняет кодогенерацию.

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* или **ANSI C++/Browse Body** для выбранного *класса* в браузере проекта.

Как видно из рассмотрения полученного *заголовочного файла*, в нем содержится объявление в соответствии с правилами синтаксиса языка *ANSI C++* всех операций и атрибутов *класса*. При этом информация о документирова-

нии операций и атрибутов помещается в комментарии перед соответствующими элементами программы [5].

В файле реализации содержится заготовка для реализации всех операций *класса* 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 среде 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 телефоном 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 [5].

Вопросы

- 3.1. Что представляет собой класс в UML?
- 3.2. Что такое «атрибут класса»?
- 3.3. Укажите основные свойства языка моделирования UML.
- 3.4. Укажите возможные типы отношений между классами UML.
- 3.5. Что определяет свойство «видимость атрибута»?
- 3.6. Укажите возможные значения видимости свойства класса.
- 3.7. Определите назначение диаграммы использования.
- 3.8. В каких случаях целесообразно использовать диаграммы деятельности?
- 3.9. Определите назначение диаграмм последовательностей.

Заключение

В заключение можно сказать, что у рассмотренных подходов есть свои сильные и слабые стороны. Так, IDEF1x является общепринятой нотацией для моделирования схем данных, а ERwin предоставляет богатые возможности по ее использованию. В свою очередь UML располагает богатым набором выразительных средств, а Rational Rose позволяет работать с целостной моделью разрабатываемого ПО, что существенно облегчает использование CASE-средства. Визуализированные средствами UML модели ИС позволяют наладить плодотворное взаимодействие между заказчиками, пользователями и командой разработчиков. Они обеспечивают ясность представления выбранных архитектурных решений и позволяют понять разрабатываемую систему во всей ее полноте [4].

Важным критерием для выбора того или иного средства является используемая методология. При использовании структурного анализа лучшим решением будет комплекс из ERwin и VPwin, а при объектно-ориентированном подходе – пакет Rational Rose с Data Modeler. В то же время достаточно эффективной может быть комбинация ERwin для проектирования БД и Rational Rose для работы над клиентским ПО.

При выборе подхода к разработке ИС следует учитывать, что визуальные модели все более широко используются в существующих технологиях управления проектированием систем, сложность, масштабы и функциональность которых постоянно возрастают. Они хорошо приспособлены для решения таких часто возникающих при создании систем задач как: физическое перераспределение вычислений и данных, обеспечение параллелизма вычислений, репликация БД, обеспечение безопасности доступа к ИС, оптимизация балансировки нагрузки ИС, устойчивость к сбоям и т.п [4].

Литература

1. **Вендров А.М.** Case-технологии. Современные методы и средства проектирования информационных систем. - М.: Финансы и статистика, 1998.
2. **Вендров А.М.** Практикум по проектированию программного обеспечения экономических информационных систем: Учебное пособие. - М.: Финансы и статистика, 2006.
3. **Вендров А.М.** Проектирование программного обеспечения экономических информационных систем: Учебник. - 2-е изд., перераб. и доп. - М.: Финансы и статистика, 2005.
4. **Грекул В.И., Денищенко Г.Н., Коровкина Н.Л.** Проектирование информационных систем. Интернет-университет информационных технологий. - ИНТУИТ.ру, 2005. - www.intuit.ru.
5. **Леоненков А.В.** Визуальное моделирование в среде IBM Rational Rose 2003. Интернет-университет информационных технологий. - ИНТУИТ.ру, 2005. - www.intuit.ru.
6. **Леоненков А.В.** Самоучитель UML. - СПб.: БХВ-Петербург, 2001.
7. **Маклаков С.В.** Моделирование бизнес-процессов с APFusion Process Modeler. - М.: Диалог-МИФИ, 2004.
8. **Маклаков С.В.** VPwin и ERwin. CASE-средства разработки информационных систем. - 2-е изд., испр. и дополн. - М.: Диалог-МИФИ, 2001.