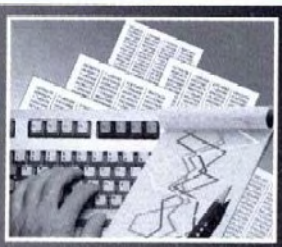
А.М.Вендров

ПРАКТИКУМ

ПО ПРОЕКТИРОВАНИЮ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ



А.М.Вендров

ПРАКТИКУМ

ПО ПРОЕКТИРОВАНИЮ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

Второе издание,
переработанное и дополненное

Допущено
Министерством образования Российской Федерации
в качестве учебного пособия
для студентов высших учебных заведений,
обучающихся по специальностям
"Прикладная информатика в экономике",
"Математическое обеспечение
и администрирование информационных систем"

МОСКВА
"ФИНАНСЫ И СТАТИСТИКА"
2006

УДК [004.415.2:33](076.5)
ББК 65с51я73
В29

РЕЦЕНЗЕНТЫ:

Кафедра проектирования экономических информационных систем
Московского государственного университета экономики, статистики и информатики;
Г. Н. Калянов,
профессор, доктор технических наук,
ведущий научный сотрудник
Института проблем управления
Российской академии наук

Вендров А. М.

В29 Практикум по проектированию программного обеспечения экономических информационных систем: Учеб. пособие. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2006. – 192 с.: ил.

ISBN 5-279-03106-2

Практикум – дополнение к учебнику "Проектирование программного обеспечения экономических информационных систем". Содержит задания и упражнения на построение моделей программных систем на стандартном языке моделирования UML с использованием объектно-ориентированного подхода. В отличие от 1-го издания (2002 г.) пересмотрено описание учебного проекта, заданий и упражнений. Учебный проект выполняется с помощью инструментальных средств IBM Rational Rose и IBM Rational ReguisitePro.

Для студентов высших учебных заведений, обучающихся по специальностям "Прикладная информатика в экономике" и "Прикладная математика и информатика". Может быть полезен разработчикам и пользователям систем программного обеспечения.

В $\frac{2404000000 - 167}{010(01) - 2006}$ 418 – 2005

ISBN 5-279-03106-2

УДК [004.415.2:33](076.5)
ББК 65с51я73

ПРЕДИСЛОВИЕ

Настоящий практикум является дополнением к учебнику "Проектирование программного обеспечения экономических информационных систем" (М.: Финансы и статистика. – 2-е изд., перераб. и доп. – 2005). Практикум содержит задания и упражнения на построение моделей программного обеспечения (ПО) с использованием методики, применяемой в унифицированном процессе создания программного обеспечения (наиболее полно реализованном в технологии IBM Rational Unified Process). Построение моделей и диаграмм стандартного языка моделирования UML выполняется с помощью инструментального средства IBM Rational Rose, а описание требований к программному обеспечению – с помощью IBM Rational RequisitePro. Использование упомянутых инструментальных средств не является строго обязательным: для реализации методики, описанной в практикуме, могут применяться средства других компаний, таких, как Borland или Sybase.

Цель данного пособия – формирование у пользователей навыков самостоятельного практического применения современных методов и средств проектирования программного обеспечения экономических информационных систем, основанных на использовании визуального моделирования.

Практикум состоит из шести глав. В главе 1 приведены основные сведения об инструментальных средствах IBM Rational Rose и Rational RequisitePro. В главах 2–6 последовательно рассматриваются моделирование бизнес-процессов, спецификация требований и анализ требований к программному обеспечению, проектирование системы и генерация кода, используется сквозной пример учебного проекта.

Приложения 1–4 содержат примеры документов учебного проекта, методические указания по курсовому проектированию, возможные темы курсовых проектов по проектированию программного обеспечения и рекомендации по установке инструментальных средств для выполнения упражнений практикума.

Практикум подготовлен в соответствии с Государственным образовательным стандартом по специальности "Прикладная ин-

форматика (по областям)", но может быть использован студентами и преподавателями других специальностей, связанных с проектированием информационных систем и программного обеспечения, в частности "Математическое обеспечение и администрирование информационных систем" и "Прикладная математика и информатика", а также системными аналитиками и разработчиками программного обеспечения.

Материал практикума используется в рамках курса "Объектно-ориентированный анализ и проектирование" на факультете вычислительной математики и кибернетики Московского государственного университета им. М. В. Ломоносова, а также в учебных курсах для специалистов различных организаций.

Автор выражает глубокую благодарность рецензентам, взявшим на себя труд прочитать рукопись и сделавшим ряд конструктивных замечаний, направленных на совершенствование методики подачи материала, а также своим близким за поддержку и терпение, которое они проявили в период написания книги.

Глава 1

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА IBM RATIONAL ROSE И RATIONAL REQUISITEPRO

1.1. ИНСТРУМЕНТАЛЬНОЕ СРЕДСТВО IBM RATIONAL ROSE

1.1.1. ОБЩИЕ СВЕДЕНИЯ

Инструментальное средство IBM Rational Rose (в дальнейшем – Rose) является одним из основных средств комплекса компании IBM Rational Software. Оно предназначено для построения моделей программного обеспечения в процессе его анализа и проектирования, а также генерации кода на различных языках программирования и выпуска проектной документации. Rose используется в процессе объектно-ориентированного анализа и проектирования ПО, описанном в технологии Rational Unified Process (RUP). В основе работы Rose лежат создание элементов моделей с их спецификациями и построение диаграмм языка моделирования UML, определяющих архитектуру системы, ее статические и динамические аспекты. В составе Rose можно выделить шесть основных компонентов: репозиторий, графический интерфейс пользователя, средства просмотра проекта (браузер), средства контроля проекта, средства сбора статистики и генератор документов. К ним добавляются генераторы кода для каждого поддерживаемого языка, состав которых меняется от версии к версии.

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

Средства автоматической генерации кода, используя информацию, содержащуюся в диаграммах классов и компонентов, формируют файлы описаний классов. Создаваемый таким образом скелет программы может быть уточнен путем прямого программирования на соответствующем языке (основные языки, поддерживаемые Rose, – C++ и Java).

В результате разработки проекта с помощью Rose формируются следующие документы:

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

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

В настоящее время компания IBM Rational Software развивает новое поколение инструментальных средств моделирования и разработки ПО (IBM Rational Software Modeler и Software Architect), входящих в комплекс IBM Software Development Platform, основой которого служит интегрированная среда разработки Eclipse. Эти средства развивают возможности Rose в части синхронизации модели и кода (исключающей необходимость прямой генерации кода и обратного преобразования кода в модель).

1.1.2. ЭЛЕМЕНТЫ ИНТЕРФЕЙСА

Интерфейс Rose состоит из пяти основных элементов – это браузер, окно документации, панели инструментов, окно диаграммы и журнал. Их назначение заключается в следующем:

- браузер (browser) используется для быстрой навигации по модели;
- окно документации (documentation window) применяется для работы с текстовым описанием элементов модели;
- панели инструментов (toolbars) используются для быстрого доступа к наиболее распространенным командам;
- окно диаграммы (diagram window) прилагается для просмотра и редактирования одной или нескольких диаграмм UML;

• журнал (log) используется для просмотра ошибок и отчетов о результатах выполнения различных команд.

На рис. 1.1 показаны части интерфейса Rose.

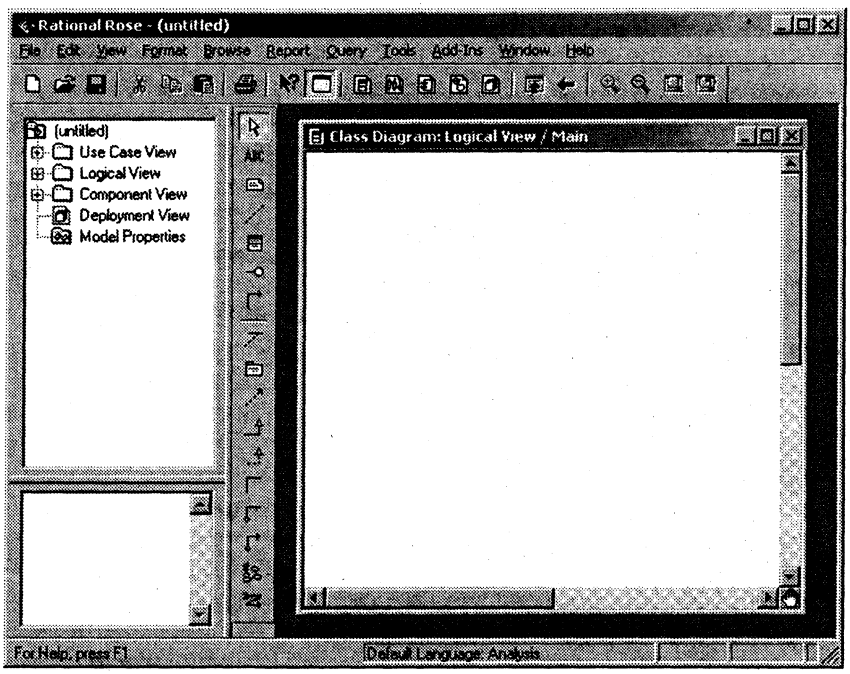


Рис. 1.1. Интерфейс Rose

Браузер – это иерархическая структура, позволяющая осуществлять навигацию по модели. Все, что добавляется к ней – действующие лица, варианты использования, классы, компоненты, – будет показано в окне браузера.

С помощью браузера можно:

• добавлять к модели элементы (действующие лица, варианты использования, классы, компоненты, диаграммы и т.д.);

• просматривать существующие элементы модели и связи между этими элементами;

• перемещать элементы модели;

• переименовывать эти элементы;

• добавлять элементы модели к диаграмме;

• связывать элемент с файлом или адресом URL (Uniform Resource Locator);
• группировать элементы в пакеты;
• работать с детализированной спецификацией элемента;
• открывать диаграмму.

Браузер поддерживает четыре представления (view): представления вариантов использования, компонентов, размещения и логическое представление. Все они и содержащиеся в них элементы модели описаны в подразд. 1.1.3.

Браузер организован в древовидном стиле. Каждый элемент модели может содержать другие элементы, находящиеся ниже него в иерархии. Знак "–" около элемента означает, что его ветвь полностью раскрыта, знак "+" – ветвь свернута.

С помощью ***окна документации*** можно документировать элементы модели Rose, например, сделать краткое описание каждого действующего лица. При документировании класса все, что будет написано в окне документации затем появится как комментарий в сгенерированном коде, что избавит от необходимости вносить эти комментарии вручную. Документация будет выводиться также в отчетах, создаваемых в среде Rose.

Панели инструментов Rose обеспечивают быстрый доступ к наиболее распространенным командам. В этой среде существуют два типа панелей инструментов: стандартная панель и панель диаграммы. Стандартная панель видна всегда, ее кнопки соответствуют командам, которые могут использоваться для работы с любой диаграммой. Панель диаграммы своя для каждого типа диаграмм UML.

Все панели инструментов могут быть изменены и настроены пользователем. Для этого выберите пункт меню Tools > Options, а затем вкладку Toolbars.

Для того чтобы показать или скрыть стандартную панель инструментов (или панель инструментов диаграммы):

1. Выберите пункт Tools > Options.
2. Выберите вкладку Toolbars.
3. Пометьте (или снимите пометку) контрольный переключатель Show Standard ToolBar (или Show Diagram ToolBar), для того чтобы сделать видимой или невидимой стандартную панель инструментов.

Для того чтобы увеличить размер кнопок на панели инструментов:

1. Щелкните правой кнопкой мыши по требуемой панели.
2. Выберите во всплывающем меню пункт Use Large Buttons (Использовать большие кнопки).

Для того чтобы настроить панель инструментов:

1. Щелкните правой кнопкой мыши по требуемой панели.
2. Выберите пункт Customize (настроить).
3. Выберите соответствующую кнопку для того чтобы добавить или удалить кнопки, а затем щелкните мышью по кнопке Add (добавить) или Remove (удалить), как показано на рис. 1.2.

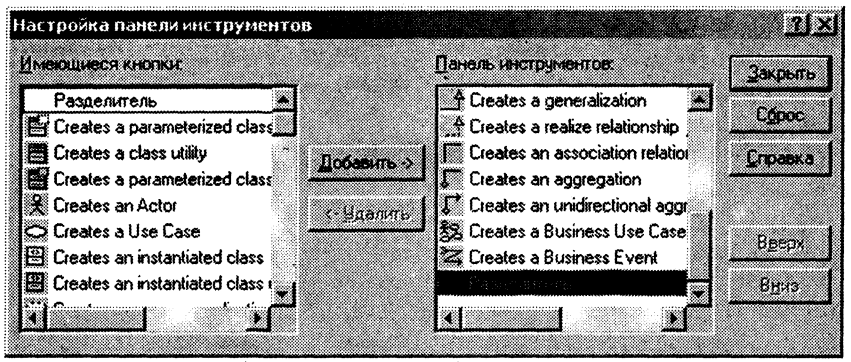


Рис. 1.2. Настройка стандартной панели инструментов

В ***окне диаграммы*** видно, как выглядит одна или несколько диаграмм UML-модели. При внесении в элементы диаграммы изменений Rose автоматически обновляет браузер. Аналогично при внесении изменений в элемент Rose автоматически обновляет соответствующие диаграммы с помощью браузера, что помогает поддерживать модель в непротиворечивом состоянии.

В процессе работы над моделью определенная информация будет направляться в окно ***журнала***, например, сообщения об ошибках, возникающих при генерации кода. Не существует способа закрыть журнал совсем, но его окно может быть минимизировано.

1.1.3. ПРЕДСТАВЛЕНИЯ МОДЕЛИ ROSE

В модели Rose поддерживаются четыре представления (views) – представления вариантов использования, логическое, компонентов и представление размещения.

Представление вариантов использования содержит модели бизнес-процессов и вариантов использования. На рис. 1.3 показано, как выглядит представление вариантов использования в браузере Rose (при использовании здесь и далее шаблона структуры модели, принятой в технологии Rational Unified Process).

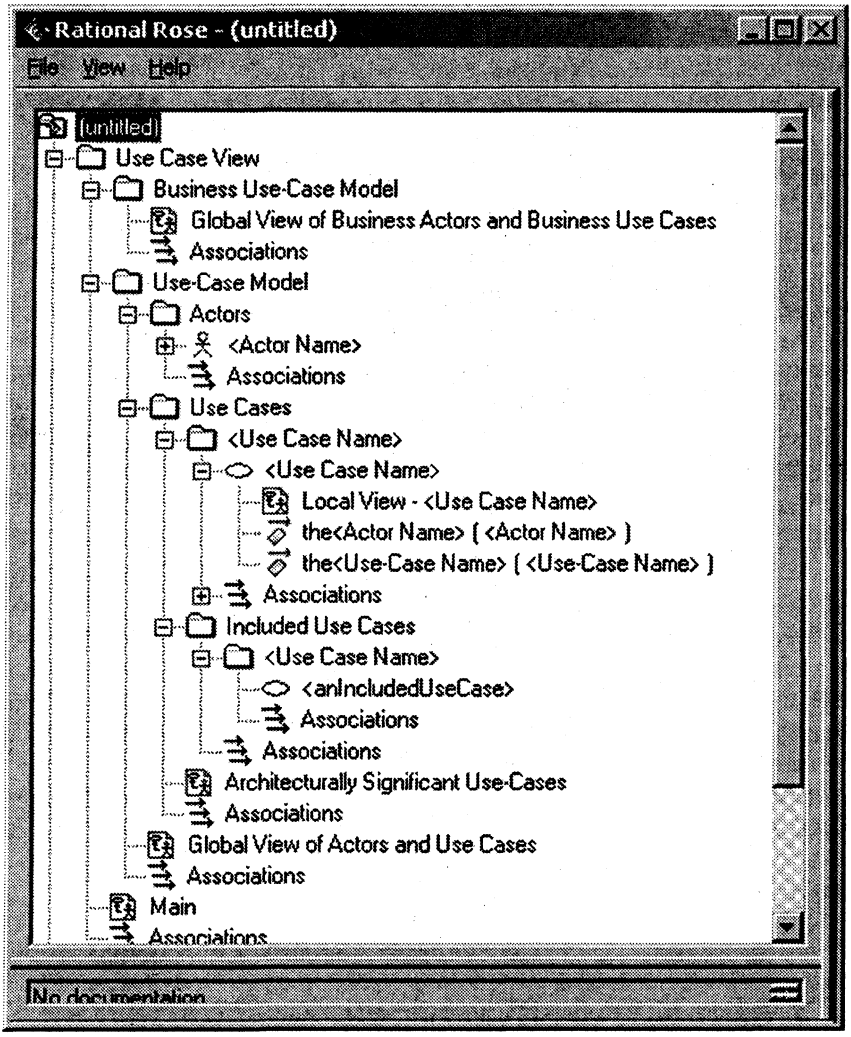


Рис. 1.3. Представление вариантов использования

Представление вариантов использования содержит:

- действующих лиц бизнес-процессов (Business Actor);
- варианты использования с точки зрения бизнес-процессов (Business Use Case);
- действующих лиц и варианты использования системы;
- документацию по вариантам использования, детализирующую их процессы (сценарии или потоки событий). Каждый документ соответствует внешнему файлу, прикрепленному к элементу модели Rose. Вид значка этого документа зависит от приложения, используемого для документирования потока событий (например, Microsoft Word);
- диаграммы вариантов использования, отображающие действующих лиц, варианты использования и взаимодействие между ними. Обычно для описания бизнес-процессов и системы применяется несколько таких диаграмм, каждая из которых показывает подмножество действующих лиц и/или вариантов использования;
- диаграммы деятельности, которые служат для наглядного описания сценариев бизнес-процессов или вариантов использования системы;
- пакеты, содержащие группы вариантов использования и/или действующих лиц.

Логическое представление (рис. 1.4) определяет то, как система будет реализовывать поведение, описанное в вариантах использования. Оно дает подробную картину составных частей системы и описывает их взаимодействие. Логическое представление включает в основном классы и диаграммы классов. С их помощью конструируется детальный проект создаваемой системы.

Логическое представление содержит:

- классы, являющиеся основными элементами архитектуры системы;
- диаграммы классов, используемые для представления классов, их атрибутов, операций и связей. Как правило, для описания системы применяется несколько диаграмм классов, каждая из которых отображает некоторое подмножество всех классов системы;
- диаграммы взаимодействия, применяемые для отображения объектов, участвующих в сценарии варианта использования или в реализации некоторой системной операции;

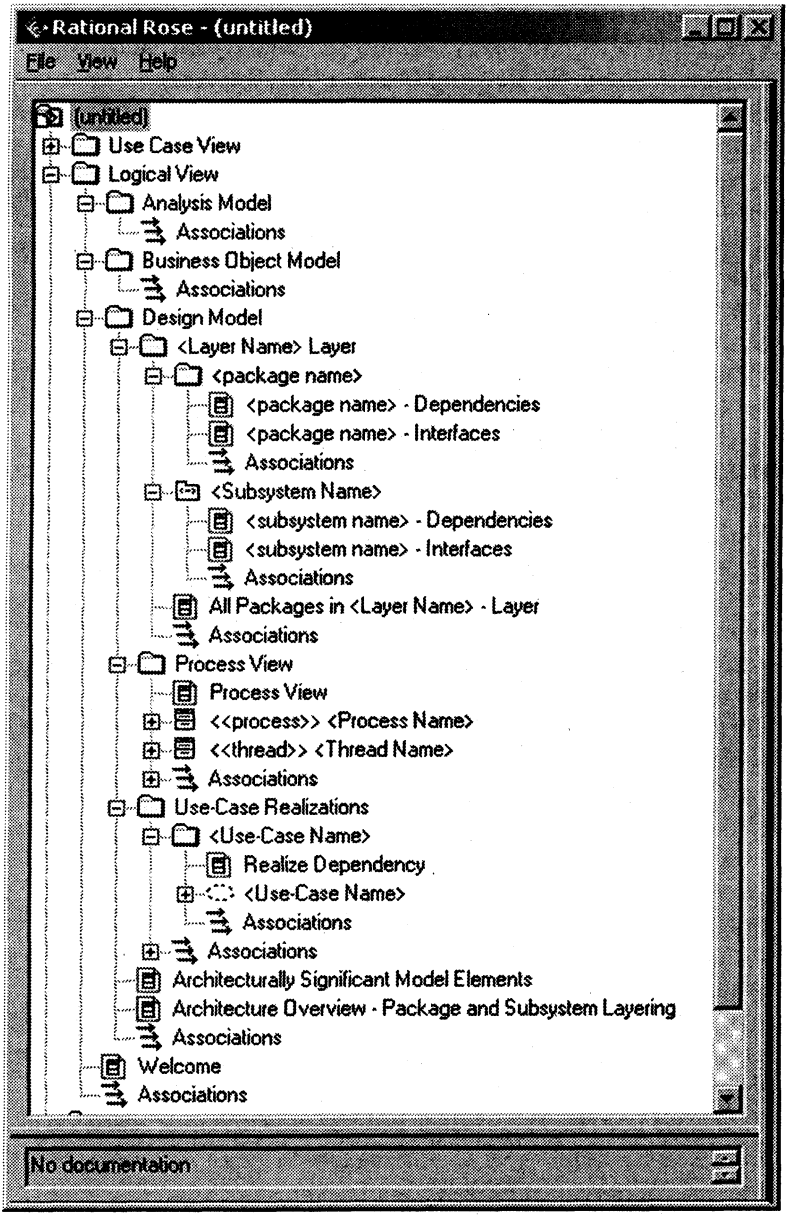


Рис. 1.4. Логическое представление

• диаграммы состояний, описывающие динамику поведения объектов некоторого класса;

• пакеты, содержащие группы взаимосвязанных классов. Типичная система может содержать сотню классов или больше, и объединение их в пакеты снижает сложность модели. Для понимания общей картины системы достаточно взглянуть на ее пакеты.

Наполнение логического представления осуществляется в три этапа.

На первом этапе (при моделировании бизнес-процессов) строится модель бизнес-анализа (пакет Business Object Model), содержащая классы со стереотипами <<business worker>> (исполнитель) и <<business entity>> (сущность). Модель бизнес-анализа может состоять из диаграмм разных типов. В состав модели обязательно должны входить диаграммы классов, содержащие исполнителей и сущности. Кроме диаграмм классов модель бизнес-анализа может включать:

• диаграммы последовательности (и кооперативные диаграммы), описывающие сценарии Business Use Case в виде последовательности обмена сообщениями между объектами-действующими лицами и объектами-исполнителями;

• диаграммы деятельности с потоками объектов и "плавательными дорожками", описывающие взаимосвязи между сценариями одного или различных Business Use Case;

• диаграммы состояний, описывающие поведение объектов отдельных классов-сущностей.

На втором этапе в процессе анализа в пакете Analysis Model определяются классы анализа: граничные классы (boundary), управляющие классы (control) и сущности (entity). Диаграммы классов, реализующих вариант использования, и диаграммы взаимодействия, отражающие взаимодействие объектов в процессе реализации сценариев варианта использования, помещаются в кооперацию с именем данного варианта использования и стереотипом <<use case realization>>. Все кооперации помещаются в пакет с именем Use Case Realizations, входящий в пакет Design Model.

На третьем этапе в процессе проектирования классы анализа преобразуются в проектные классы (design class) и помещаются в пакет Design Model. Данный пакет содержит также описание архитектурных уровней, подсистем и структуры потоков управления.

Представление компонентов содержит:

- компоненты, являющиеся физическими модулями кода;
- диаграммы компонентов, отображающие компоненты и их связи. Связи между компонентами системы отражают зависимости, возникающие при компиляции;
- пакеты, содержащие группы связанных компонентов. Как и в случае классов, одним из критериев объединения компонентов в пакеты является повторное использование.

Представление размещения определяет физическую архитектуру системы, которая может отличаться от ее логической архитектуры.

В представление размещения входят:

- процессоры, в том числе любые компьютеры, способные обрабатывать данные. Любой процесс системы, определенный в структуре потоков управления, выполняется на одном или нескольких процессорах;
- устройства – любая аппаратура, не способная обрабатывать данные, например, терминалы ввода-вывода и принтеры;
- процессы, использующие отведенные для них ресурсы (процессоры);
- диаграмма размещения, на которой показаны процессоры и устройства сети, а также физические соединения между ними. Кроме того, на диаграмме размещения изображают процессы и обозначают, какие процессы выполняются и на каких компьютерах.

1.1.4. РАБОТА В СРЕДЕ ROSE

Создание моделей

Первым шагом в работе с Rose является создание моделей. Их можно строить либо с нуля, либо взяв за основу существующий шаблон. Готовую модель Rose со всеми элементами и диаграммами можно сохранить в одном файле, имеющем расширение .mdl.

Для создания модели:

1. Выберите в меню пункт File > New.
2. Если установлен мастер шаблонов (Framework Wizard), то на экране появится список доступных шаблонов (рис. 1.5). Выбе-

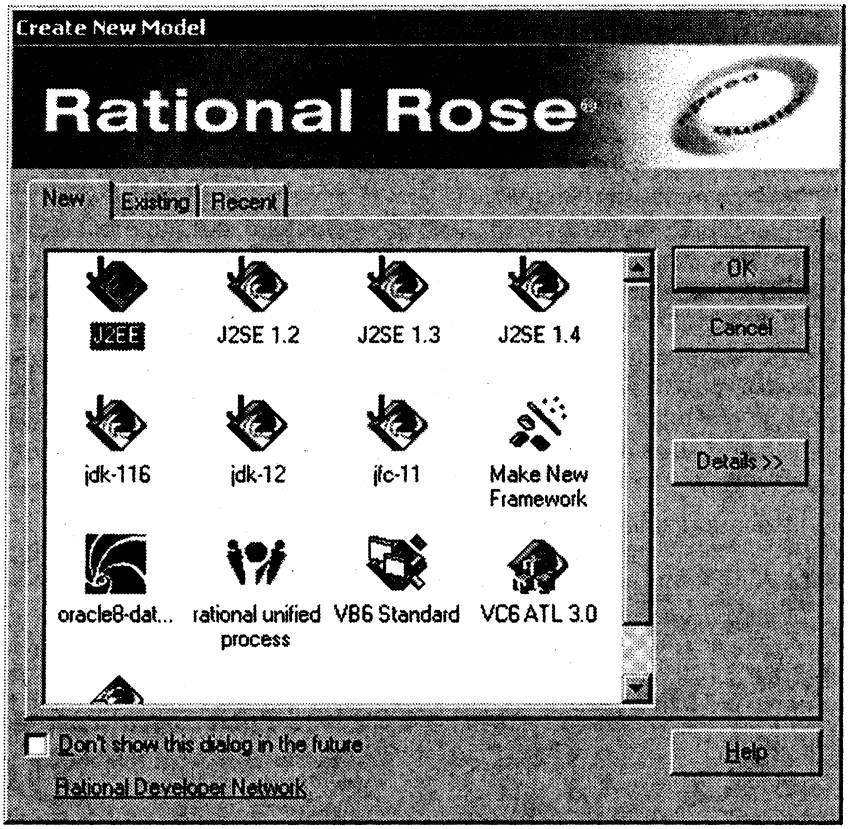


Рис. 1.5. Окно мастера шаблонов

рите шаблон и щелкните по кнопке OK. Если не планируется работать с шаблонами, щелкните по кнопке Cancel.

После выбора шаблона Rose автоматически загружает установленные по умолчанию для этого шаблона пакеты, классы и компоненты. Например, шаблон Rational Unified Process по умолчанию формирует структуру модели (см. рис. 1.3 и 1.4).

Rose позволяет создавать собственный шаблон. В этом случае состав и структура модели подлежат согласованию с общими стандартами проектирования, принятыми в организации, и должны использоваться всеми участниками проекта.

Сохранение моделей

Как и в случае других приложений, рекомендуется периодически сохранять файлы во время работы с ними. Вся модель сохраняется в одном файле. Кроме того, в отдельном файле можно сохранить журнал.

Для сохранения модели выберите в меню пункт File > Save или щелкните мышью по кнопке Save стандартной панели инструментов.

Для сохранения журнала:

1. Выделите окно журнала.
2. Выберите в меню пункт File > Save log As или щелкните мышью по кнопке Save стандартной панели инструментов.
3. Введите название журнала.

Экспорт и импорт моделей

Rose поддерживает экспорт и импорт моделей и их элементов с целью дальнейшего повторного использования.

Для экспорта модели:

1. Выберите в меню пункт File > Export Model.
2. Введите имя экспортируемого файла.

Для экспорта пакета с классами:

1. Выберите пакет на диаграмме классов, который нужно экспортировать.
2. Выберите в меню пункт File > Export <Package>.
3. Введите имя экспортируемого файла.

Для экспорта класса:

1. Выберите класс на диаграмме классов, который нужно экспортировать.
2. Выберите в меню пункт File > Export <Class>.
3. Введите имя экспортируемого файла.

Для импорта модели, пакета или класса:

1. Выберите в меню пункт File > Import Model.
2. Укажите файл, который требуется импортировать. Более подробно с информацией, относящейся к типу импортируемых файлов, можно ознакомиться в справочной информации Rose.

Публикация модели на Web-странице

С помощью Rose можно публиковать модели на Web-страницах, например, в корпоративной сети группы участников проек-

та. Таким образом, все желающие смогут изучить созданные модели, не будучи пользователями Rose и не распечатывая большое количество соответствующей документации.

Для публикации модели в Web:

1. Выберите в меню пункт Tools > Web Publisher.
2. Укажите представления модели и пакеты в окне "Web Publisher" (рис. 1.6).

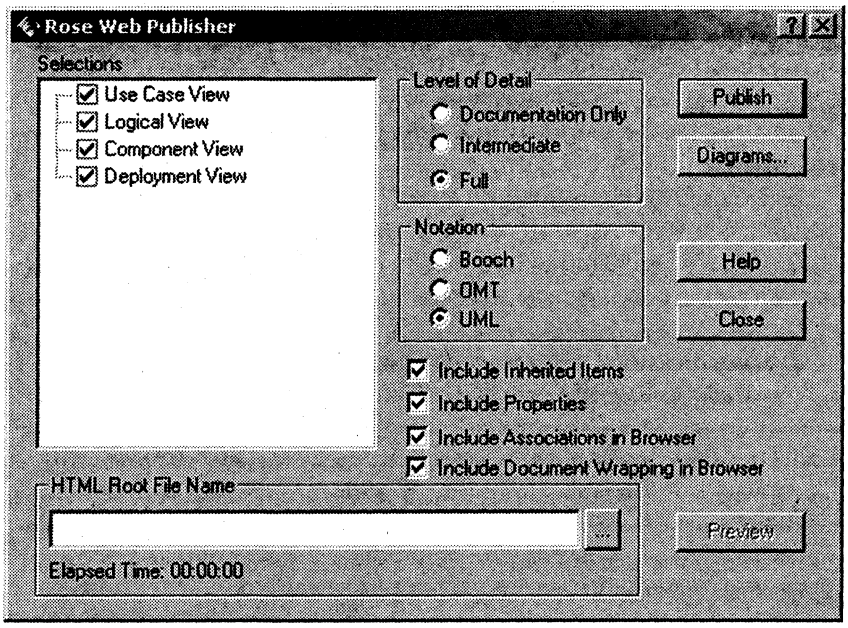


Рис. 1.6. Окно "Web Publisher"

3. Выберите требуемый уровень детализации. Уровень Documentation Only (только документация) соответствует наиболее общей информации, при этом не будут показаны никакие свойства элементов модели. При выборе уровня Intermediate (промежуточный) будут показаны те свойства элементов модели, которые описаны на вкладке "General" их спецификаций. Уровень Full (полный) означает публикацию всех свойств, включая те, что содержатся на вкладке "Detail" спецификации элементов модели.

4. Выберите требуемую нотацию. По умолчанию будет использоваться та нотация, что принята по умолчанию в среде Rose.

5. Укажите, нужно ли публиковать наследуемые элементы (inherited item), свойства и ассоциации (связи между элементами модели). Если установлен соответствующий флажок, то в дереве браузера будут показаны ассоциации.

6. Введите название корневого файла .html, в котором будет опубликована модель.

7. Нажмите на кнопку Diagrams. Если нужно использовать для диаграмм графические файлы, в появившемся окне "Diagram Options" укажите формат, в котором будут опубликованы диаграммы.

8. Закончив, щелкните по кнопке Publish. В результате будут созданы все Web-страницы, требуемые для публикации модели.

9. Щелкните мышью по кнопке Preview, если необходимо ознакомиться с результатом публикации модели.

Добавление в элемент модели Rose ссылок на файлы и адресов URL

Различные данные, находящиеся вне модели, например, документ с требованиями к системе или спецификацию варианта использования, можно связать с определенными элементами модели Rose. Затем открыть этот документ, дважды щелкнув по его значку в окне браузера.

Для подключения к элементу модели файла или адреса URL:

1. Щелкните правой кнопкой по нужному элементу в браузере.

2. Выполните New > File или New > URL.

3. Укажите нужный файл или адрес URL в окне файлов.

4. Щелкните дважды по значку адреса или файла для его открытия.

5. После того как будет подключен адрес или файл, щелкните по значку файла или адреса URL в браузере правой кнопкой и выберите Delete, чтобы удалить файл или адрес URL. С помощью этой операции удаляется связь между моделью Rose и файлом (адресом), но сам файл остается в системе.

Удаление элементов модели

Существуют два способа удалить элемент модели – из одной диаграммы или из всей модели.

Для того чтобы удалить элемент модели из диаграммы:

1. Выделите элемент на диаграмме.
2. Нажмите на клавишу Delete.
3. Обратите внимание на то, что, хотя элемент и удален с диаграммы, он остался в браузере и на других диаграммах системы.

Для того чтобы удалить элемент из модели:

1. Выделите элемент на диаграмме.
2. Выберите в меню пункт Edit > Delete from Model или нажмите сочетание клавиш CTRL + D.

Установка параметров изображения

Глобальные параметры, например такие, как шрифт и цвет применяются для всех элементов модели – классов, вариантов использования, интерфейсов, пакетов и т.д. Значения для этих параметров по умолчанию устанавливаются в меню Tools > Options.

В среде Rose разрешается изменять шрифты элементов на диаграмме. Назначить элементу новые шрифт или размер шрифта можно следующим образом:

1. Выделите нужный элемент или элементы.
2. Выберите в меню пункт Font > Format.
3. Выберите требуемый шрифт, а также его стиль и размер.

Индивидуально изменять можно не только шрифт, но и цвет линии элемента.

Для изменения цвета линии элемента:

1. Выделите нужный элемент или элементы.
2. Выберите в меню пункт Format > Line Color.
3. Выберите требуемый цвет линии.

Для того чтобы изменить цвет заливки элемента:

1. Выделите нужный элемент или элементы.
2. Выберите в меню пункт Format > Fill Color.
3. Выберите требуемый цвет заливки.

В Rose имеется возможность настроить диаграммы классов так, чтобы:

- показывать все атрибуты и операции;
- скрыть операции;

• скрыть атрибуты;
• показывать только некоторые атрибуты или операции;
• показывать операции вместе с их полными сигнатурами или только их имена;
• показывать или не показывать видимость атрибутов и операций;
• показывать или не показывать стереотипы атрибутов и операций.

Значения каждого параметра по умолчанию можно задать с помощью окна, открываемого при выборе пункта меню Tools > Options.

У данного класса на диаграмме можно:
• показать все атрибуты;
• скрыть все атрибуты;
• показать только выбранные вами атрибуты;
• подавить вывод атрибутов.

Подавление вывода атрибутов приведет не только к исчезновению атрибутов с диаграммы, но и к удалению линии, показывающей место расположения атрибутов в классе.

Существуют два способа изменения параметров представления атрибутов на диаграмме. Первый способ заключается в том, чтобы установить нужные значения у каждого класса индивидуально. Второй – изменить значения нужных параметров по умолчанию до начала создания диаграммы классов. Внесенные таким образом изменения повлияют только на вновь создаваемые диаграммы.

Для того чтобы показать все атрибуты класса:
1. Выделите на диаграмме нужный класс.
2. Щелкните по нему правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. Выберите в меню Options > Show All Attributes.

Для того чтобы показать у класса только избранные атрибуты:
1. Выделите на диаграмме нужный класс.
2. Щелкните по нему правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. Выберите в меню Options > Select Compartment Items.
4. Укажите нужные атрибуты в окне Edit Compartment.

Для того чтобы подавить вывод всех атрибутов класса диаграммы:
1. Выделите на диаграмме нужный класс.

2. Щелкните по нему правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.

3. Выберите в меню Options > Suppress Attributes.

Для того чтобы изменить принятый по умолчанию вид атрибута:

1. Выберите в меню модели пункт Tools > Options.

2. Перейдите на вкладку "Diagram".

3. Воспользуйтесь контрольными переключателями Suppress Attributes и Show All Attributes для установки значений параметров отображения атрибутов по умолчанию. Изменение этих значений по умолчанию повлияет только на новые диаграммы, вид существующих диаграмм классов не изменится.

Как и в случае атрибутов, имеется несколько вариантов представления операций на диаграммах, например:

- показать все операции;
- показать только некоторые операции;
- скрыть все операции;
- подавить вывод операций.

Кроме того, можно:

- показать только имя операции. Это означает, что на диаграмме будет представлено только имя операции, но не аргументы или тип возвращаемого значения;
- показать полную сигнатуру операции. Это означает, что на диаграмме будет представлено не только имя операции, но и все ее параметры, типы данных параметров и тип возвращаемого значения операции.

Для того чтобы показать все операции класса:

1. Выделите на диаграмме нужный класс.

2. Щелкните по нему правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.

3. Выберите в меню Options > Show All Operations.

Для того чтобы показать только избранные операции класса:

1. Выделите на диаграмме нужный класс.

2. Щелкните по нему правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.

3. Выберите в меню Options > Select Compartment Items.

4. Укажите нужные операции в окне "Edit Compartment".

Для того чтобы подавить вывод всех операций класса диаграммы:

1. Выделите на диаграмме нужный класс.

2. Щелкните по нему правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.

3. Выберите в меню Options > Suppress Operations.

Для того чтобы показать на диаграмме классов сигнатуру операции:

1. Выделите на диаграмме нужный класс.

2. Щелкните по нему правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.

3. Выберите в меню Options > Show Operation Signature.

Для того чтобы изменить принятый по умолчанию вид операции:

1. Выберите в меню модели пункт Tools > Options.

2. Перейдите на вкладку "Diagram".

3. Воспользуйтесь контрольными переключателями Suppress Operations, Show All Operations и Show Operation Signatures для установки значений параметров отображения операций по умолчанию.

Для того чтобы показать видимость атрибута или операции класса:

1. Выделите на диаграмме нужный класс.

2. Щелкните по нему правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.

3. Выберите в меню Options > Show Visibility.

Для того чтобы изменить принятое по умолчанию значение параметра показа видимости:

1. Выберите в меню модели пункт Tools > Options.

2. Перейдите на вкладку "Diagram".

3. Воспользуйтесь контрольным переключателем Show Visibility для установки параметров отображения видимости по умолчанию.

Для переключения между нотациями видимости Rose и UML:

1. Выберите в меню модели пункт Tools > Options.

2. Перейдите на вкладку "Notation".

3. Воспользуйтесь переключателем Visibility as Icons для переключения между нотациями. Если этот переключатель помечен, будет использоваться нотация Rose. Если нет, то нотация UML. Изменение данного параметра повлияет только на новые диаграммы, существующие диаграммы классов останутся прежними.

1.2. ИНСТРУМЕНТАЛЬНОЕ СРЕДСТВО IBM RATIONAL REQUISITEPRO

1.2.1. ОБЩИЕ СВЕДЕНИЯ

Назначение RequisitePro – управление требованиями (организация требований и отслеживание их изменений в жизненном цикле ПО). Каждое требование имеет определенный тип (используемый для классификации требований) и наименование. Требования содержат текст и обладают следующим стандартным набором атрибутов, который может быть расширен при необходимости:

- приоритет (высокий, средний, низкий);
- статус (предложено, одобрено, утверждено, реализовано, верифицировано);
- стоимость (высокая, средняя, низкая или числовое значение);
- сложность реализации (высокая, средняя, низкая);
- стабильность (высокая, средняя, низкая);
- исполнитель.

Требования могут быть созданы в документе или в представлении (view). Вся информация о требованиях сохраняется в базе данных.

Проект RequisitePro включает базу данных требований и набор связанных с ней документов (например, документы вариантов использования в RequisitePro содержат их обычные текстовые описания). Требования в этих документах связаны с базой данных, в которой хранится дополнительная информация о требованиях – атрибуты, связи трассировки, сведения о версии, история изменений, уровень обеспечения безопасности проекта и др. Из базы данных RequisitePro можно запросить информацию о требовании для проверки его выполнения и оценки влияния изменения. Проект обычно создается менеджером проекта, который определяет его структуру и устанавливает права доступа для участников проекта.

Проектная база данных – база данных требований, управляемая RequisitePro (одна из трех физических баз данных – Microsoft Access, Oracle или Microsoft SQL Server). Каждый проект RequisitePro имеет собственную базу данных.

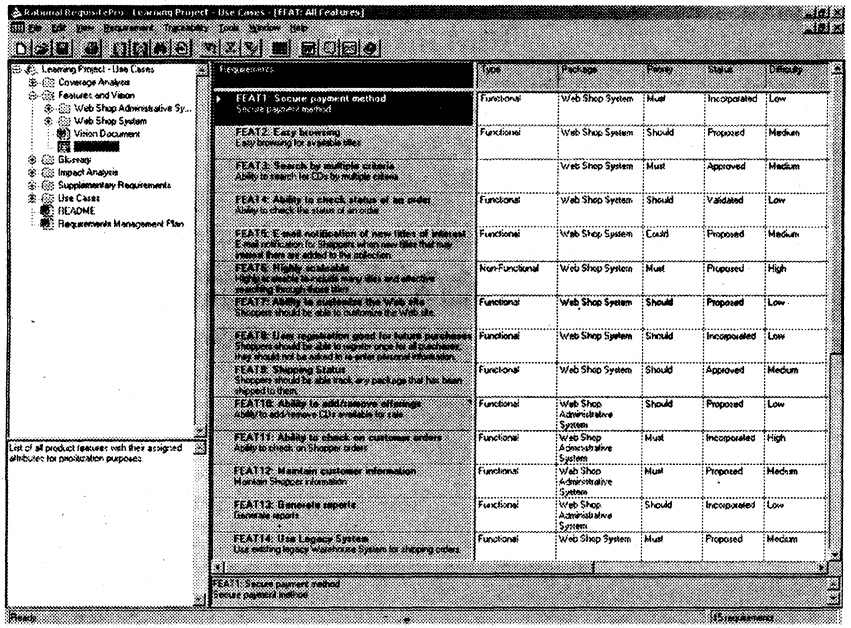


Рис. 1.7. Интерфейс RequisitePro

Основным элементом интерфейса RequisitePro (рис. 1.7) является Проводник (Explorer). В его окне отображаются в иерархической упорядоченности различные рабочие продукты проекта (документы, требования, представления и содержащие их пакеты). Проводник позволяет осуществлять навигацию по проекту.

1.2.2. СОЗДАНИЕ ПРОЕКТА REQUISITEPRO

Для создания проекта RequisitePro:

1. Запустите RequisitePro из меню программ, сразу после запуска может появиться интерфейс системной помощи Let's Go. Закройте его. Затем появится диалоговое окно "Open Project" (рис. 1.8). Оно позволяет выбрать проект для работы с ним (вкладка "Existing") или создать новый проект (вкладка "New").

2. После выбора вкладки "New" появится диалоговое окно "Create Project" (рис. 1.9). Здесь необходимо указать шаблон, на основе которого будет создан новый проект. Шаблоны "Composite

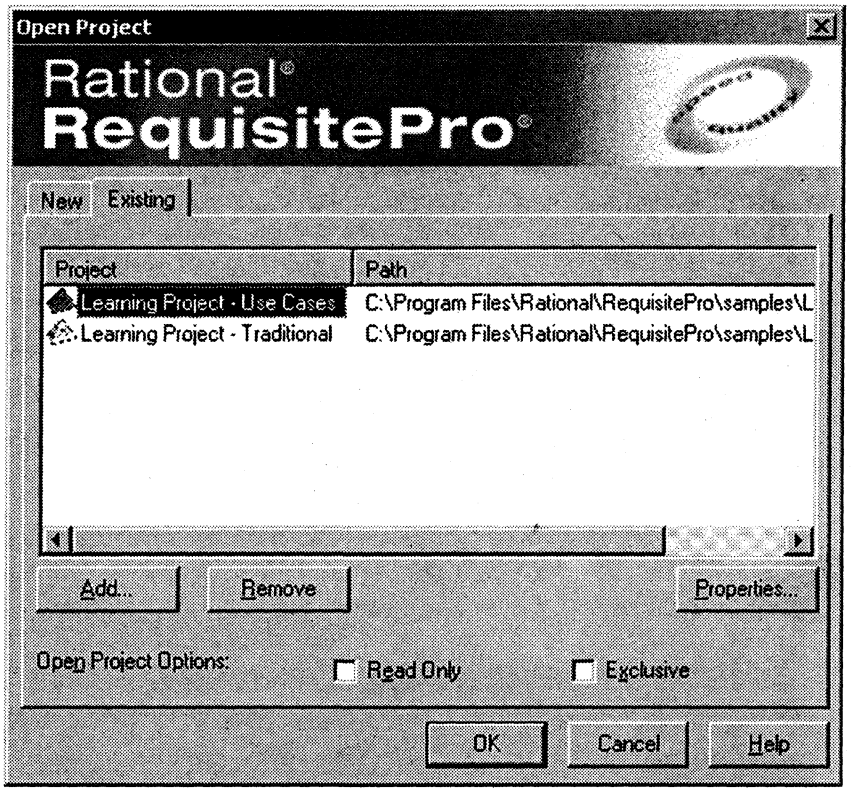


Рис. 1.8. Диалоговое окно "Open Project"

Template", "Traditional Template" и "Use Case Template" содержат готовые наборы типов требований и типов документов, которые можно использовать для того, чтобы приступить к новому проекту RequisitePro. При выборе одного из указанных шаблонов эти типы требований и документов будут добавлены в новый проект. Описание выделенного шаблона можно получить в нижнем поле окна "Create Project" (например, на рис. 1.9 описанием выделенного шаблона "Blank" является "Use this template to create a blank project"). Выбор пустого шаблона "Blank" позволит создать новый проект с чистого листа. Выбор "Make New Template" запускает специальный мастер, позволяющий самостоятельно построить новый шаблон на основе имеющегося проекта. При этом в шаблон войдут уже имеющиеся требования и документы этого проекта.

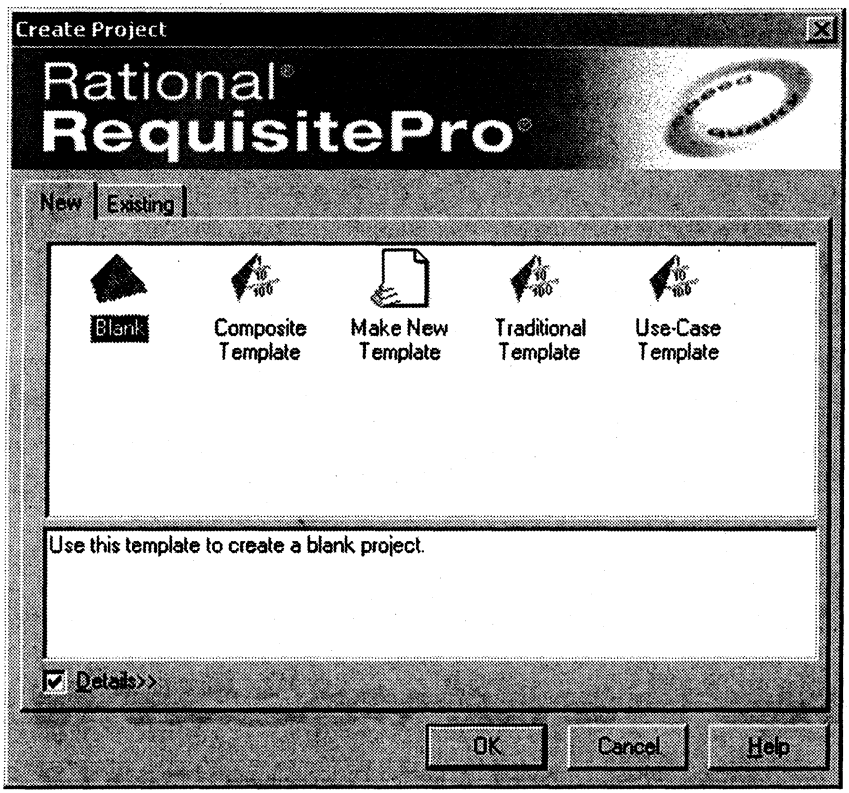


Рис. 1.9. Диалоговое окно Create Project

3. Выберите шаблон "Blank" и щелкните по кнопке Create. Появится новое диалоговое окно "Rational RequisitePro Project Properties" (рис. 1.10), в котором необходимо указать название создаваемого проекта (под которым он будет зарегистрирован в RequisitePro), заранее созданную папку на диске (где будут храниться файлы проекта), тип базы данных (эта база данных будет содержать требования проекта и дополнительную информацию) и описание проекта.

4. Щелкните по кнопке OK для создания проекта. Проект появится в окне Проводника. Если появится сообщение "Project directory does not exist. Do you want to create it?", щелкните по кнопке Yes.

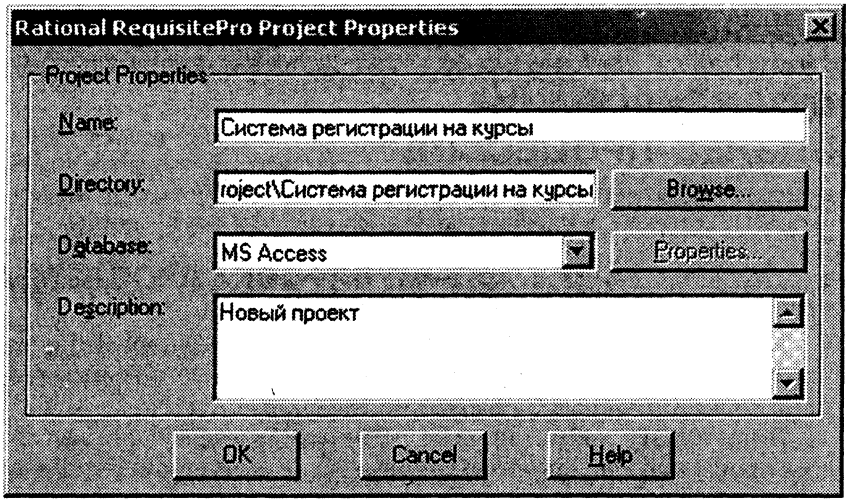


Рис. 1.10. Окно свойств проекта

МОДЕЛИРОВАНИЕ БИЗНЕС-ПРОЦЕССОВ

2.1. ФОРМУЛИРОВКА ПРОБЛЕМЫ

В данном практикуме мы будем рассматривать проблему, связанную с созданием системы регистрации студентов университета на дополнительные платные курсы.

В настоящее время процесс регистрации студентов, желающих прослушать те или иные дополнительные курсы, и распределения курсов между профессорами непрост и продолжителен.

После того как профессора примут решение, какие курсы они собираются преподавать в очередном семестре, сотрудник деканата вводит полученную информацию в базу данных, содержащую всю информацию о курсах (каталог курсов) и распечатывает отчет, содержащий сведения о распределении педагогической нагрузки. Соответствующий каталог курсов также публикуется и передается студентам. Информация о каждом курсе включает имя профессора, наименование кафедры и требования к предварительному уровню подготовки (прослушанным курсам).

Каждый студент заполняет специальную форму, отмечает выбранные курсы и передает форму в деканат. Студент в течение семестра может посещать занятия по четырем курсам. На каждый курс может записаться не более 10 и не менее 3 студентов (если менее 3, то курс не состоится). Данные из форм, полученных от студентов, поступают в систему. Как только вся информация будет введена, выполняется процедура формирования учебного плана. В большинстве случаев первый вариант выбора, предъявленный студентом, оказывается окончательным. Если возникают разногласия или несоответствия, студента приглашают в деканат, где его требования и предпочтения уточняются и учитываются вновь. По завершении согласования студентам рассылаются "твердые" копии расписаний занятий. На весь процесс обычно уходит одна-две недели.

После того как процесс регистрации студента завершен, сотрудник деканата направляет информацию в расчетную систему, чтобы студент мог внести плату за семестр.

После окончания регистрации профессорам передаются списки студентов по каждому курсу.

2.2. СОЗДАНИЕ МОДЕЛИ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ ДЛЯ БИЗНЕС-ПРОЦЕССОВ

Моделирование бизнес-процессов предусматривает построение двух моделей:

- модели вариантов использования для бизнес-процессов (Business Use Case Model);
- модели бизнес-анализа (Business Analysis Model).

Модель Business Use Case – модель, описывающая бизнес-процессы организации в терминах ролей и их потребностей. Она представляет собой расширение модели вариантов использования UML за счет введения набора стереотипов – Business Actor (стереотип действующего лица) и Business Use Case (стереотип варианта использования).

Business Actor (действующее лицо бизнес-процессов) – это некоторая роль, внешняя по отношению к бизнес-процессам организации. Потенциальными кандидатами в действующие лица бизнес-процессов являются:

- акционеры;
- заказчики;
- поставщики;
- партнеры;
- потенциальные клиенты;
- местные органы власти;
- сотрудники подразделений организации, деятельность которых не охвачена моделью;
- внешние системы.

Список действующих лиц составляется по результатам ответов на следующие вопросы:

- кто извлекает пользу из существования организации?
- кто помогает организации осуществлять свою деятельность?
- кому организация передает информацию и от кого получает?

Для правильного распознавания состава действующих лиц следует в первую очередь определить рассматриваемую организа-

цию или область деятельности. В данном случае роль такой организации выполняет деканат, отвечающий за регистрацию студентов на курсы, а следовательно, действующими лицами бизнес-процессов являются:

- студент – записывается на курсы;
- профессор – выбирает курсы для преподавания.

Упражнение 2.1.

Создание действующих лиц бизнес-процессов в среде Rose

При запуске Rose в окне "Create New Model" выберите шаблон "Rational Unified Process" (см. рис. 1.5). Для того чтобы поместить действующее лицо в браузер:

1. Щелкните правой кнопкой мыши по пакету Business Use Case Model представления Use Case View в браузере.
2. Выберите в открывшемся меню пункт New > Actor. В браузере появится новое действующее лицо под названием NewClass. Слева от его имени вы увидите пиктограмму действующего лица UML.
3. Выделите новое действующее лицо и введите его имя.
4. Щелкните правой кнопкой мыши по действующему лицу.
5. Выберите пункт Open Specification в открывшемся меню.
6. Выберите Business Actor в поле стереотипа и щелкните по кнопке OK.
7. Сохраните модель под именем Coursereg с помощью пункта меню File > Save после создания действующих лиц.

Business Use Case (вариант использования с точки зрения бизнес-процессов) определяется как описание последовательности действий в рамках некоторого бизнес-процесса, приносящих ощутимый результат конкретному действующему лицу.

При использовании данной методики необходимо сосредоточить внимание в первую очередь на *элементарных* бизнес-процессах. Элементарный бизнес-процесс можно определить как задачу, выполняемую одним человеком в одном месте в одно время в ответ на некоторое событие, приносящую конкретный результат и переводящую данные в некоторое устойчивое состояние (например, подтверждение платежа по кредитной карточке). Решение такой задачи обычно включает от 5 до 10 шагов и может занимать от нескольких минут до нескольких дней, но рассматривается как один сеанс взаимодействия действующего лица с исполнителями.

Каждый Business Use Case отражает цель или потребность некоторого действующего лица. Исходя из потребностей действую-

щих лиц (студентов и профессоров), можно выделить следующие варианты использования: Зарегистрироваться на курсы и Выбрать курсы для преподавания.

Упражнение 2.2.
Создание вариантов использования для бизнес-процессов в среде Rose

Для того чтобы поместить вариант использования в браузер:

1. Щелкните правой кнопкой мыши по пакету Business Use Case Model представления Use Case View в браузере.
2. Выберите в появившемся меню пункт New > Use Case.
3. Новый вариант использования под названием NewUseCase появится в браузере. Слева от него будет видна пиктограмма варианта использования UML.
4. Выделите новый вариант использования и введите его название.
5. Щелкните правой кнопкой мыши по варианту использования.
6. Выберите пункт Open Specification в открывшемся меню.
7. Выберите Business Use Case в поле стереотипа и щелкните по кнопке OK.

Создадим диаграмму вариантов использования для бизнес-модели деканата. Требуемые для этого действия подробно перечислены ниже. Готовая диаграмма вариантов использования приведена на рис. 2.1.

Упражнение 2.3.
Построение диаграммы вариантов использования для бизнес-модели

Для создания новой диаграммы вариантов использования:

1. Откройте пакет Business Use Case Model в представлении вариантов использования (Use Case View), щелкнув левой кнопкой мыши по значку "+" слева от него.
2. Щелкните дважды по названию диаграммы Global View of Business Actors and Business Use Cases в браузере, чтобы открыть ее.
3. Перетащите мышью из браузера действующее лицо или вариант использования на диаграмму вариантов использования.
4. Нарисуйте ассоциации между действующими лицами и вариантами использования с помощью кнопки Unidirectional

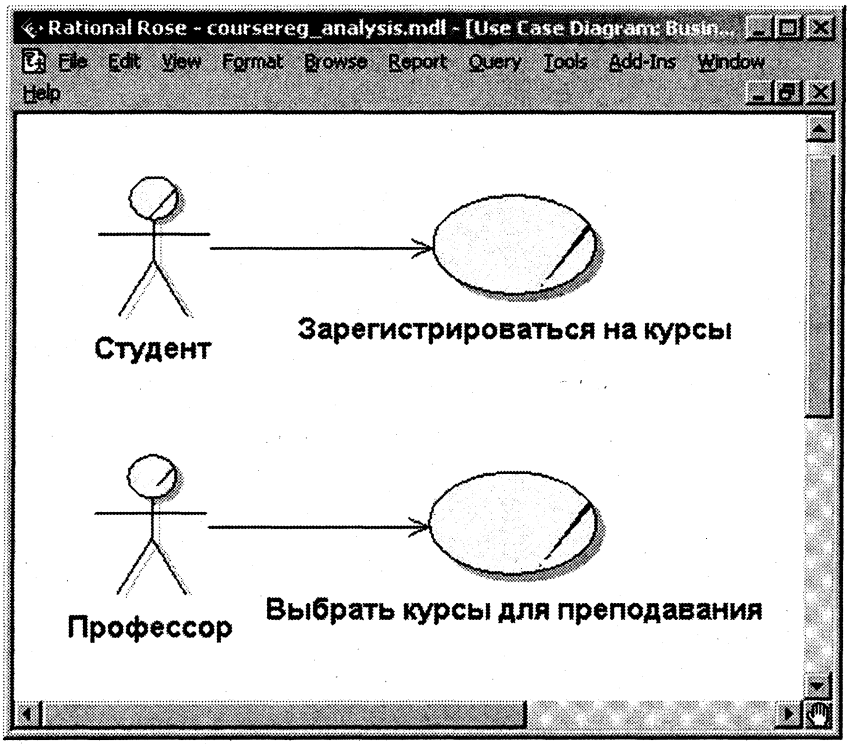


Рис. 2.1. Диаграмма вариантов использования бизнес-модели

Association (Однонаправленная ассоциация) панели инструментов.

Описание Business Use Case представляет собой спецификацию, которая состоит из следующих пунктов:

- наименование;
- краткое описание;
- цели и результаты (с точки зрения действующего лица);
- описание сценариев (основного и альтернативных);
- специальные требования (ограничения по времени выполнения или другим ресурсам);
- расширения (частные случаи);
- связи с другими Business Use Case;
- диаграммы деятельности (для наглядного описания сценариев – при необходимости).

Упражнение 2.4.

Добавление описаний к вариантам использования

Для добавления описаний:

1. Выделите в браузере вариант использования "Зарегистрироваться на курсы".
2. Введите в окне документации следующее описание к этому варианту использования: "Данный Business Use Case позволяет студенту зарегистрироваться на конкретные курсы в текущем семестре".
3. Создайте с помощью MS Word текстовый файл с приведенным ниже описанием варианта использования "Зарегистрироваться на курсы".

Спецификация Business Use Case "Зарегистрироваться на курсы"

Наименование:

Зарегистрироваться на курсы.

Краткое описание:

Данный Business Use Case позволяет студенту зарегистрироваться на предлагаемые курсы в текущем семестре.

Основной сценарий:

1. Студент приходит к сотруднику деканата и передает ему заполненную форму регистрации на курсы.
2. Сотрудник деканата подтверждает правильность заполнения формы.
3. Сотрудник деканата подтверждает, что студент выполнил предварительные требования для каждого выбранного курса (прохождение определенных курсов), а также наличие свободных мест.
4. Сотрудник деканата вносит студента в список каждого курса, выбранного им.
5. Сотрудник деканата формирует график студента на текущий семестр и передает его студенту.

Альтернативные сценарии:

2а. Неправильно заполнена форма регистрации.

Сотрудник деканата возвращает студенту форму для исправления ошибок.

3а. Не выполнены предварительные требования или заполнен курс:

Если сотрудник деканата обнаружит, что студент не выполнил необходимые предварительные требования или выбранный им курс заполнен (уже записались 10 студентов), то он предлагает студенту изменить свой выбор курсов либо забрать форму и вернуться к нему позже.

Упражнение 2.5.
Прикрепление файла к варианту использования

Для прикрепления файла к варианту использования:

1. Щелкните правой кнопкой мыши по варианту использования.
2. Выберите пункт Open Specification в открывшемся меню.
3. Перейдите на вкладку Files.
4. Щелкните правой кнопкой мыши по белому полю и выберите из открывшегося меню пункт Insert File.
5. Укажите созданный ранее файл и щелкните по кнопке Open, чтобы прикрепить файл к варианту использования.

2.3. СОЗДАНИЕ МОДЕЛИ БИЗНЕС-АНАЛИЗА

Для каждого Business Use Case строится *модель бизнес-анализа* – объектная модель, описывающая реализацию бизнес-процесса в терминах взаимодействующих объектов (бизнес-объектов – Business Object), принадлежащих к двум классам – Business Worker и Business Entity.

Business Worker (исполнитель) – активный класс, представляющий собой абстракцию исполнителя, выполняющего некоторые действия в рамках бизнес-процесса. Исполнители взаимодействуют между собой и манипулируют различными сущностями, участвуя в реализациях сценариев Business Use Case. На диаграмме классов UML исполнитель представляется в виде класса со стереотипом <<business worker>>.

Business Entity (сущность) – пассивный класс, не инициирующий никаких взаимодействий. Объект такого класса может участвовать в реализациях различных Business Use Case. Сущность является объектом различных действий со стороны исполнителей. На диаграмме классов UML сущность представляется в виде класса со стереотипом <<business entity>>.

Модель бизнес-анализа может состоять из диаграмм разных типов. В состав модели обязательно должна входить диаграмма классов, содержащая исполнителей и сущности.

Согласно формулировке проблемы, роль исполнителя выполняет сотрудник деканата (назовем его "регистратор"), который формирует учебный план и каталог курсов, записывает студентов на курсы, ведет все данные о курсах, профессорах и студентах. Сущностями, которыми он манипулирует, являются:

- студент;
- профессор;
- график студента (список курсов);
- курс (в программе обучения);
- предлагаемый курс (курс в расписании).

Для Business Use Case "Зарегистрироваться на курсы" список сущностей будет следующим:

- студент;
- график студента;
- предлагаемый курс.

Диаграмма классов для модели бизнес-анализа, описывающей Business Use Case "Зарегистрироваться на курсы", приведена на рис. 2.2 (для данных классов использовано изображение сте-

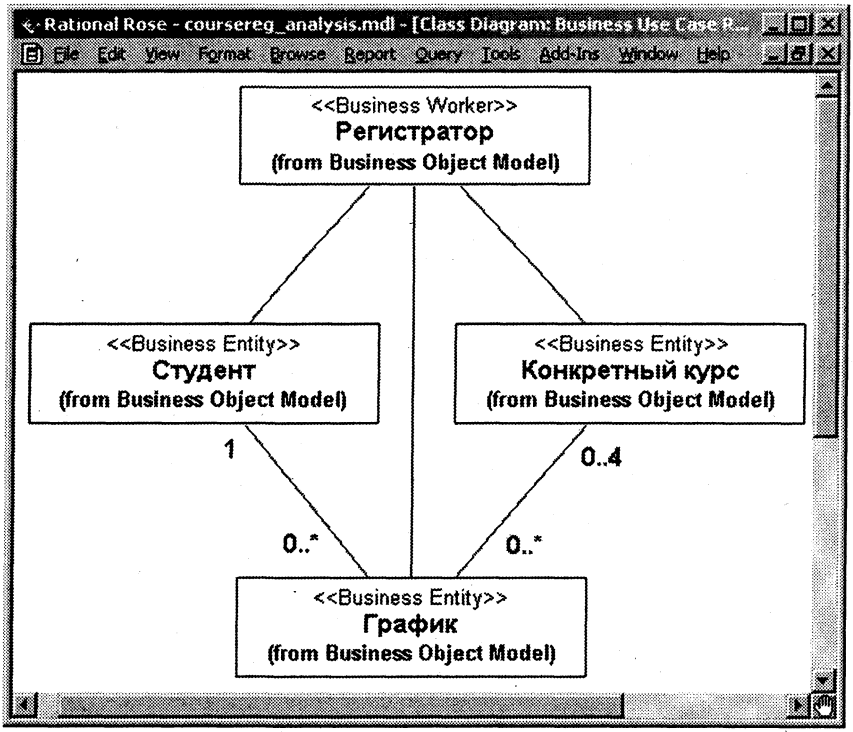


Рис. 2.2. Диаграмма классов модели бизнес-анализа

реотипа в виде метки – label). Настройка изображения стереотипа может быть выполнена следующими способами:

1. Для всей модели – в меню Tools > Options > Diagram > Stereotype Display.
2. Для отдельного элемента модели – в его контекстном меню Options > Stereotype Display.
3. Для нескольких сгруппированных элементов модели – в меню Format > Stereotype Display.

Упражнение 2.6.

Создание классов, участвующих в реализации бизнес-процесса "Зарегистрироваться на курсы", и кооперации, описывающей реализацию бизнес-процесса

Для создания классов, кооперации и диаграммы классов:

1. Щелкните правой кнопкой мыши по пакету Business Object Model представления Logical View в браузере.
2. Выберите в открывшемся меню пункт New > Class. Новый класс под названием NewClass появится в браузере.
3. Выделите его и введите имя "Регистратор".
4. Щелкните правой кнопкой мыши по данному классу.
5. Выберите в открывшемся меню пункт Open Specification.
6. Выберите Business Worker в поле стереотипа и щелкните по кнопке OK.
7. Создайте аналогичным образом классы-сущности со стереотипом <<Business Entity>>.
8. Щелкните правой кнопкой мыши по пакету Business Object Model представления Logical View в браузере.
9. Выберите в открывшемся меню пункт New > Package.
10. Назовите новый пакет Business Use Case Realizations.
11. Создайте кооперацию "Зарегистрироваться на курсы" (кооперация представляет собой вариант использования со стереотипом "business use-case realization", который задается в спецификации варианта использования) в пакете Business Use Case Realizations.
12. Щелкните правой кнопкой мыши по созданной кооперации.
13. Выберите в открывшемся меню пункт New > Class Diagram.
14. Назовите новую диаграмму классов Participating Classes.

15. Откройте диаграмму классов Participating Classes и перетащите классы на открытую диаграмму в соответствии с рис. 2.2.

Для того чтобы на диаграмме классов создать ассоциацию:

1. Щелкните на панели инструментов по кнопке Association (после установки Rose эта кнопка по умолчанию не показана на панели. Для вывода кнопки необходимо настроить панель).

2. Проведите мышью линию ассоциации от одного класса к другому.

Для того чтобы задать мощность связи:

1. Щелкните правой кнопкой мыши по одному концу связи.

2. Выберите в открывшемся контекстном меню связи пункт Multiplicity.

3. Укажите нужную мощность.

4. Повторите действия п. 1–3 для другого конца связи.

5. Откройте спецификацию связи (дважды щелкнув по ней левой кнопкой мыши) и установите точное значение мощности в поле Multiplicity на вкладке "Role A Detail" или "Role B Detail", если требуемое значение мощности не совпадает со стандартными значениями в контекстном меню связи.

Кроме диаграммы классов, модель бизнес-анализа может включать:

- диаграммы деятельности с потоками объектов и "плавательными дорожками", описывающие взаимосвязи между сценариями одного или различных Business Use Case;
- диаграммы последовательности (и кооперативные диаграммы), описывающие сценарии Business Use Case в виде последовательности обмена сообщениями между объектами-действующими лицами и объектами-исполнителями. Такие диаграммы помогают явно определить в модели обязанности каждого исполнителя в виде набора его операций;
- диаграммы состояний, описывающие поведение отдельных бизнес-объектов.

Упражнение 2.7.
Создание диаграммы деятельности

Для описания процесса формирования каталога курсов и рассылки его студентам можно построить следующую диаграмму деятельности (рис. 2.3).

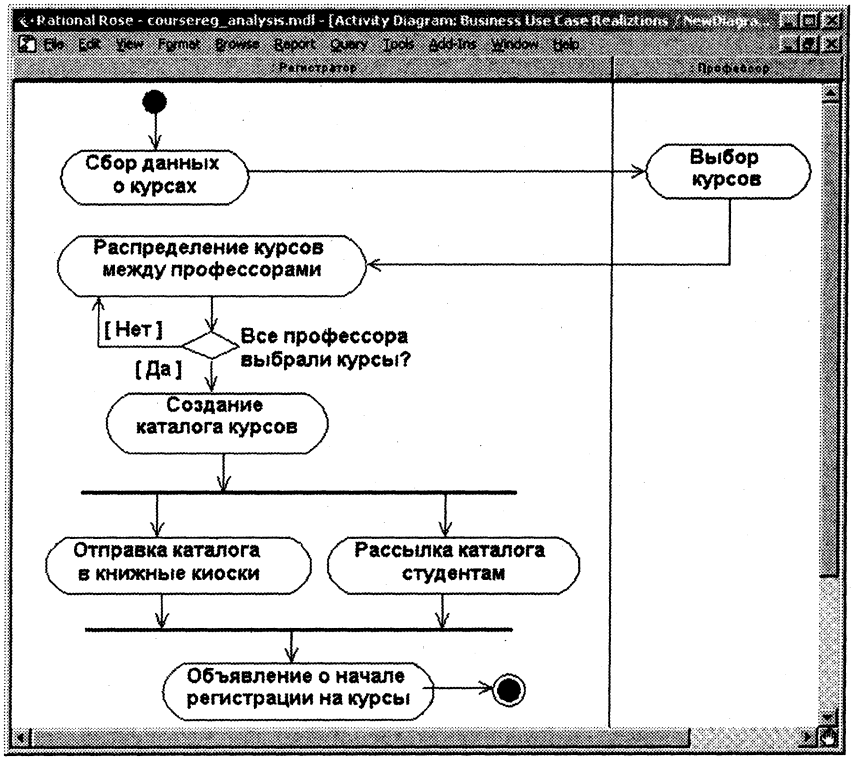


Рис. 2.3. Диаграмма деятельности

Для того чтобы добавить диаграмму деятельности:

1. Щелкните правой кнопкой по пакету Business Use Case Realizations в браузере.
2. Выполните в меню команду New > Activity Diagram.
3. Rose создаст в браузере элемент State/Activity Model, находящийся ниже пакета.
4. Присвойте имя новой диаграмме деятельности.
5. Щелкните дважды по этой диаграмме, чтобы открыть ее.

В созданную диаграмму деятельности с помощью панели инструментов добавляются "плавательные дорожки", деятельности и другие объекты.

Для добавления в диаграмму дорожек:

1. Выберите на панели инструментов кнопку Swimlane.

2. Щелкните внутри диаграммы. Появится новая дорожка с присвоенным по умолчанию именем NewSwimlane.

3. Откройте спецификацию дорожки, щелкнув правой кнопкой по ее имени NewSwimlane и выбрав пункт Open Specification.

4. Присвойте дорожке новое имя, удалив имя в поле Name и выбрав в списке поля Class значение "Регистратор" из модели Business Object (рис. 2.4).

5. Создайте еще одну дорожку с именем "Профессор" из модели Business Use Case.

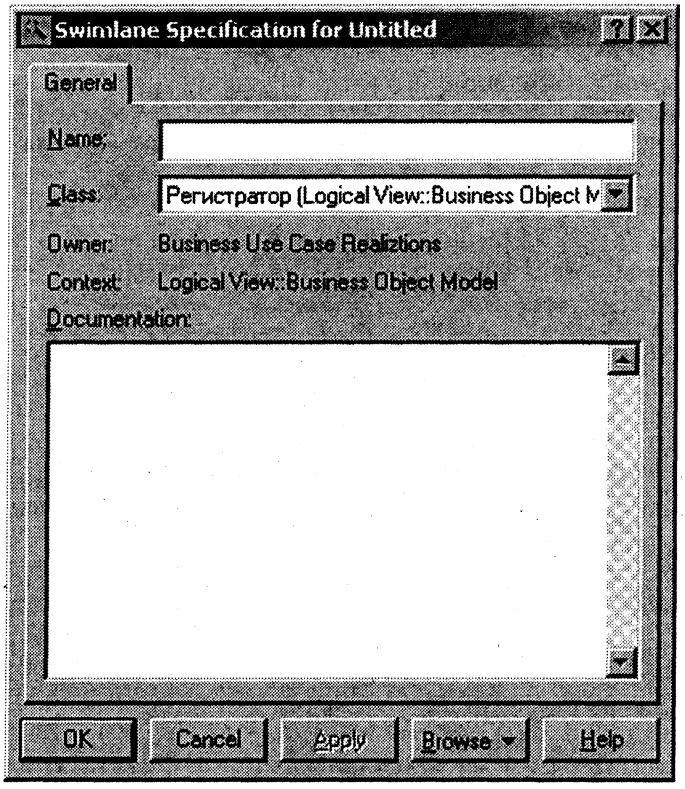


Рис. 2.4. Спецификация "плавательной дорожки"

Для того чтобы добавить в диаграмму начальное (конечное) состояние:

1. Выберите на панели инструментов кнопку Start (End) State.

2. Щелкните внутри диаграммы и внутри соответствующей дорожки.

Для того чтобы добавить в диаграмму новую деятельность:

1. Выберите на панели инструментов кнопку Activity.

2. Щелкните внутри диаграммы и внутри нужной дорожки, которая определяет действующее лицо или исполнителя, отвечающего за выполнение деятельности.

3. Присвойте имя новой деятельности.

Для того чтобы создать переход между деятельностями:

1. Выберите на панели инструментов кнопку State Transition.

2. Перетащите указатель мыши с одной деятельности на другую.

Для того чтобы добавить точку принятия решения:

1. Выберите на панели инструментов кнопку Decision.

2. Щелкните внутри диаграммы для вставки решения.

3. Откройте окно спецификаций для решения и введите его имя (рис. 2.5).

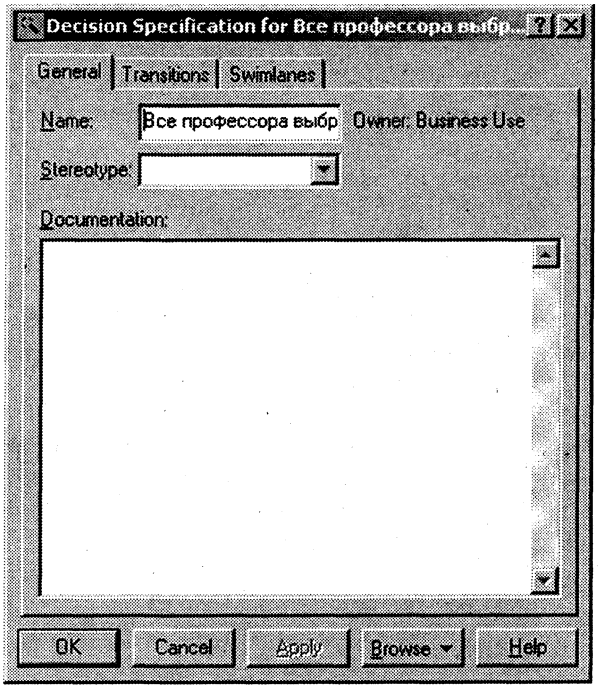


Рис. 2.5. Спецификация решения

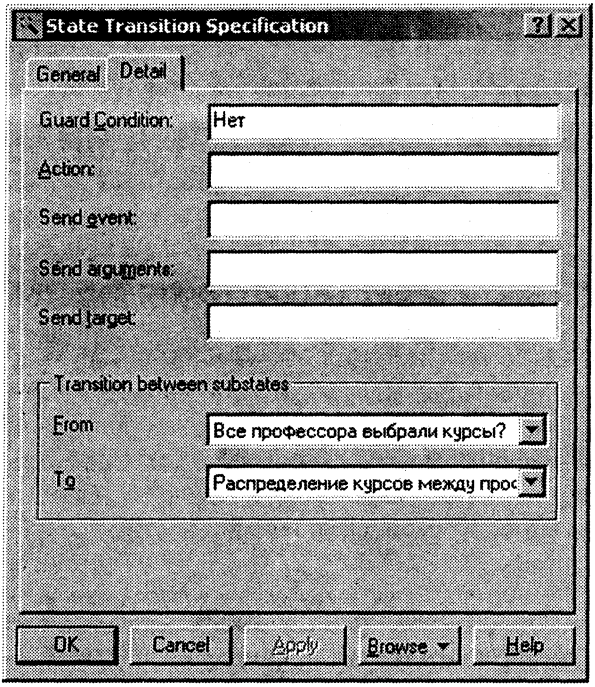


Рис. 2.6. Спецификация перехода

4. Нарисуйте переход от деятельностей к решению или от решения к одной или нескольким деятельностям.

Для того чтобы добавить граничное условие:

1. Щелкните правой кнопкой по переходу.

2. Выберите Open Specification – откроется окно спецификаций для перехода.

3. Перейдите на вкладку "Details" (рис. 2.6).

4. Введите граничное условие в поле Guard Condition (можно ввести граничное условие непосредственно на стрелку перехода, заключив это условие в квадратные скобки).

Для того чтобы добавить линейки синхронизации:

1. Выберите на панели значок Vertical Synchronization или Horizontal Synchronization.

2. Щелкните внутри диаграммы для вставки линейки синхронизации.

3. Нарисуйте переходы между синхронизуемыми деятельностями.

СПЕЦИФИКАЦИЯ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

3.1. ПОСТАНОВКА ЗАДАЧИ РАЗРАБОТКИ НОВОЙ СИСТЕМЫ РЕГИСТРАЦИИ

Требования к ПО документируются в виде ряда документов и моделей. К основным документам относятся:

- концепция – определяет глобальные цели проекта и основные особенности разрабатываемой системы. Существенной частью концепции является постановка задачи разработки (подразд. 3.1), определяющая требования к выполняемым системой функциям*;
- словарь предметной области (глоссарий) – устанавливает общую терминологию для всех моделей и описаний требований к системе. Глоссарий (подразд. 3.2) предназначен для описания терминологии предметной области и может быть использован как словарь данных системы;
- дополнительные спецификации (технические требования) – содержат описание нефункциональных требований к системе, таких, как надежность, удобство использования, производительность, сопровождаемость и др. (подразд. 3.3).

В целях ускорения и повышения эффективности процесса регистрации на курсы университета (см. подразд. 2.1) перед руководителем информационной службы университета ставится задача разработки новой клиент-серверной системы регистрации студентов взамен старой системы на мейнфрейме. Новая система должна позволять студентам регистрироваться на курсы и просматривать свои табели успеваемости с персональных компьютеров, подключенных к локальной сети университета. Профессора должны иметь доступ к онлайновой системе, чтобы ука-

* Полный вариант концепции для системы регистрации, соответствующий шаблону технологии Rational Unified Process, приведен в приложении 1.

зать курсы, которые они будут читать, и проставить оценки за курсы.

Университет не в состоянии заменить сразу всю существующую систему из-за недостатка средств. База данных, содержащая всю информацию о курсах (каталог курсов), остается функционировать в прежнем виде, она поддерживается реляционной СУБД. Новая система будет работать с существующей БД в режиме доступа, без обновления.

В начале каждого семестра студенты могут запросить каталог курсов, содержащий список курсов, предлагаемых для изучения в данном семестре. Информация о каждом курсе должна включать имя профессора, наименование кафедры и требования к предварительному уровню подготовки (прослушанным курсам).

Новая система должна позволять студентам выбирать четыре курса в предстоящем семестре. В дополнение каждый студент может указать два альтернативных курса на тот случай, если какой-либо из выбранных им курсов окажется уже заполненным или отмененным. На каждый курс могут записаться не более 10 и не менее 3 студентов (если менее 3, то курс будет отменен). В каждом семестре существует период времени, когда студенты могут изменить свои планы. В это время студенты должны иметь доступ к системе, чтобы добавить или удалить выбранные курсы. После того как процесс регистрации некоторого студента завершен, система регистрации направляет информацию в расчетную систему, чтобы студент мог внести плату за семестр. Если курс окажется заполненным в процессе регистрации, студент должен быть извещен об этом до того, как его личный учебный план будет окончательно сформирован.

В конце семестра студенты должны иметь доступ к системе для просмотра своих электронных табелей успеваемости. Поскольку эта информация конфиденциальная, система должна обеспечивать ее защиту от несанкционированного доступа.

Профессора должны иметь доступ к онлайновой системе, чтобы указать курсы, которые они будут читать, и просмотреть список студентов, записавшихся на их курсы. Кроме этого, профессора должны иметь возможность проставить оценки за курсы.

3.2. СОСТАВЛЕНИЕ ГЛОССАРИЯ ПРОЕКТА

Глоссарий предназначен для описания терминологии предметной области. Он может быть использован как неформальный *словарь данных* системы.

Ниже приведены термины проекта и их значения.

Термин	Значение
Курс	Учебный курс, предлагаемый университетом.
Предлагаемый курс	Предлагаемое чтение данного курса в конкретном семестре (один и тот же курс может вестись в нескольких параллельных сессиях). Включает конкретные дни недели и время.
Каталог курсов	Полный каталог всех курсов, предлагаемых университетом.
Расчетная система	Система обработки информации об оплате за курсы.
Оценка	Оценка, полученная студентом за конкретный курс.
Профессор	Преподаватель университета.
Табель успеваемости	Все оценки за все курсы, полученные студентом в данном семестре.
Список курса	Список всех студентов, записавшихся на предлагаемый курс.
Студент	Личность, проходящая обучение в университете.
Учебный график	Курсы, выбранные студентом в текущем семестре.

3.3. ОПИСАНИЕ ДОПОЛНИТЕЛЬНЫХ СПЕЦИФИКАЦИЙ

Назначение дополнительных спецификаций – определить требования к системе регистрации курсов, которые не охватывает модель вариантов использования. Вместе они образуют полный набор требований к системе.

Дополнительные спецификации определяют нефункциональные требования к системе, такие, как удобство использования, надежность, производительность, а также ряд функциональных требований, являющихся общими для нескольких вариантов использования: безопасность, проектные ограничения.

Функциональные возможности

Система должна обеспечивать многопользовательский режим работы.

Удобство использования

Пользовательский интерфейс должен быть Windows-совместимым.

Пользовательский интерфейс системы должен быть простым и не требующим дополнительного обучения для пользователей, обладающих компьютерной грамотностью.

Каждая функция системы должна сопровождаться встроенной онлайновой помощью, которая должна включать пошаговые инструкции по работе с системой, а также определения терминов и сокращений.

Надежность

Система должна быть в работоспособном состоянии 24 ч в день 7 дней в неделю, время простоя – не более 10%. Среднее время безотказной работы должно превышать 300 ч.

Производительность

Система должна поддерживать до 2000 пользователей, одновременно работающих с центральной базой данных, и до 500 пользователей, одновременно работающих с локальными серверами.

Система должна обеспечивать доступ к унаследованной БД каталога курсов со временем ожидания не более 10 с.

Система должна быть способна завершать 80% всех транзакций в течение 2 мин.

Безопасность

Система не должна позволять студентам изменять любые учебные графики, кроме своих собственных, а также позволять профессорам модифицировать конкретные курсы, выбранные другими профессорами.

Только профессора имеют право ставить оценки студентам.

Только регистратор может изменять любую информацию о студентах.

Проектные ограничения

Система должна быть интегрирована с существующей системой каталога курсов, функционирующей на основе реляционной СУБД.

3.4. СОЗДАНИЕ НАЧАЛЬНОЙ ВЕРСИИ МОДЕЛИ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Функциональные требования к системе моделируются и документируются с помощью вариантов использования (use case), которые трактуются следующим образом:

- вариант использования фиксирует соглашение между участниками проекта относительно поведения системы;
- вариант использования описывает поведение системы при различных условиях, когда система отвечает на запрос одного из участников, называемого основным действующим лицом;
- основное действующее лицо инициирует взаимодействие с системой, чтобы добиться некоторой цели. Система отвечает, соблюдая интересы всех участников.

Варианты использования – это вид документации, применяемой, когда требуется сконцентрировать усилия на обсуждении принципиальных требований к разрабатываемой системе, а не на подробном описании их. Стиль их написания зависит от масштаба, количества участников и критичности проекта. При описании вариантов использования (расположенных по степени повышения точности) существуют четыре уровня точности:

- действующие лица и цели (перечисляются действующие лица и все их цели, которые будет обеспечивать система);
- краткое изложение варианта использования (в один абзац) или основной поток событий (без анализа возможных ошибок);
- условия отказа (анализ мест возникновения возможных ошибок в основном потоке событий);
- обработка отказа (написание альтернативных потоков событий).

Спецификация требований в технологии Rational Unified Process не предполагает обязательного моделирования бизнес-процессов организации, для которых создается ПО, однако наличие бизнес-модели существенно упрощает построение системной модели вариантов использования. При переходе от бизнес-модели к начальной версии модели вариантов использования выполняются следующие правила:

- для каждого исполнителя в модели бизнес-анализа, который в перспективе станет пользователем новой системы, в модели вариантов использования создается действующее лицо с та-

ким же наименованием. В состав действующих лиц включаются также внешние системы, играющие в бизнес-процессах пассивную роль источников информации;

• варианты использования для данного действующего лица создаются на основе анализа обязанностей соответствующего исполнителя (в простейшем случае для каждой операции исполнителя создается вариант использования, реализующий данную операцию в системе).

Такая начальная версия модели описывает минимальный вариант системы, пользователями которой являются только исполнители в бизнес-процессах. Если в дальнейшем в процессе развития системы ее непосредственными пользователями будут действующие лица бизнес-процессов, то модель вариантов использования начнет модифицироваться.

Применение данных правил для системы регистрации приводит к появлению следующего списка действующих лиц для начальной версии системы:

• регистратор – формирует учебный план и каталог курсов, записывает студентов на курсы, ведет все данные о курсах, профессорах, успеваемости и студентах;

• расчетная система – получает информацию по оплате курсов от данной системы.

• каталог курсов – база данных, содержащая информацию о курсах.

Упражнение 3.1.
Создание действующих лиц в среде Rose

Для того чтобы поместить действующее лицо в браузер:

1. Щелкните правой кнопкой мыши по пакету Actors, входящему в пакет Use Case Model представления вариантов использования в браузере.

2. Выберите в открывшемся меню пункт New > Actor.

3. В браузере появится новое действующее лицо под названием NewClass. Слева от его имени вы увидите пиктограмму действующего лица UML.

4. Выделив новое действующее лицо, введите его имя.

Исходя из потребностей действующих лиц, выделяются следующие варианты использования:

• Войти в систему.

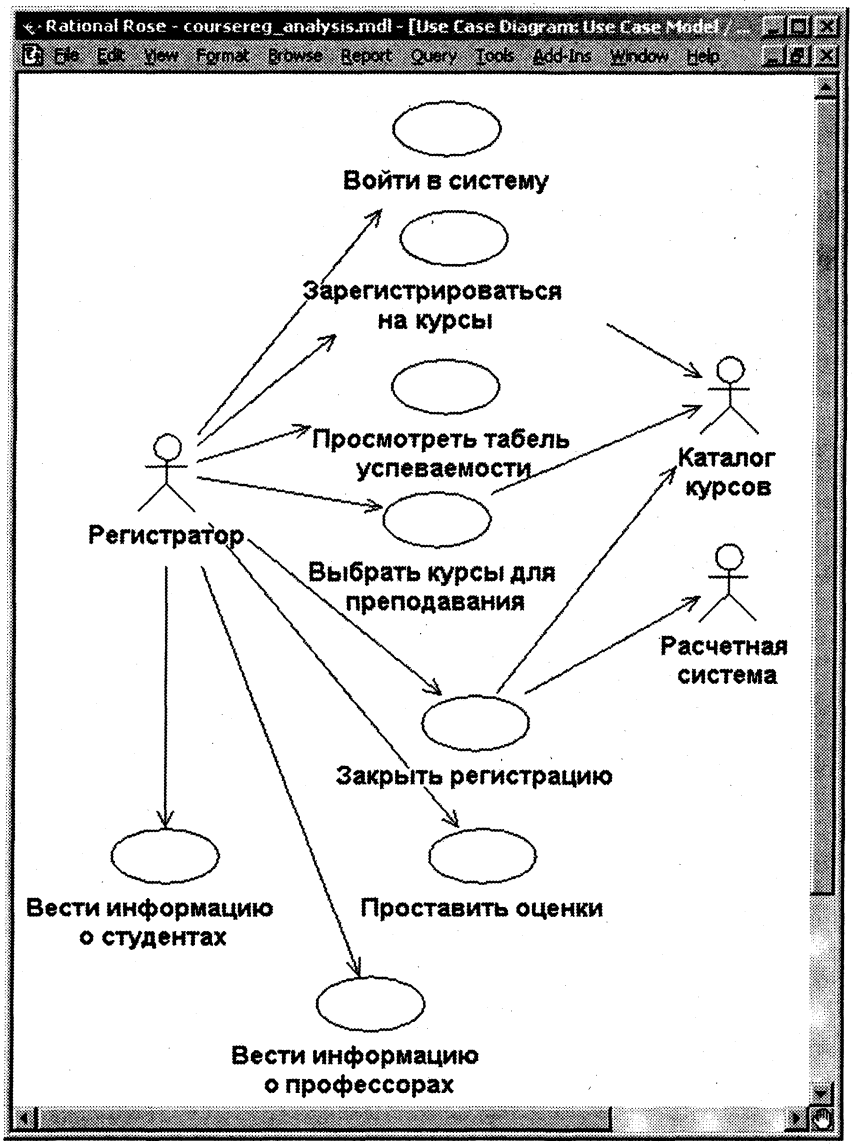


Рис. 3.1. Начальная версия диаграммы вариантов использования

- Зарегистрироваться на курсы.
- Просмотреть табель успеваемости.
- Выбрать курсы для преподавания.
- Проставить оценки.
- Вести информацию о профессорах.
- Вести информацию о студентах.
- Закрыть регистрацию.

Начальная версия диаграммы вариантов использования показана на рис. 3.1.

Упражнение 3.2.

Создание вариантов использования в среде Rational Rose

Для того чтобы поместить вариант использования в браузер:

1. Щелкните правой кнопкой мыши по пакету Use Cases, входящему в пакет Use Case Model представления вариантов использования в браузере.

2. Выберите в появившемся меню пункт New > Use Case.

3. В браузере появится новый вариант использования под названием NewUseCase. Слева от него будет видна пиктограмма варианта использования UML.

4. Выделите новый вариант использования и введите его название.

Создадим начальную диаграмму вариантов использования для системы регистрации. Требуемые для этого действия подробно перечислены ниже. Готовая диаграмма вариантов использования должна выглядеть, как на рис. 3.1.

В среде Rose диаграммы вариантов использования создаются в представлении вариантов использования. Главная диаграмма (Global View of Actors and Use Cases) предлагается по умолчанию. Затем для моделирования системы можно разработать столько дополнительных диаграмм, сколько необходимо.

Для создания новой диаграммы вариантов использования:

1. Щелкните правой кнопкой мыши по пакету Use Case Model представления вариантов использования в браузере.

2. Выберите из всплывающего меню пункт New > Use Case Diagram.

3. Выделите новую диаграмму, введя ее имя.

4. Щелкните дважды по названию новой диаграммы в браузере, чтобы открыть ее.

Упражнение 3.3.

Построение начальной версии диаграммы вариантов использования

Для того чтобы построить начальную версию диаграммы вариантов использования:

1. Откройте диаграмму вариантов использования Global View of Actors and Use Cases.
2. Удалите с нее все элементы.
3. Чтобы поместить действующее лицо или вариант использования на диаграмму, перетащите его мышью из браузера на диаграмму вариантов использования.
4. Нарисуйте с помощью кнопки Unidirectional Association панели инструментов ассоциации между действующими лицами и вариантами использования.

3.5. МОДИФИКАЦИЯ МОДЕЛИ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Согласно постановке задачи в состав пользователей системы следует ввести студентов и профессоров. При этом в описание действующих лиц и вариантов использования вносятся изменения. Модифицированная версия диаграммы вариантов использования показана на рис. 3.2. Поскольку вход в систему абсолютно одинаков для регистратора, студента и профессора, их поведение можно обобщить и ввести новое действующее лицо "Пользователь" (супертип) с общим вариантом использования "Войти в систему", подтипами которого являются Регистратор, Студент и Профессор.

Действующие лица:

- Студент – записывается на курсы и просматривает табель успеваемости.
- Профессор – выбирает курсы для преподавания и ставит оценки.
- Регистратор – формирует учебный план и каталог курсов, ведет все данные о курсах, профессорах и студентах.
- Расчетная система – получает от данной системы информацию по оплате за курсы.

• Каталог курсов – база данных, содержащая информацию о курсах.

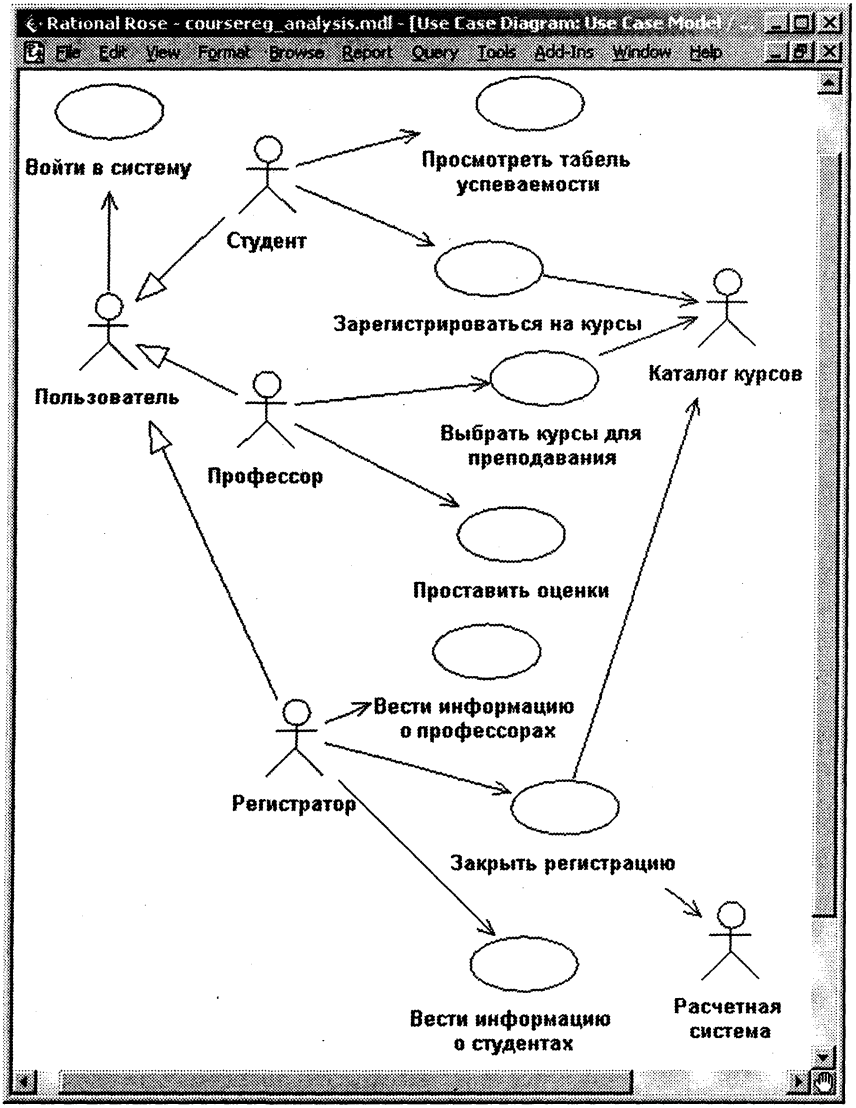


Рис. 3.2. Модифицированная диаграмма вариантов использования

Варианты использования:
- Войти в систему;
- Зарегистрироваться на курсы;
- Просмотреть табель успеваемости;
- Выбрать курсы для преподавания;
- Проставить оценки;
- Вести информацию о профессорах;
- Вести информацию о студентах;
- Закрыть регистрацию.

Упражнение 3.4.
Добавление описаний к вариантам использования

Для добавления описаний:

1. Выделите в браузере вариант использования "Зарегистрироваться на курсы".

2. Введите в окне документации следующее описание к этому варианту использования: "Этот вариант использования дает студенту возможность зарегистрироваться на курсы в текущем семестре".

3. Создайте с помощью MS Word три текстовых файла с приведенными ниже описаниями вариантов использования: "Войти в систему", "Зарегистрироваться на курсы" и "Закрыть регистрацию"*. Имена файлов должны совпадать с наименованиями вариантов использования.

При определении вариантов использования нужно назначить каждому из них приоритет, определяющий порядок дальнейшей реализации их в проекте.

Для того чтобы назначить варианту использования приоритет:

1. Щелкните правой кнопкой мыши по варианту использования в браузере или на диаграмме вариантов использования.

2. Выберите пункт Open Specification в открывшемся меню.

3. Введите приоритет в поле Rank на вкладке General.

---

* Описания остальных вариантов использования для системы регистрации приведены в приложении 1.

Вариант использования "Войти в систему"

Краткое описание:

Данный вариант использования описывает вход пользователя в систему регистрации курсов.

Основной поток событий:

Данный вариант использования начинает выполняться, когда пользователь хочет войти в систему регистрации курсов.

1. Система запрашивает имя пользователя и пароль.
2. Пользователь вводит имя и пароль.
3. Система подтверждает имя и пароль, после чего открывается доступ в систему.

Альтернативные потоки:

Неправильное имя/пароль:

Если во время выполнения основного потока обнаружится, что пользователь ввел неправильное имя и/или пароль, система выводит сообщение об ошибке. Пользователь может вернуться к началу основного потока или отказаться от входа в систему, при этом выполнение варианта использования завершается.

Вариант использования "Зарегистрироваться на курсы"

Краткое описание:

Данный вариант использования позволяет студенту зарегистрироваться на предлагаемые курсы в текущем семестре. Студент может изменить свой выбор (обновить или удалить курсы), если изменение выполняется в установленное время в начале семестра. Система каталога курсов предоставляет список всех предлагаемых курсов текущего семестра.

Основной поток событий:

Данный вариант использования начинает выполняться, когда студент хочет зарегистрироваться на конкретные курсы или изменить свой график курсов.

1. Система запрашивает требуемое действие (создать график, обновить график, удалить график).
2. Когда студент указывает действие, выполняется один из подчиненных потоков (создать, обновить, удалить или принять график).

Создать график:

1. Система выполняет поиск в каталоге курсов доступных предлагаемых курсов и выводит их список.
2. Система отображает пустой график для заполнения.
3. Студент выбирает из списка четыре основных и два альтернативных курса для включения в график.
4. Для каждого выбранного курса выполняется подчиненный поток "Добавить курс в график".
5. Система сохраняет график студента.

Обновить график:

1. Система выводит текущий график студента.
2. Система выполняет поиск в каталоге курсов доступных предлагаемых курсов и выводит их список.
3. Студент может обновить свой выбор курсов, удаляя или добавляя предлагаемые курсы.
4. Для каждого выбранного курса выполняется подчиненный поток "Добавить курс в график".
5. Система сохраняет график студента.

Удалить график:

1. Система выводит текущий график студента.
2. Система запрашивает у студента подтверждение удаления графика.
3. Студент подтверждает удаление.
4. Система удаляет график. Если график включает предлагаемые курсы, на которые записался студент, он должен быть удален из списков этих курсов.

Добавить курс в график:

Для каждого выбранного курса система проверяет факт выполнения студентом предварительных требований (прохождение определенных курсов) и наличие приема на предлагаемый курс. Затем система добавляет студента в список выбранного курса. Курс отмечается в графике как "зарегистрированный".

Альтернативные потоки:

Сохранить график:

Студент может сохранить график в любой момент, не фиксируя в нем выбранные курсы. В этом случае график сохраняется в системе, но система не добавляет студента в списки выбранных курсов. Курсы отмечаются в графике как "выбранные".

Не выполнены предварительные требования или курс заполнен:

Если во время выполнения подчиненного потока "Добавить курс в график" система обнаружит, что студент не выполнил необходимые предварительные требования или выбранный им курс заполнен, то выдается сообщение об ошибке. Студент может либо выбрать другой курс и продолжить выполнение варианта использования, либо отменить операцию, после чего основной поток начнется сначала.

График не найден:

Если во время выполнения подчиненных потоков "Обновить график" или "Удалить график" система не может найти график студента, то выдается сообщение об ошибке. После того как студент подтвердит это сообщение, основной поток начнется сначала.

Система каталога курсов недоступна:

Если окажется, что установить связь с системой каталога курсов невозможно, то будет выдано сообщение об ошибке. После того как студент подтвердит это сообщение, вариант использования завершится.

Регистрация на курсы закончена:

Если в самом начале выполнения варианта использования окажется, что регистрация на текущий семестр закончена, будет выдано сообщение и вариант использования завершится.

Предусловия:

Перед началом выполнения данного варианта использования студент должен войти в систему.

Вариант использования "Закрыть регистрацию"

Краткое описание:

Данный вариант использования позволяет регистратору закрывать процесс регистрации. Предлагаемые курсы, на которые не записалось достаточного количества студентов (менее трех), отменяются. В расчетную систему передается информация о каждом студенте по каждому предлагаемому курсу, чтобы студенты могли внести оплату за курсы.

Основной поток событий:

Данный вариант использования начинает выполняться, когда регистратор запрашивает прекращение регистрации.

1. Система подтверждает завершение процесса регистрации.
2. Для каждого предлагаемого курса система проверяет, ведет ли его какой-либо профессор и записалось ли на него не менее трех студентов. Если эти условия выполняются, система окончательно фиксирует курс в каждом графике, который включает данный курс.
3. Система закрывает все курсы, рассчитывает плату за обучение для каждого студента в текущем семестре и направляет информацию в систему оплаты счетов. Система оплаты счетов посылает студентам счета для оплаты с копией их окончательных графиков.

Альтернативные потоки:

Регистрация не завершена:

Если при проверке завершения процесса регистрации выясняется, что регистрация еще выполняется, выдается сообщение и вариант использования завершается.

На курс записалось менее трех студентов:

Если во время выполнения основного потока обнаруживается, что на некоторый курс записалось менее трех студентов, то этот курс отменяется и выполняется подчиненный поток "Отмена курса".

Курс никто не ведет:

Если во время выполнения основного потока обнаруживается, что некоторый курс не ведется никаким профессором, то этот курс отменяется и выполняется подчиненный поток "Отмена курса".

Отмена курса:

Система отменяет предлагаемый курс. Для каждого студента, записавшегося на отмененный курс, система модифицирует его график. Первый доступный альтернативный курс подставляется вместо отмененного курса. Если альтернативных курсов нет, то подстановки не происходит и управление передается в основной поток событий для обработки следующего предлагаемого курса.

После обработки всех графиков текущего семестра система по электронной почте извещает студентов об изменениях в их графиках.

Расчетная система недоступна:

Если невозможно установить связь с расчетной системой, система вновь попытается связаться с ней через некоторое установленное время. Попытки будут повторяться до тех пор, пока связь не установится.

Предусловия:

Перед началом выполнения данного варианта использования регистратор должен войти в систему.

3.6. СОЗДАНИЕ БАЗЫ ДАННЫХ ТРЕБОВАНИЙ REQUISITEPRO И РАБОТА С НЕЙ

Прежде чем создавать типы требований, необходимо создать новый проект RequisitePro в соответствии с подразд. 1.2.2.

Упражнение 3.5.

Создание типов требований в проекте RequisitePro

Перед описанием требований необходимо определить их типы:

1. Выберите пункт меню File > Properties, появится диалоговое окно "Project Properties". Для добавления новых типов требований активизируйте вкладку "Requirement Types" (рис. 3.3).

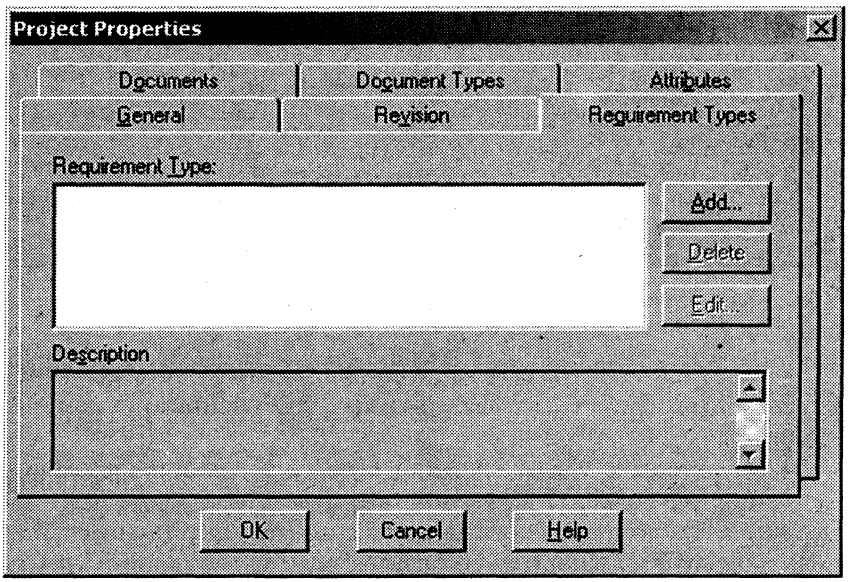


Рис. 3.3. Окно "Project Properties"

2. Для добавления нового типа требований нужно щелкнуть по кнопке "Add...", а для редактирования существующего – по кнопке "Edit...". Затем заполните следующие поля в открывшемся диалоге "Requirement Type" (рис. 3.4):

• "Name" – название типа требования (обязательное поле).
• "Description" – описание типа требования.
• "Initial Requirement #" – уникальный номер, который будет присвоен первому требованию данного типа и станет увеличиваться на единицу для всех последующих (обязательное поле).

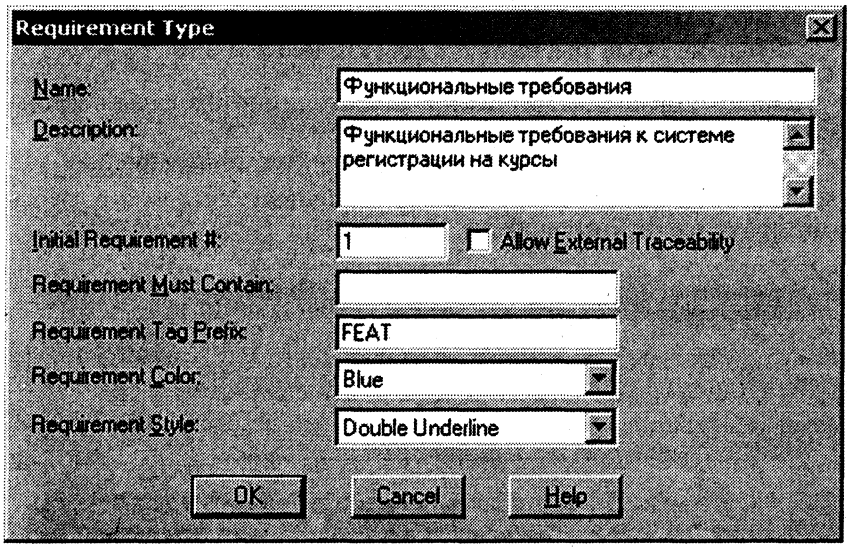


Рис. 3.4. Окно "Requirement Type"

• "Allow External Traceability" – следует активировать, если требования этого типа будут связываться с требованиями из других проектов.
• "Requirement Must Contain" – слово или простая фраза, содержащие не более 32 символов, которые обязательно должны входить в состав требований данного типа (RequisitePro будет выводить предупреждающее сообщение, если при создании требований данного типа это условие не будет выполнено).
• "Requirement Tag Prefix" – префикс, который добавляется всем требованиям данного типа (обязательное поле).
• "Requirement Color" и "Requirement Style" – характеристики форматирования требований данного типа, с помощью которых последние выделяются в документах Microsoft Word (выбрать из списка).

3. Щелкните по кнопке OK и повторите действия для следующего типа требований (рис. 3.5).

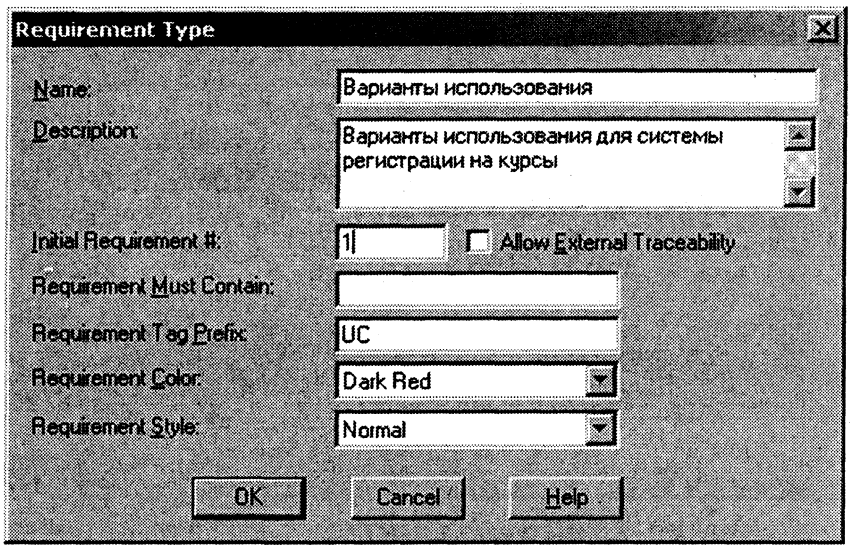


Рис. 3.5. Новый тип требований

Упражнение 3.6.
Определение атрибутов требований

Для созданных типов требований необходимо определить атрибуты, которые позволяют ввести метрики для оценки требований с точки зрения пользователя. Примеры атрибутов приведены в подразд. 1.2.1.

Некоторые атрибуты определяются только для служебных целей и создаются автоматически при выполнении определенных операций. Например, атрибуты "RoseItemId", "RoseModePath" и "RoseType" создаются при выполнении интеграции некоторой модели Rose с проектом RequisitePro.

Для добавления или изменения атрибутов нужно активизировать вкладку "Attributes" (рис. 3.6).

В левом окне группы "Requirement Attributes" перечислены основные атрибуты типа требования, выбранного в "Requirement Type". Если выделен один из этих атрибутов, то справа выводится дополнительная информация.

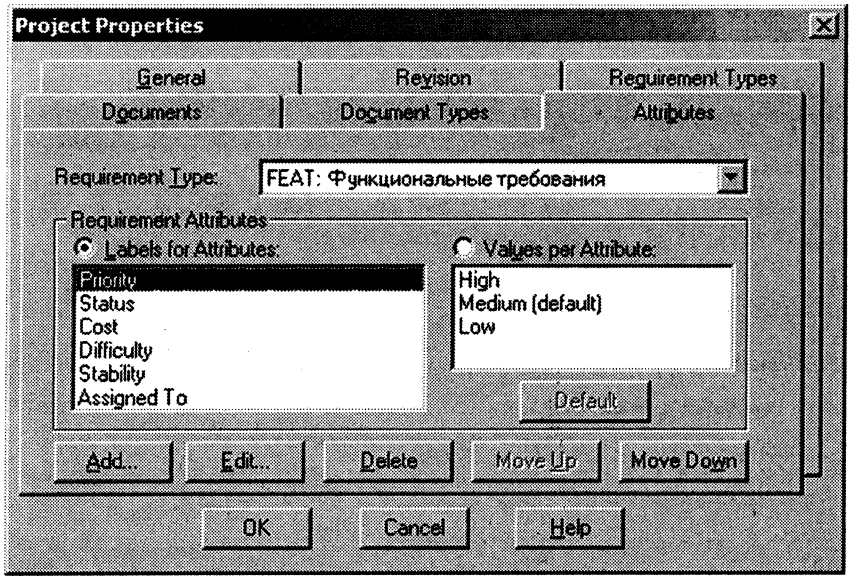


Рис. 3.6. Атрибуты требований

Активный элемент управления "Labels for Attributes" означает, что кнопки группы "Requirement Attributes" позволяют выполнять соответствующие операции над атрибутами списка (добавить атрибут – кнопка "Add...", изменить свойства атрибута – кнопка "Edit...", удалить атрибут – "Delete", передвинуть атрибут в списке – соответственно кнопки "Move Up" и "Move Down"). Щелчок по кнопке "Add..." приводит к появлению диалогового окна "Add Attribute" (рис.3.7).

Поля для заполнения:

- "Label" – название атрибута.
- "Type" – тип атрибута (например, список значений, текстовая строка, целочисленное поле, поле даты и т.д.).
- "List Values" – список возможных значений атрибута (выводится, если в "Type" выбран список).
- "Deafult Value" – значение атрибута, устанавливаемое по умолчанию для создаваемых требований (выводится, если в "Type" указан какой-либо простой тип, например, строковый).
- "Hidden from display" – флажок, установка которого позволяет скрывать данный атрибут в представлениях RequisitePro.

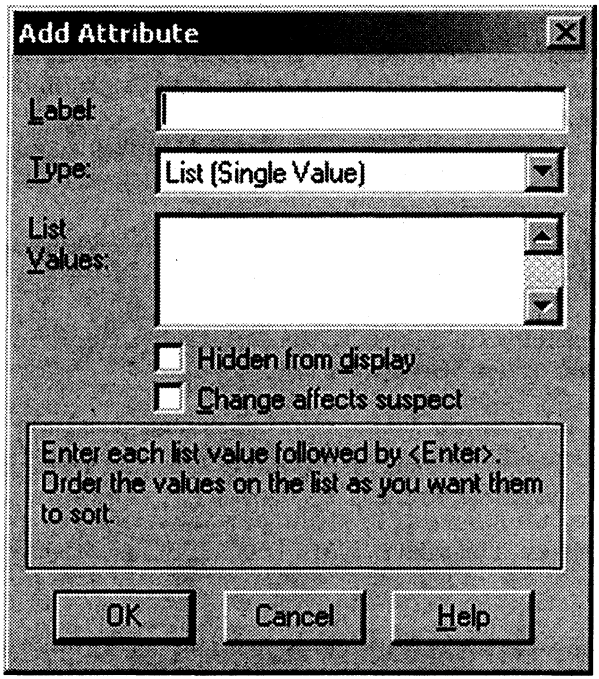


Рис. 3.7. Добавление атрибута

• "Change affect suspect" – флажок, установка которого окажет влияние на состояние трассируемых (по отношению к текущему) требований при изменении данного атрибута (в этом случае изменение атрибута приведет к установке всех трассируемых требований в состояние "подозрительных", т.е. представление RequisitePro будет приравнивать изменение атрибута к изменению самого требования).

При создании нового типа требований RequisitePro по умолчанию создает для него начальный набор атрибутов, который можно взять за основу.

В проекте системы регистрации на курсы удалим существующие атрибуты для каждого созданного типа требований и создадим следующие:

• "Приоритет" (список значений; возможные значения: "Высокий", "Средний", "Низкий") – приоритет функционального требования или варианта использования;

• "Состояние" (список значений; возможные значения: "Предложено", "Одобрено", "Реализовано", "Протестировано") – состояние, в котором находится процесс реализации требования на данный момент;

• "Сложность реализации" (список значений; возможные значения: "Высокая", "Средняя", "Низкая") – оценочная сложность реализации требования;

• "Ответственный" (строка) – указывается фамилия ответственного за реализацию требования.

Упражнение 3.7.

Создание типов документов

Каждый документ RequisitePro должен иметь свой тип, который определяет основное назначение этого документа (область использования в проекте).

Для создания нового типа документов необходимо активизировать вкладку "Document Types" (рис. 3.8) и щелкнуть по кнопке "Add...".

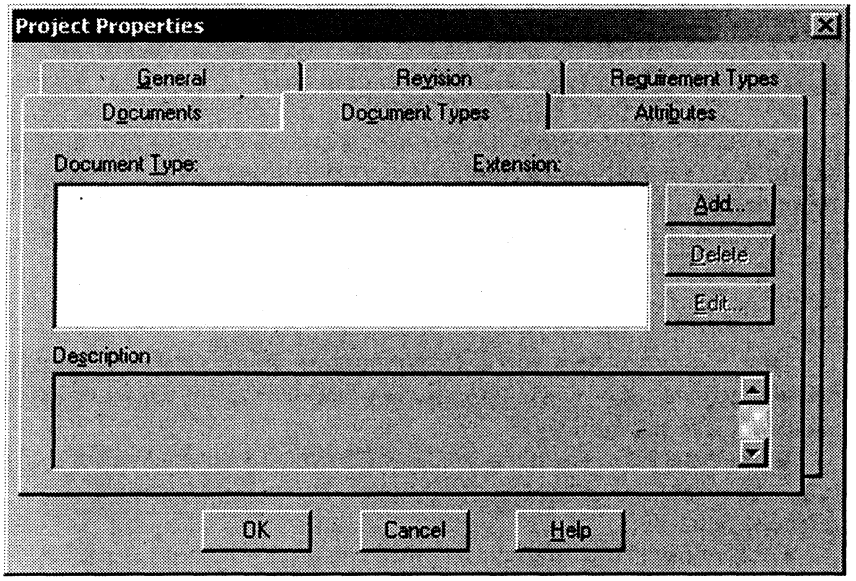
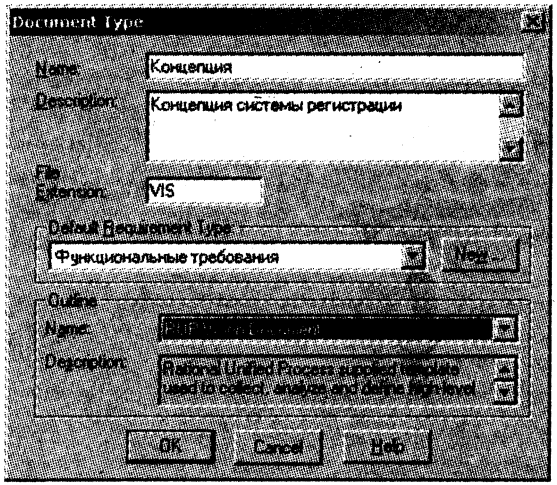


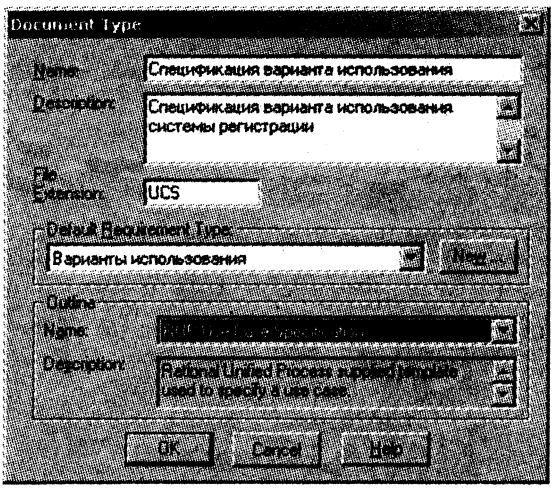
Рис. 3.8. Типы документов

Откроется диалоговое окно "Document Type" (рис. 3.9). Поля для заполнения:

- "Name" – название типа документа (обязательное поле);



а



б

Рис. 3.9. Окно "Document Type":

а – общий вид; *б* – для спецификации варианта использования

• "Description" – описание типа документа;
• "File Extension" – расширение, с которым будут сохраняться файлы документов данного типа (обязательное поле);
• "Default Requirement Type" – тип требования по умолчанию (обязательное поле);
• "Outline" – указание на шаблон Microsoft Word, который будет использоваться при создании новых документов данного типа.

Для проекта системы регистрации создадим два типа документов: концепция (см. рис. 3.9) и спецификация варианта использования (рис. 3.10).

Упражнение 3.8.

Связывание модели Rose и проекта RequisitePro

Для использования модели Rose совместно с RequisitePro необходимо, чтобы в Rose было активизировано соответствующее встроенное средство (Add-In). Для этого нужно выбрать пункт меню "Add-Ins/Add-In Manager...", что приведет к появлению окна "Add-In Manager", и проконтролировать, чтобы пункт "RequisitePro" (рис. 3.10) был активен.

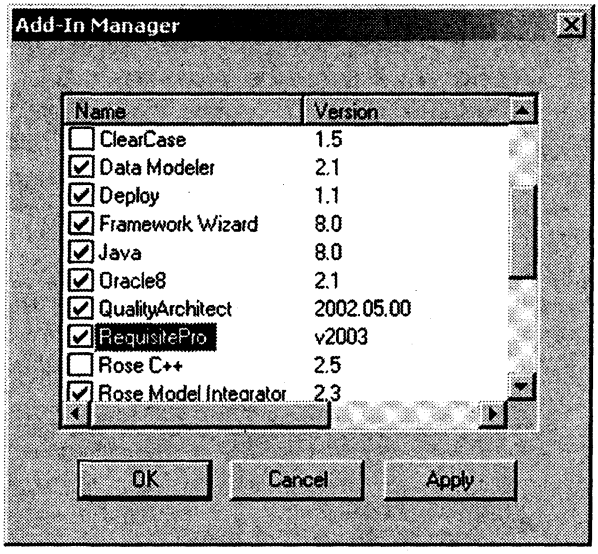


Рис. 3.10. Окно "Add-In Manager"

В результате появятся дополнительные пункты в главном и различных контекстных меню, позволяющие работать с RequisitePro из среды Rose. Затем нужно связать текущий файл модели Rose с проектом RequisitePro с помощью пункта меню "Tools/Rational RequisitePro/Associate Model To Project". В открывшемся окне "Associate Model To RequisitePro Project" следует указать проект RequisitePro, в который будут экспортированы варианты использования (рис. 3.11), а также типы требований и документов по умолчанию, с которыми будут автоматически связаны экспортированные варианты использования. В дальнейшем указанные типы могут быть заменены на любые другие с помощью этого окна.

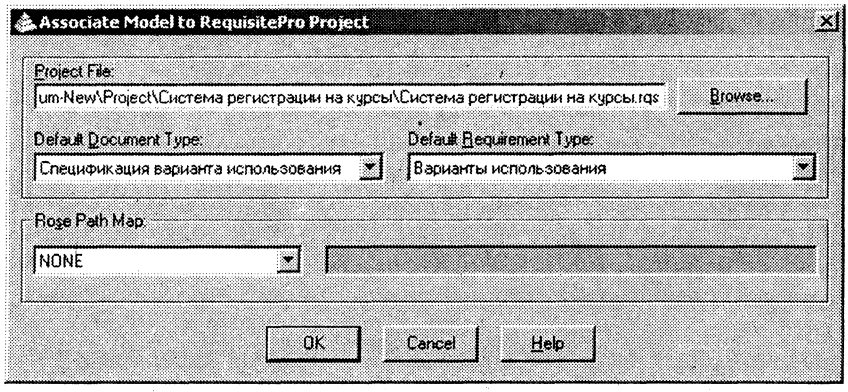


Рис. 3.11. Окно "Associate Model To RequisitePro Project"

Щелчок по кнопке OK приведет к связыванию модели Rose и проекта RequisitePro.

Упражнение 3.9.

Экспорт вариантов использования из модели Rose в проект RequisitePro

Рассмотрим экспорт вариантов использования на примере системы регистрации. Для экспорта варианта использования следует выбрать пункт его контекстного меню "Requirement Properties/New...". При этом появится форма добавления требований "Requirement Properties..." из RequisitePro (рис. 3.12).

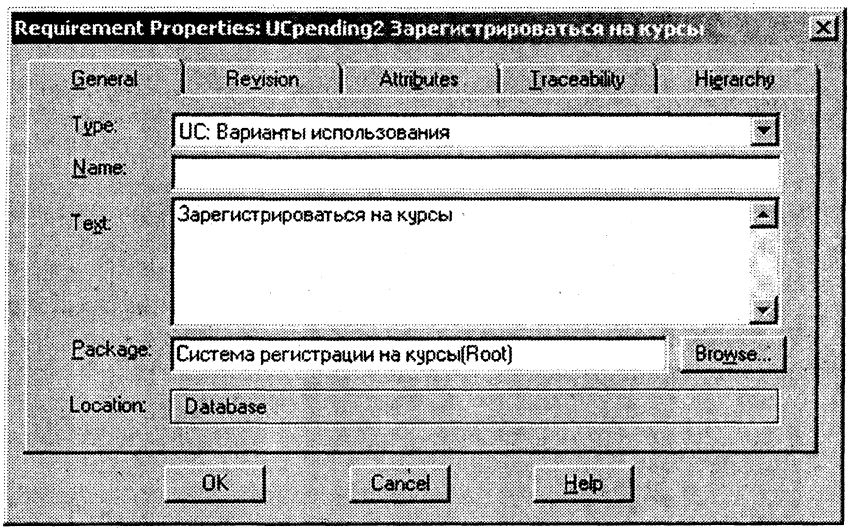


Рис. 3.12. Форма добавления требований

Данная форма позволяет установить значения атрибутов вариантов использования (вкладка "Attributes", можно выполнить эту установку самостоятельно), создать связи с существующими требованиями любых типов (вкладка "Traceability") и сформировать иерархию вариантов использования (вкладка "Hierarchy"). Щелчок по кнопке OK приведет к созданию требования типа "Варианты использования" в базе данных RequisitePro.

Таким образом, можно экспортировать все остальные варианты использования в проект RequisitePro:

- Войти в систему;
- Просмотреть табель успеваемости;
- Выбрать курсы для преподавания;
- Проставить оценки;
- Вести информацию о профессорах;
- Вести информацию о студентах;
- Закрыть регистрацию.

Требования, созданные в RequisitePro, следует перенести в пакет, предназначенный для их хранения (создать новые пакеты "Документы", "Требования" и "Варианты использования" (рис. 3.13) с помощью контекстного меню проекта в Проводнике RequisitePro, используя пункты New > Package).

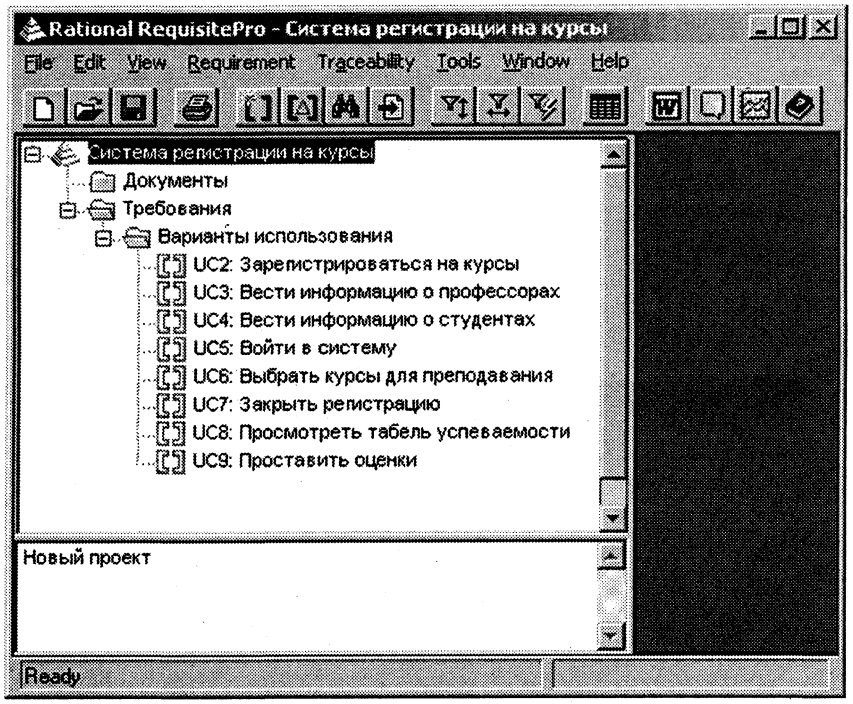


Рис. 3.13. Структура проекта после экспорта всех вариантов использования

Упражнение 3.10.

Создание представлений в проекте RequisitePro

Представления (views) RequisitePro отображают в табличном или древовидном виде сами требования с их атрибутами или связи трассировки между требованиями различных типов. Можно создавать представления трех видов:

- матрица атрибутов (Attribute Matrix). Она показывает все требования заданного типа с их атрибутами;
- матрица трассировки (Traceability Matrix). Она показывает связи трассировки между требованиями двух типов;
- дерево трассировки (Traceability Tree). Оно показывает цепочки связей трассировки, направленные от требований заданного типа или к ним.

С помощью представлений можно выполнить следующие действия:

- создать и модифицировать наименование, текст и атрибуты требования, а также его связи трассировки;
- упорядочить и отфильтровать информацию;
- сохранить представление и вывести его на печать.

Для создания матрицы атрибутов для вариантов использования создадим в Проводнике RequisitePro пакет "Матрицы атрибутов", выберем в его контекстном меню пункт New > View..., после чего появится окно "View Properties". Зададим в окне значения, как показано на рис. 3.14.

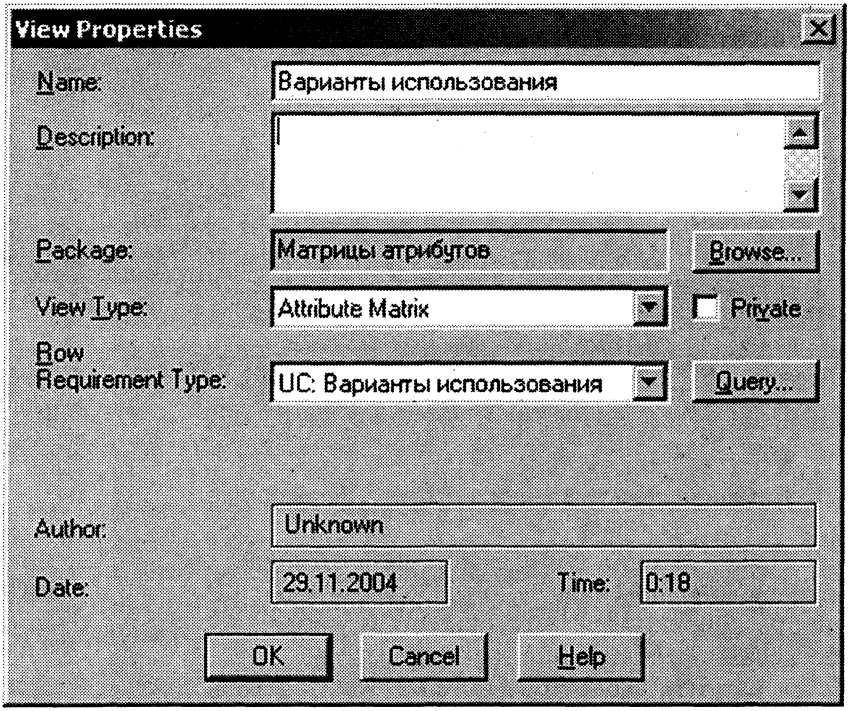


Рис. 3.14. Окно "View Properties"

После щелчка по кнопке OK на экране появится окно матрицы атрибутов для вариантов использования (рис. 3.15). С помощью контекстного меню для каждой клетки матрицы установите значения атрибутов требований, как показано на данном рисунке.

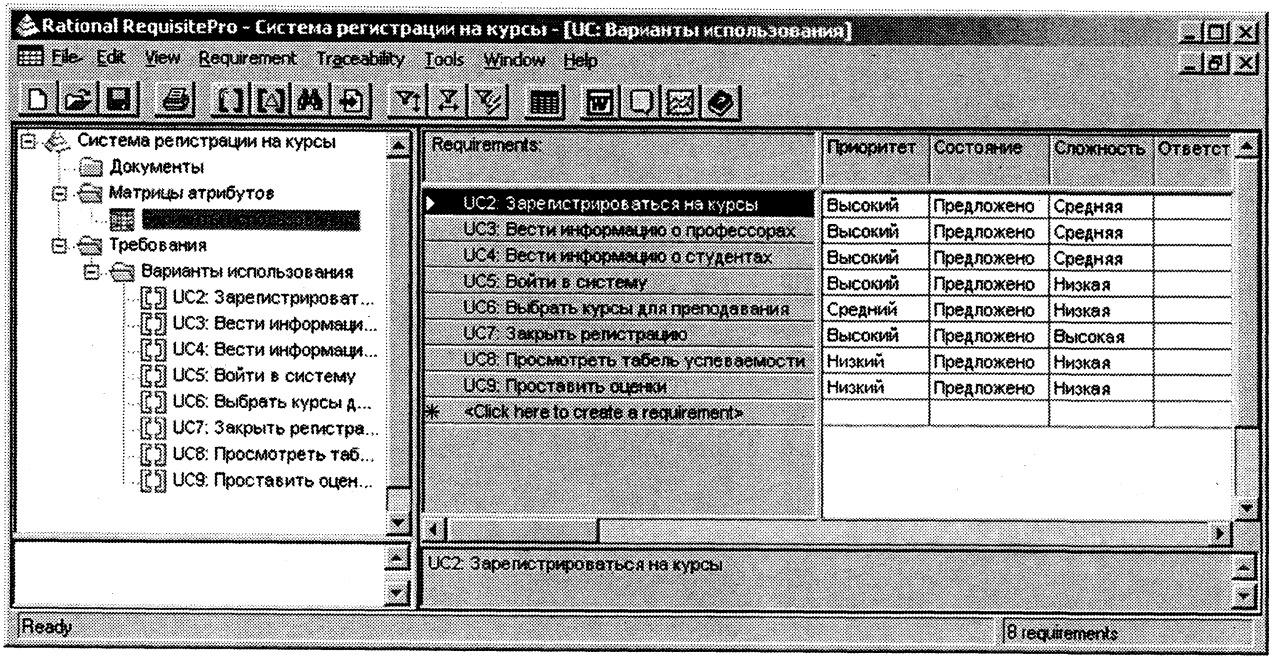


Рис. 3.15. Матрица атрибутов для вариантов использования

Упражнение 3.11.
Добавление документов в проект RequisitePro

Множество документов проекта может содержать самые разнообразные требования к создаваемому продукту или к процессу создания этого продукта. Такие документы должны быть созданы или импортированы в проект RequisitePro. Далее в них будут созданы необходимые требования.

Создать документ в RequisitePro можно, импортировав готовый документ с помощью пункта меню "File > Import...". В появившемся окне "Import Wizard" (рис. 3.16) укажите в качестве источника Microsoft Word Document, в поле Name of the document to import укажите (с помощью Browse) файл Зарегистрироваться на курсы.doc, созданный в упражнении 3.4, затем щелкните по кнопке Next.

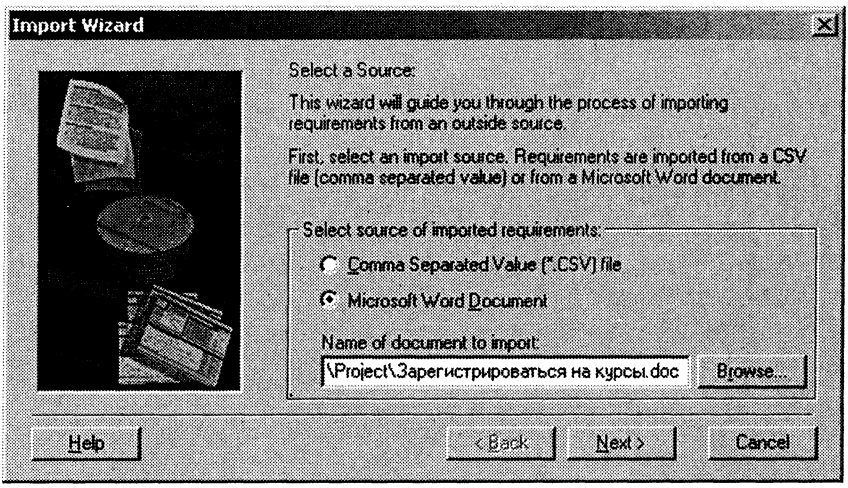


Рис. 3.16. Окно "Import Wizard"

В окне выбора содержимого укажите импорт Document only (рис. 3.17), затем щелкните по кнопке Next.

В окне "Document Properties" введите описание документа (рис. 3.18) и щелкните по кнопке OK.

В сообщении Create Document появится запрос на переформатирование документа в соответствии с определенным ранее для такого типа документов шаблоном RUP Use Case Specification, щелкните по кнопке Yes, чтобы сохранить текущее форматирование документа.

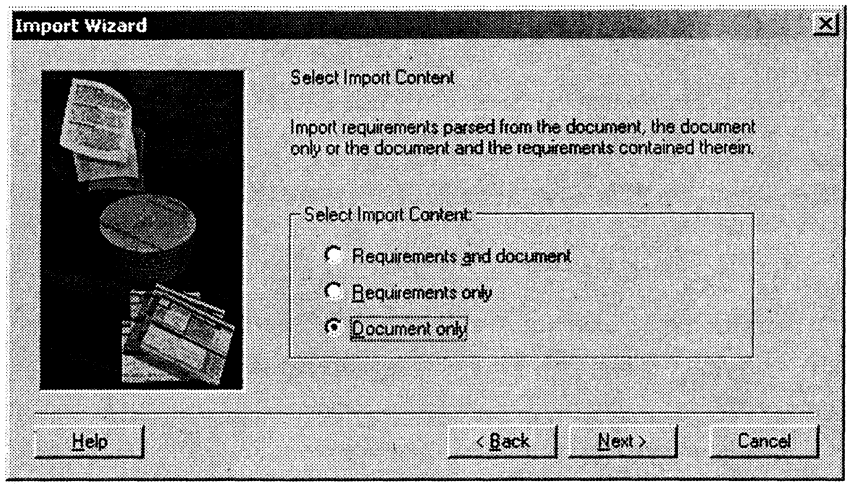


Рис. 3.17. Окно выбора содержимого

Рис. 3.18. Окно "Document Properties"

Упражнение 3.12.

Создание иерархии требований в документе

Заполнив новый документ, можно начать создавать требования в нем. Для этого нужно выделить необходимый фрагмент текста, который представляет очередное требование, и с помощью главного или контекстного меню Word активировать пункт "New Requirement".

В открывшемся диалоговом окне "Requirement Properties" указываются следующие данные:

- на вкладке "General" – название требования (некоторое имя, которое позволит выделить данное требование среди других в каком-либо представлении RequisitePro) и его тип;
- на вкладке "Attributes" – конкретные значения атрибутов;
- на вкладке "Traceability" – требования, с которыми должно быть связано данное требование;
- на вкладке "Hierarchy" – родительское требование по отношению к данному (для создания иерархии требований);
- на вкладке "Discussions" – темы для обсуждения текущего требования между участниками проекта.

Создадим требование для фрагмента текста с названием варианта использования "Зарегистрироваться на курсы" (рис. 3.19).

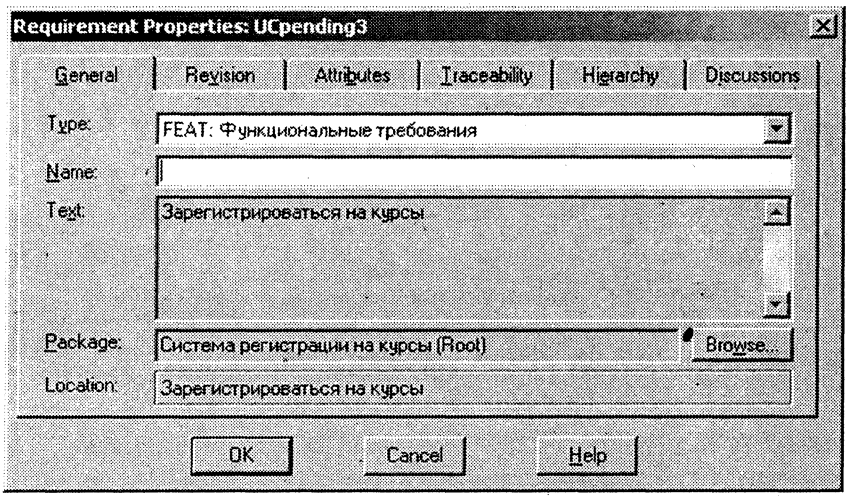


Рис. 3.19. Окно "Requirement Properties" для требования "Зарегистрироваться на курсы"

Далее создадим требование для фрагмента текста с названием подчиненного потока событий "Создать график" (рис. 3.20). Используя вкладку "Hierarchy", выберем для него в качестве родительского требования "Зарегистрироваться на курсы" (рис. 3.21 и 3.22).

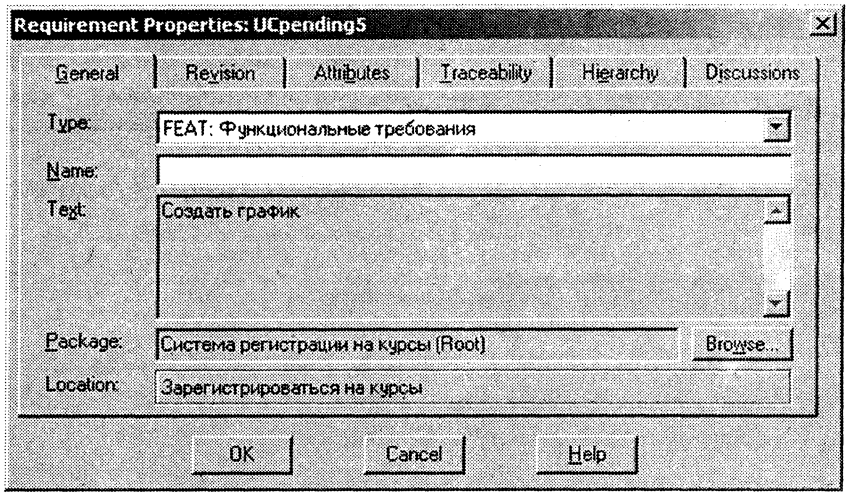


Рис. 3.20. Окно "Requirement Properties" для требования "Создать график"

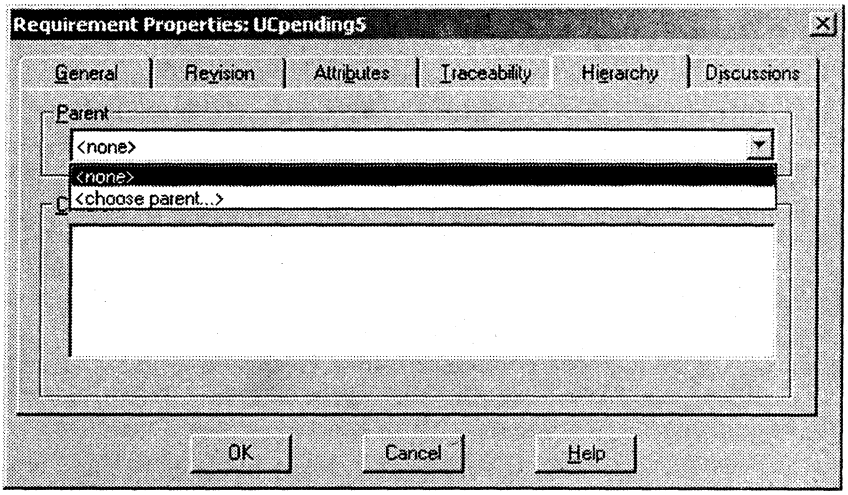


Рис. 3.21. Вкладка "Hierarchy"

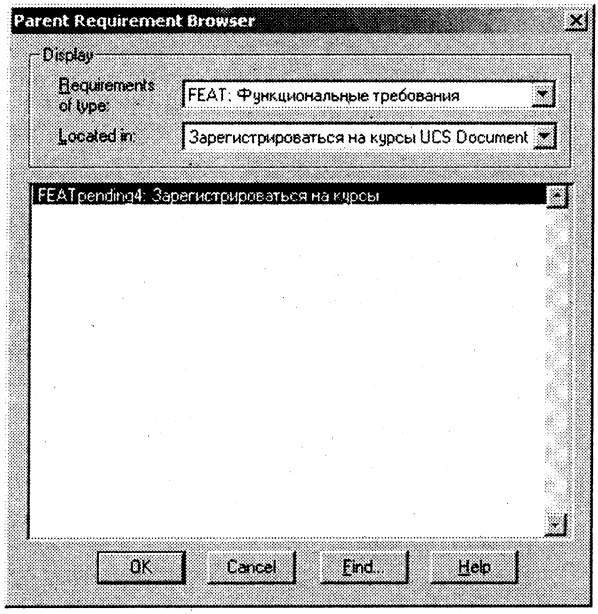


Рис. 3.22. Выбор родительского требования

Выполните те же самые действия для подчиненных потоков событий "Обновить график", "Удалить график", "Принять график" и "Сохранить график". Кроме того, создайте требование для фрагмента текста "поиск в каталоге курсов" из подчиненного потока "Создать график", выбрав для него в качестве родительского требования "Создать график".

В результате импортированный документ примет вид, показанный на рис. 3.23. Выберите в меню пункт RequisitePro > Document > Save, обратите внимание, что метки "pending" исчезли. После сохранения документа требования будут перенесены в базу данных проекта и появятся в окне Проводника. Выберите в меню пункт RequisitePro > Document > Close.

Создадим в окне Проводника RequisitePro новый пакет "Функциональные требования" и перенесем туда вновь созданные требования. Создадим новую матрицу атрибутов для функциональных требований. После ее открытия интерфейс RequisitePro должен принять вид, показанный на рис. 3.24.

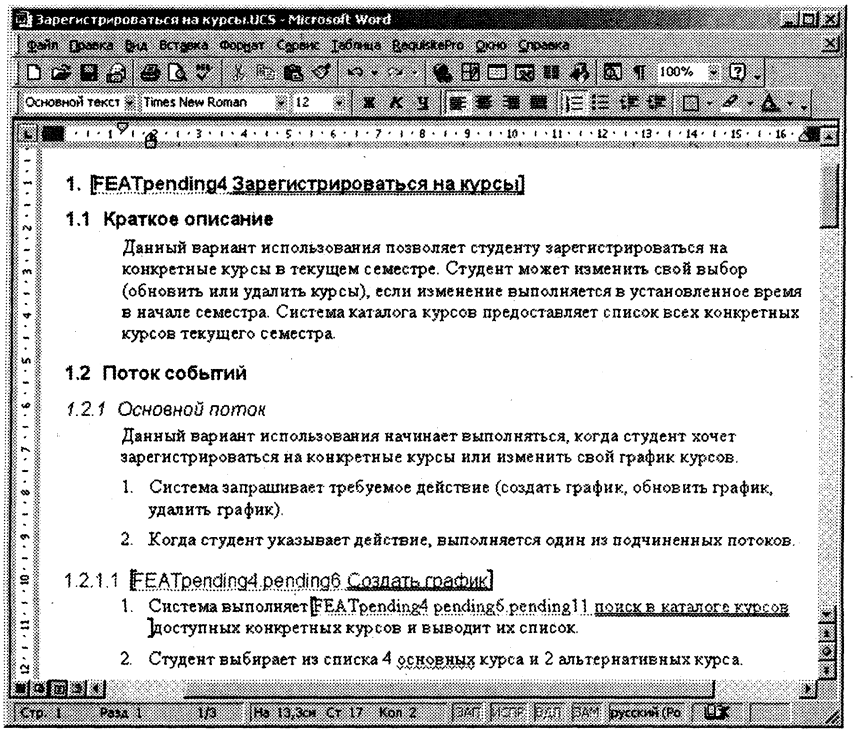


Рис. 3.23. Вид документа после создания требований

Упражнение 3.13.

Связывание вариантов использования Rose с документами RequisitePro

После экспорта вариантов использования из Rose в RequisitePro можно связать каждый из них со своим документом (спецификацией варианта использования), предварительно импортированным в проект RequisitePro. Выполним такое связывание для варианта использования "Зарегистрироваться на курсы" (рис. 3.25). Для этого в его контекстном меню нужно выбрать пункт Use Case Document > Associate.

В появившемся окне "Associate Document to Use Case" (рис. 3.26) выберите из списка нужный документ из ранее импор-

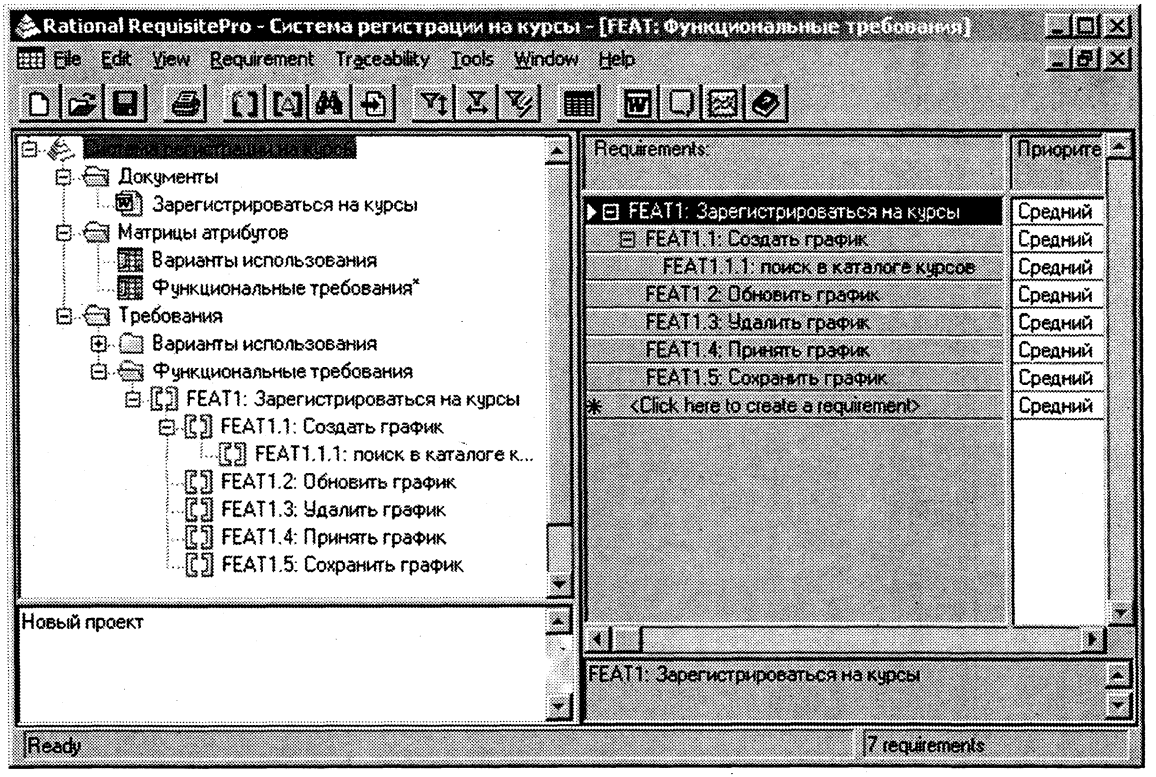


Рис. 3.24. Матрица атрибутов для функциональных требований

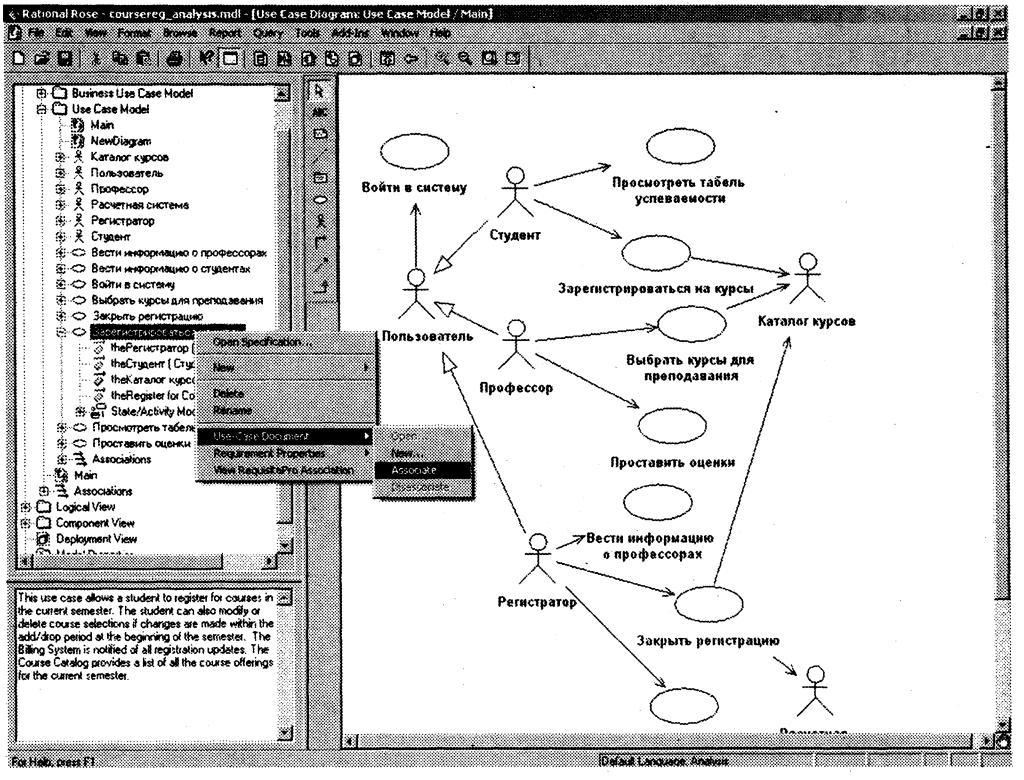


Рис. 3.25. Связывание варианта использования с документом

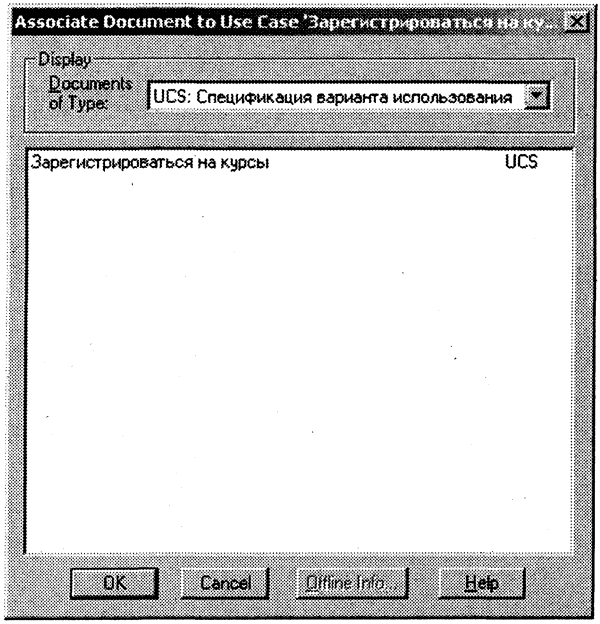


Рис. 3.26. Окно "Associate Document to Use Case"

тированных документов (в данном случае в списке присутствует только один документ).

В результате RequisitePro загрузит в окне "Microsoft Word" нужный документ (рис. 3.27), а в браузере Rational Rose под вариантом использования "Зарегистрироваться на курсы" появится ссылка на этот документ, с помощью которой его впоследствии можно будет загружать.

Упражнение 3.14.

Импорт документов и требований в проект RequisitePro

Требования могут быть созданы автоматически путем импорта данных из внешнего источника или из других документов, хранящихся в других проектах.

Создадим новые нефункциональные требования в проекте путем импорта данных из документа "Дополнительные спецификации". Предварительно следует создать в проекте новый тип документа с названием "Дополнительные спецификации" и новый тип требований с названием "Нефункциональные требования".

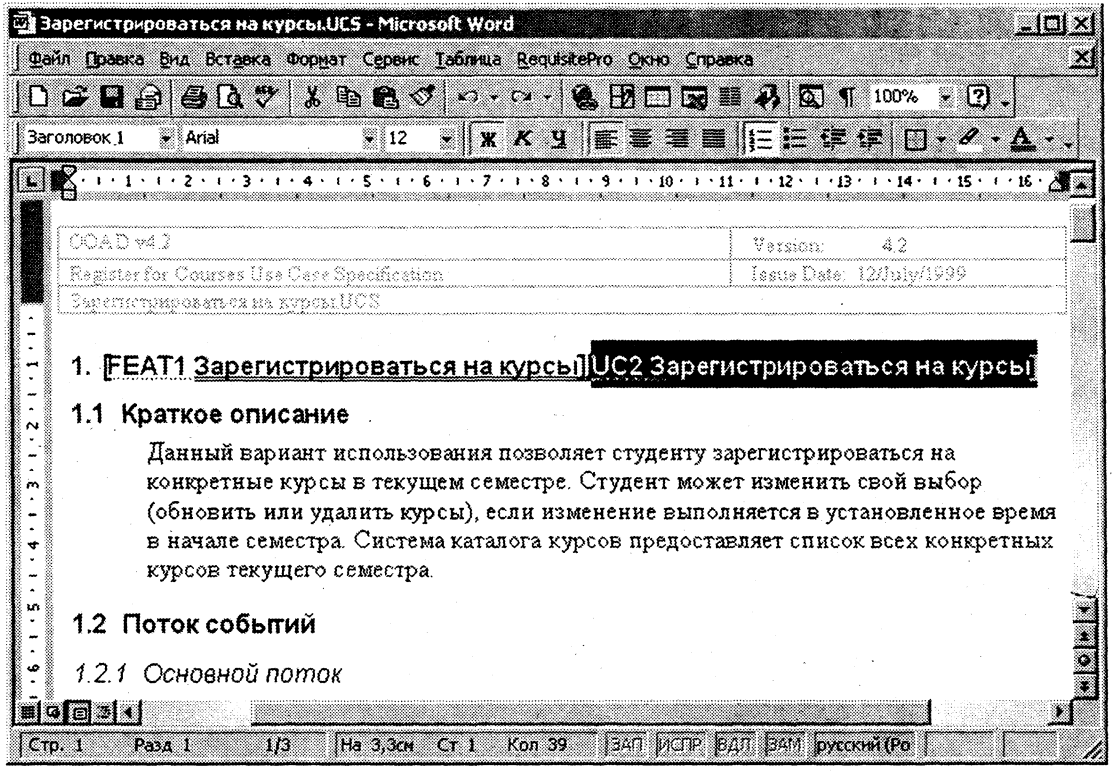


Рис. 3.27. Спецификация варианта использования после связывания

Импортируйте готовый документ аналогично упражнению 3.11 с помощью пункта меню "File > Import...". В окне "Import Wizard" укажите в качестве источника Microsoft Word Document, в поле Name of the document to import укажите файл Дополнительные спецификации.doc, который должен быть создан с использованием описания в подразд. 3.3, затем щелкните по кнопке Next. В окне выбора содержимого укажите импорт Requirements and document, затем щелкните по кнопке Next (рис. 3.28).

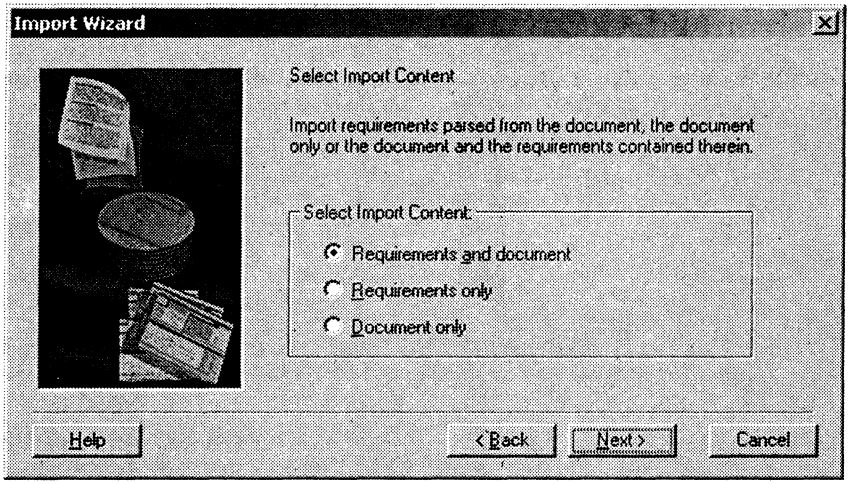


Рис. 3.28. Окно выбора содержимого

В окне "Document Properties" введите описание документа (рис. 3.29) и щелкните по кнопке OK.

В новом окне "Import Wizard" (рис. 3.30) в списке Requirement Type выберите "Нефункциональные требования", активируйте опции Keyword(s) и Sentence, в поле Keyword укажите "должна", затем щелкните по кнопке "Add...", укажите "должен", еще раз щелкните по кнопкам "Add..." и Next.

Таким образом формируется указание для анализатора импорта по распознаванию требований в импортируемом документе (каждое требование сопровождается ключевым словом "должна" или "должен"). В результате RequisitePro выполнит поиск в документе и создаст требования в соответствии с указанными критериями. Каждое требование появляется отдельно в окне

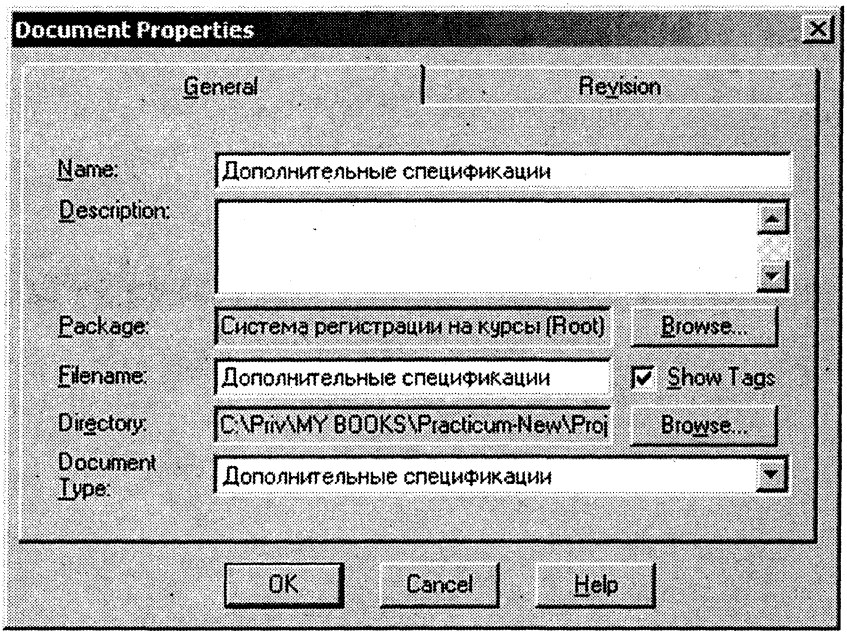


Рис. 3.29. Окно "Document Properties"

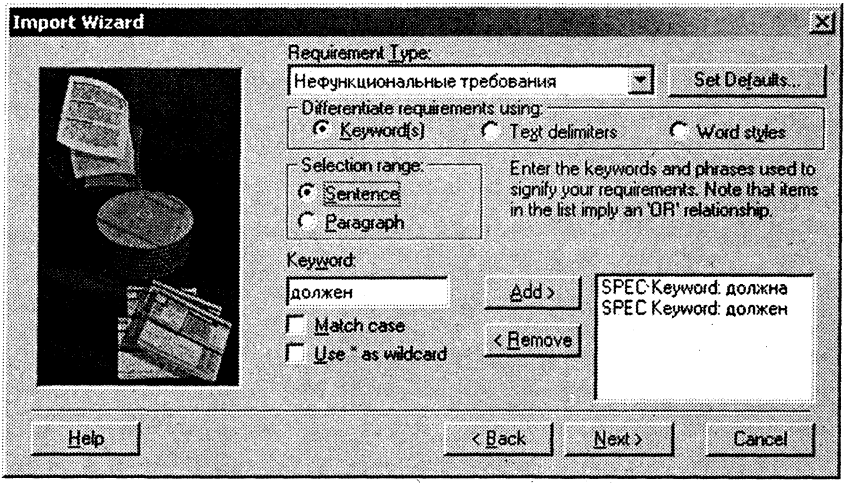


Рис. 3.30. Формирование условия для распознавания требований

(рис. 3.31), для его принятия щелкните по кнопке Yes (или по кнопке Yes to All для принятия всех требований полностью).

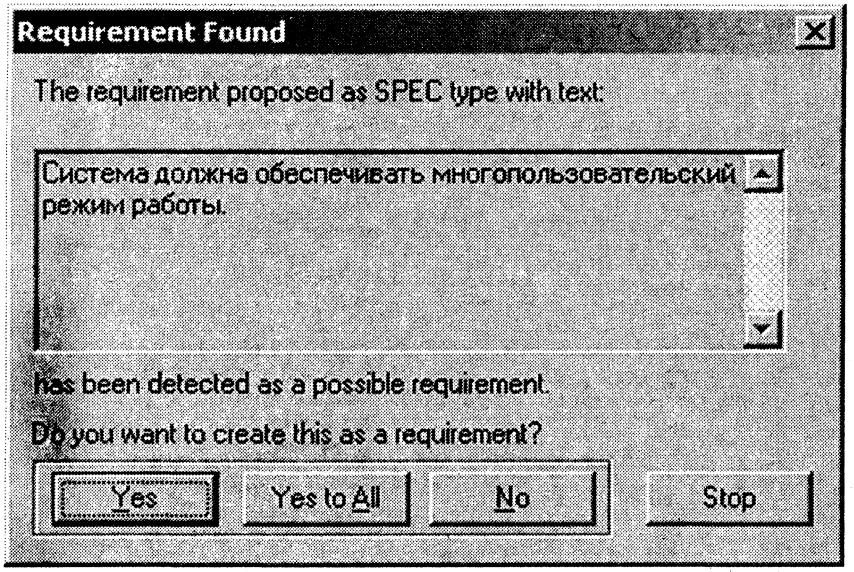


Рис. 3.31. Окно принятия требования

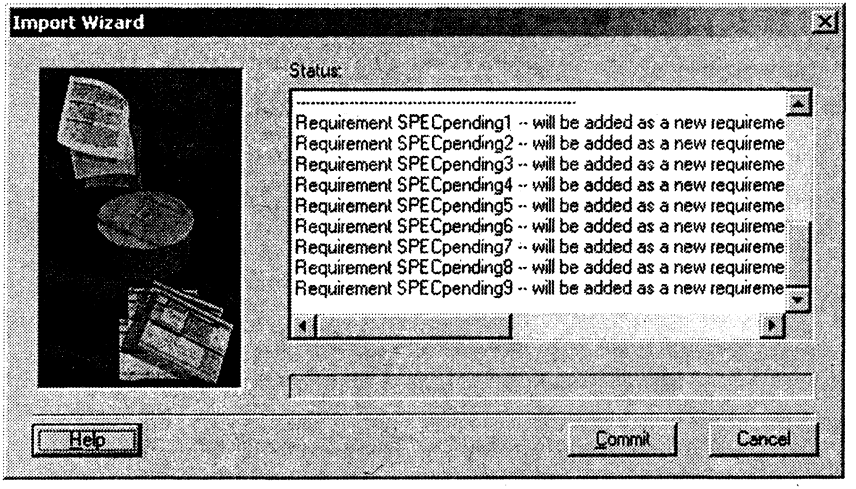


Рис. 3.32. Сохранение требований в базе данных проекта

В последнем окне "Import Wizard" (рис. 3.32) щелкните по кнопке Commit для сохранения требований в базе данных проекта.

Просмотрите документ "Дополнительные спецификации" на предмет выявления ошибочно помеченных или пропущенных требований и закройте его.

Создайте новый пакет "Нефункциональные требования" и соответствующую матрицу атрибутов, перенесите вновь созданные требования в этот пакет (рис. 3.33).

Попробуйте самостоятельно извлечь функциональные требования из документа "Концепция".

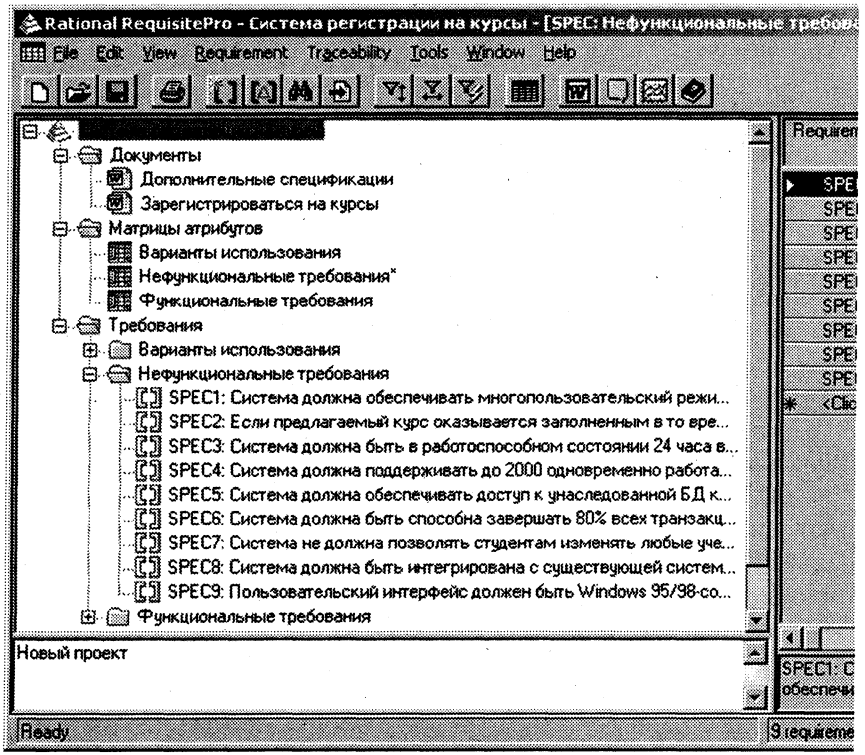


Рис. 3.33. Вновь созданные нефункциональные требования

Упражнение 3.15.
Создание новых требований в базе данных проекта

Требования могут не только импортироваться из внешнего источника или создаваться в документе, но и непосредственно помещаться в базу данных.

Выберите матрицу атрибутов "Нефункциональные требования" и нажмите строку создания нового требования с надписью "Click here to create a requirement". В открывшемся окне введите наименование требования "Система должна быть написана на языке Java".

Щелкните левой кнопкой мыши вне поля текста. Требование будет введено в базу данных проекта.

Добавьте еще несколько требований по своему усмотрению. Выберите одно из вновь созданных требований и отредактируйте его атрибуты.

Упражнение 3.16.
Трассировка требований

Трассировка требований определяет логическую связь между двумя различными требованиями. При добавлении нового требования необходимо оценить, изменение каких требований может привести к изменению добавляемого (или, говоря иначе, какие другие требования должны быть проверены на предмет возможных изменений). Если такие требования выявлены, нужно установить трассировку по отношению к ним со стороны текущего.

Матрица трассировки (Traceability Matrix) позволяет устанавливать такую трассировку, а также диагностировать, какие требования могут измениться. При создании подобного представления необходимо указать типы требований, между требованиями которых установлена трассировка (рис. 3.34). Можно трассировать не только требования одного или разных типов, но и требования, расположенные в разных проектах RequisitePro. Для этого в свойствах данного типа требований должен быть установлен переключатель "Allow External Traceability".

После создания и открытия матрица трассировки примет вид, показанный на рис. 3.35.

Выберите пустую ячейку матрицы (рис. 3.36), щелкните правой кнопкой мыши и в появившемся контекстном меню выбери-

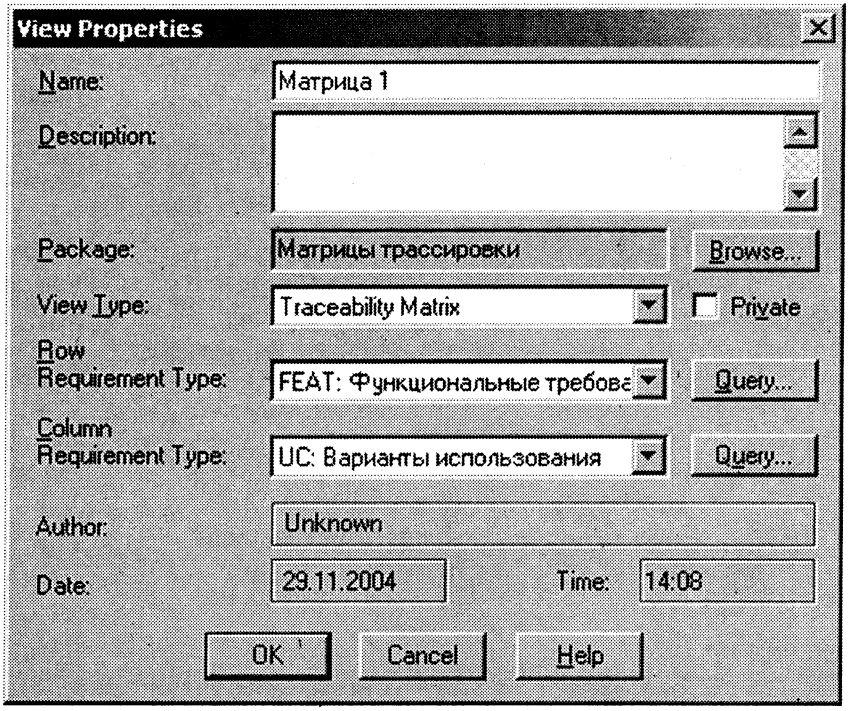


Рис. 3.34. Свойства матрицы трассировки

те Trace To. При этом установится связь трассировки между указанными требованиями, с помощью которой можно определить, какие функциональные требования необходимо просмотреть на предмет возможных изменений, если изменились любые варианты использования.

Теперь изменим в модели Rose наименование варианта использования "Зарегистрироваться на курсы" на "Записаться на курсы". Для синхронизации с вариантом использования в RequisitePro выберем в модели Rose пункт контекстного меню измененного варианта использования Requirement Properties > Open... (рис. 3.37).

После подтверждения в открывшемся окне "Requirement Properties" в проекте RequisitePro требование "Зарегистрироваться на курсы" из группы "Варианты использования" будет также

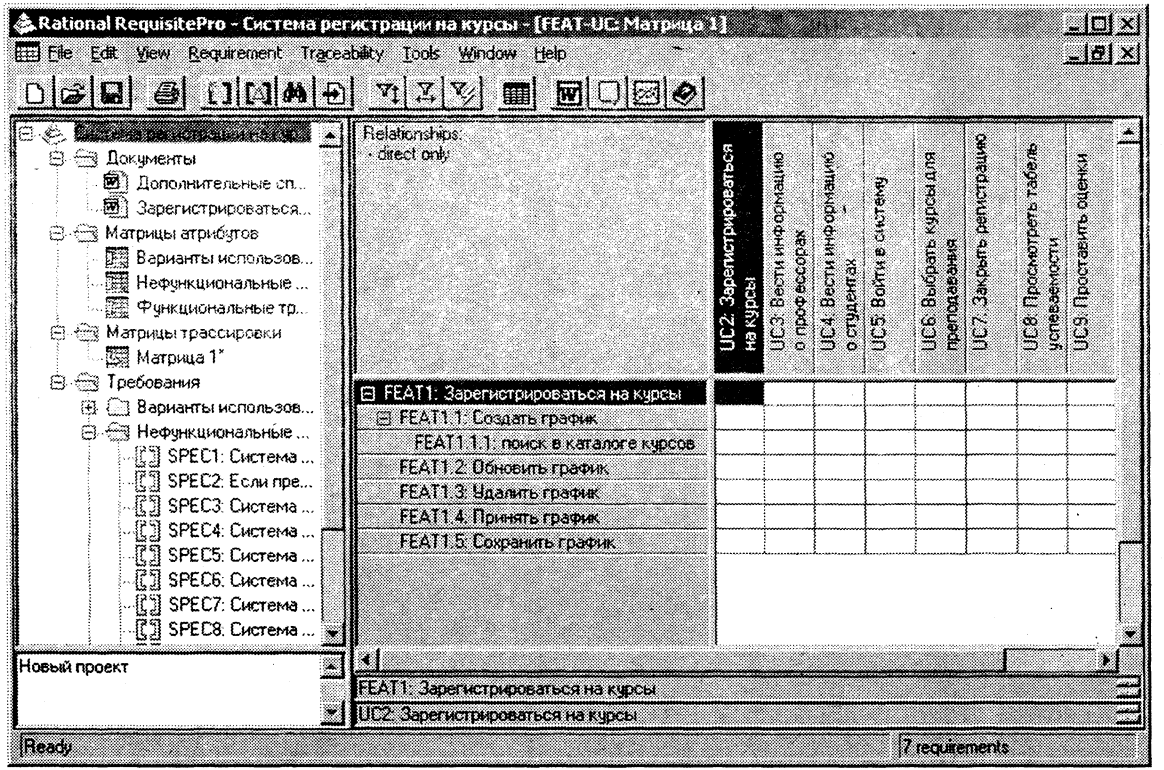


Рис. 3.35. Матрица трассировки

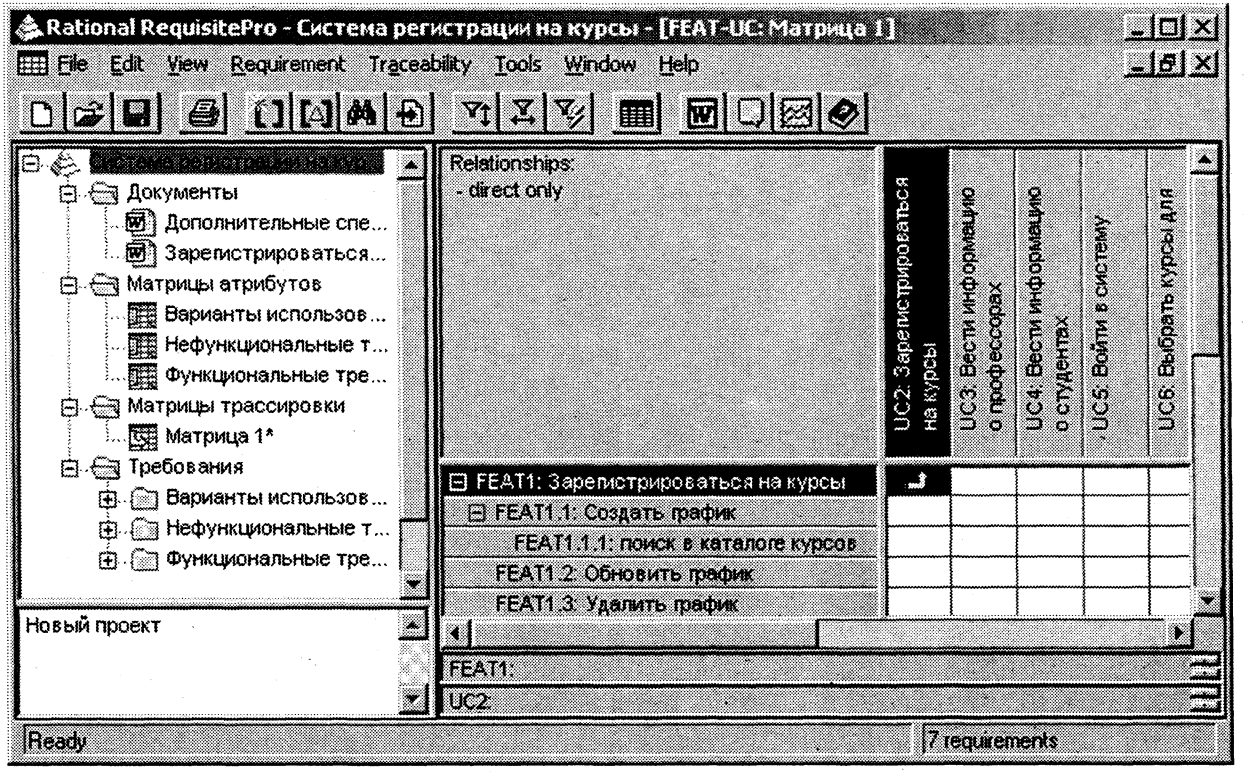


Рис. 3.36. Установка связи трассировки

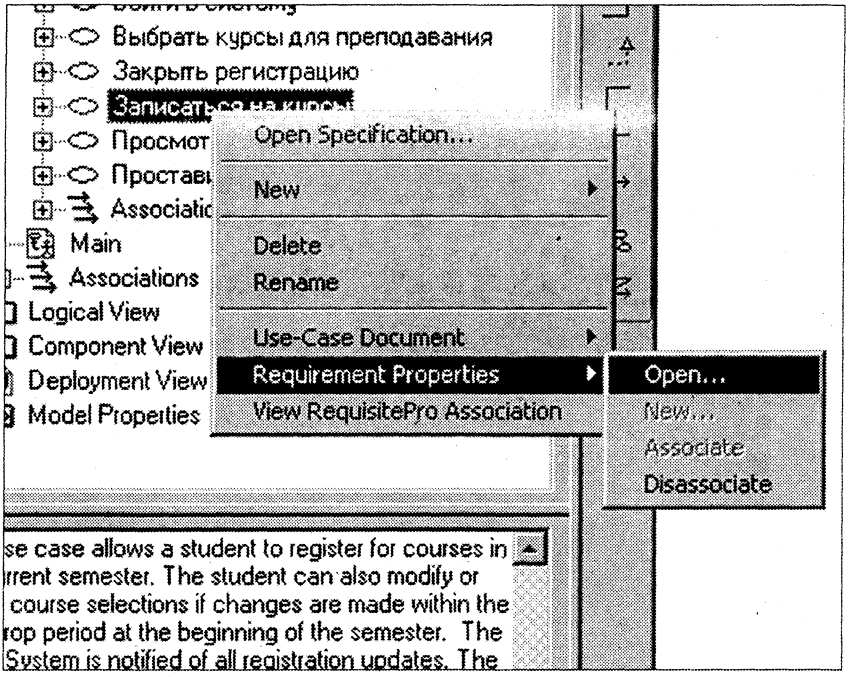


Рис. 3.37. Изменение наименования варианта использования

изменено (рис. 3.38). Связь в матрице трассировки будет помечена красной чертой как "подозрительная" (suspect).

Перечеркнутый значок трассировки показывает, что необходимо просмотреть функциональное требование "Зарегистрироваться на курсы" и все его дочерние требования. Очевидно, что после изменения названия варианта использования нужно как минимум изменить формулировку родительского требования на "Записаться на курсы".

После внесения изменений в функциональные требования нужно вручную убрать значок перечеркивания, выбрав пункт контекстного меню Clear Suspect на перечеркнутом поле. Каждая "подозрительная" связь должна быть проверена, затем признак "подозрительности" снимается вручную.

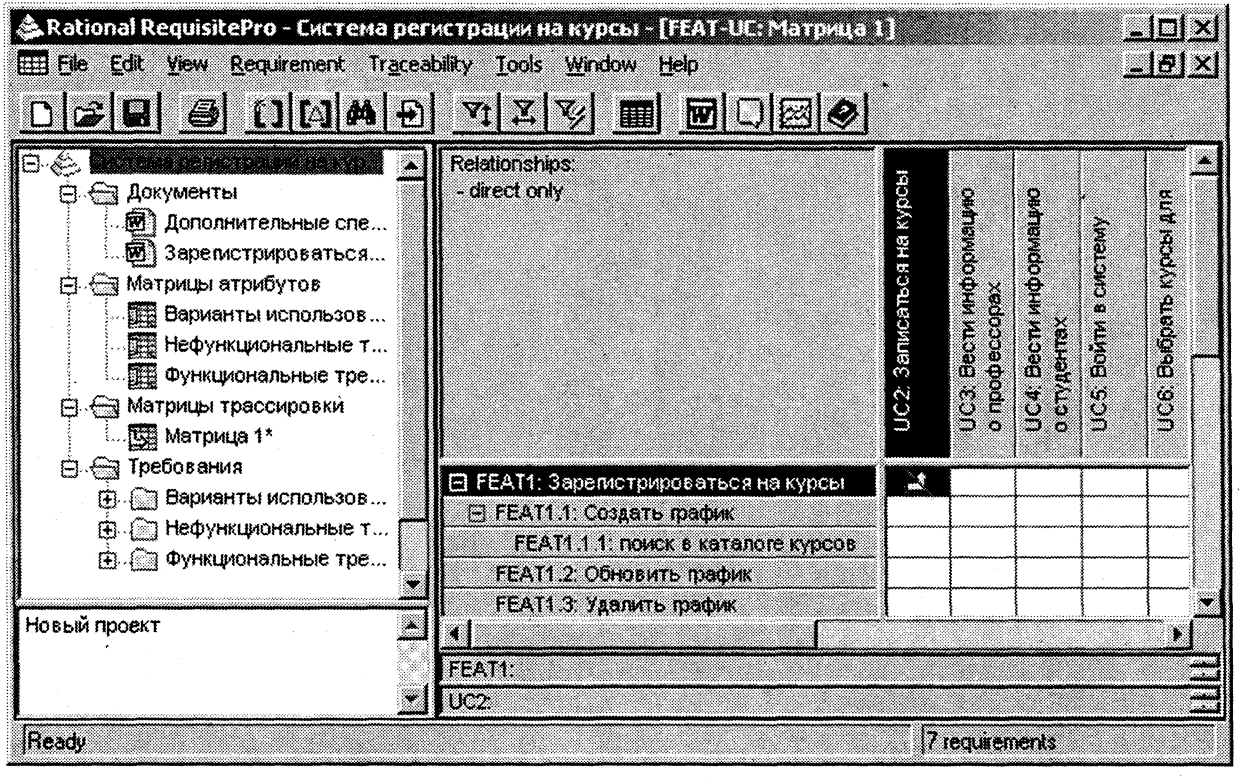


Рис. 3.38. Появление "подозрительной" связи

Упражнение 3.17.

Выполнение запросов (фильтрация и упорядочение требований)

После создания представления (матрицы атрибутов) можно выполнять любые запросы к базе данных требований, выбирая и упорядочивая требования по заданным критериям. Например, показать только те варианты использования, которые имеют высокий приоритет и среднюю сложность реализации.

Для создания запроса откройте матрицу атрибутов вариантов использования, затем выберите в меню пункт View > Query Row Requirements. В результате откроется окно "Select Attribute" (рис. 3.39).

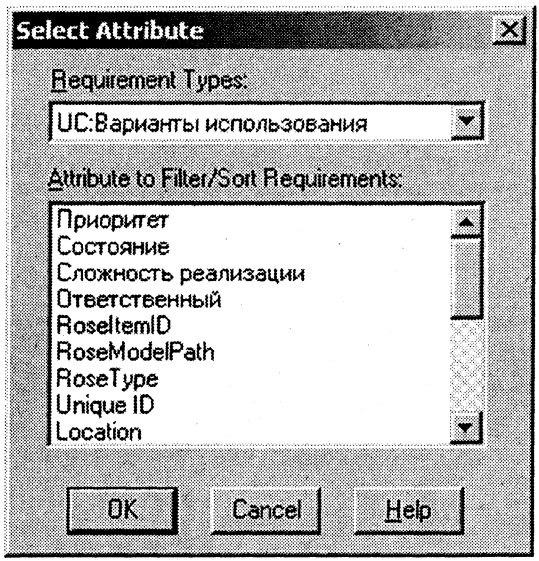


Рис. 3.39. Окно "Select Attribute"

Выберите атрибут "Приоритет" и щелкните по кнопке OK. Появится окно "Query Requirements". Щелкните по кнопке None для очистки значений атрибута, затем выберите значение "Высокий" (рис. 3.40) и щелкните по кнопке OK.

В окне "Query Row Requirements" появится критерий запроса. Щелкните по кнопке "Add..." и задайте аналогичным образом фильтр для атрибута "Сложность реализации", указав значение

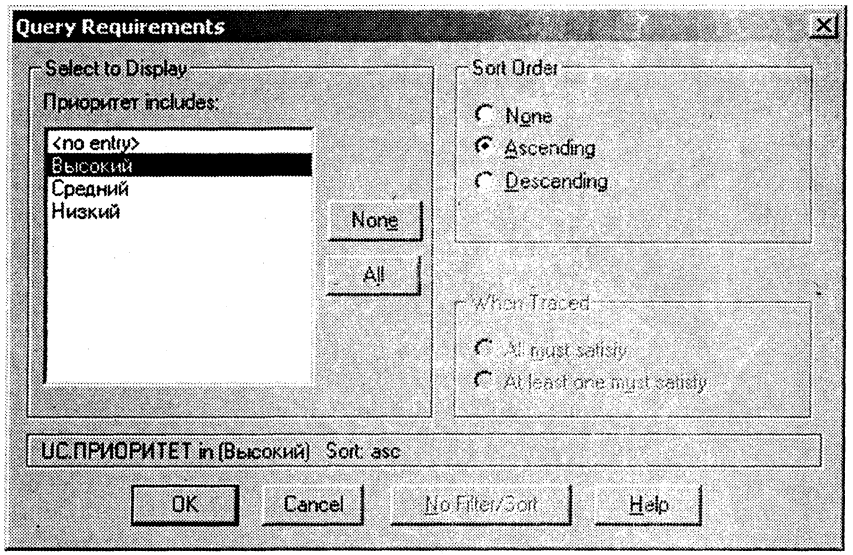


Рис. 3.40. Окно "Query Requirements"

"Средняя" (рис. 3.41). Щелкните по кнопке OK для выполнения запроса.

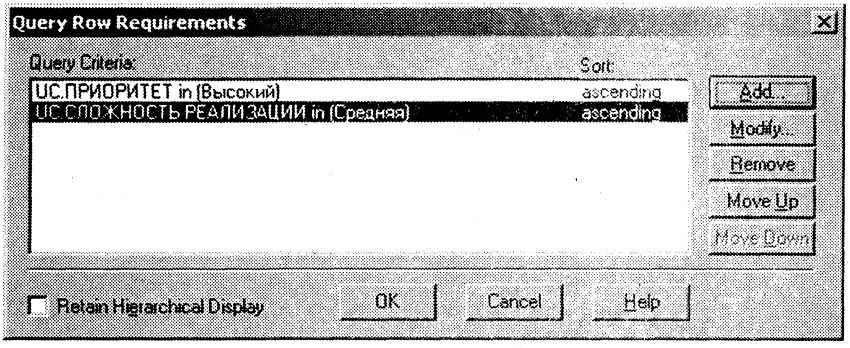


Рис. 3.41. Окно "Query Row Requirements"

В результате матрица атрибутов примет вид, показанный на рис. 3.42. Для того чтобы вернуть ее в исходное состояние, следует выбрать в меню пункт View > Refresh.

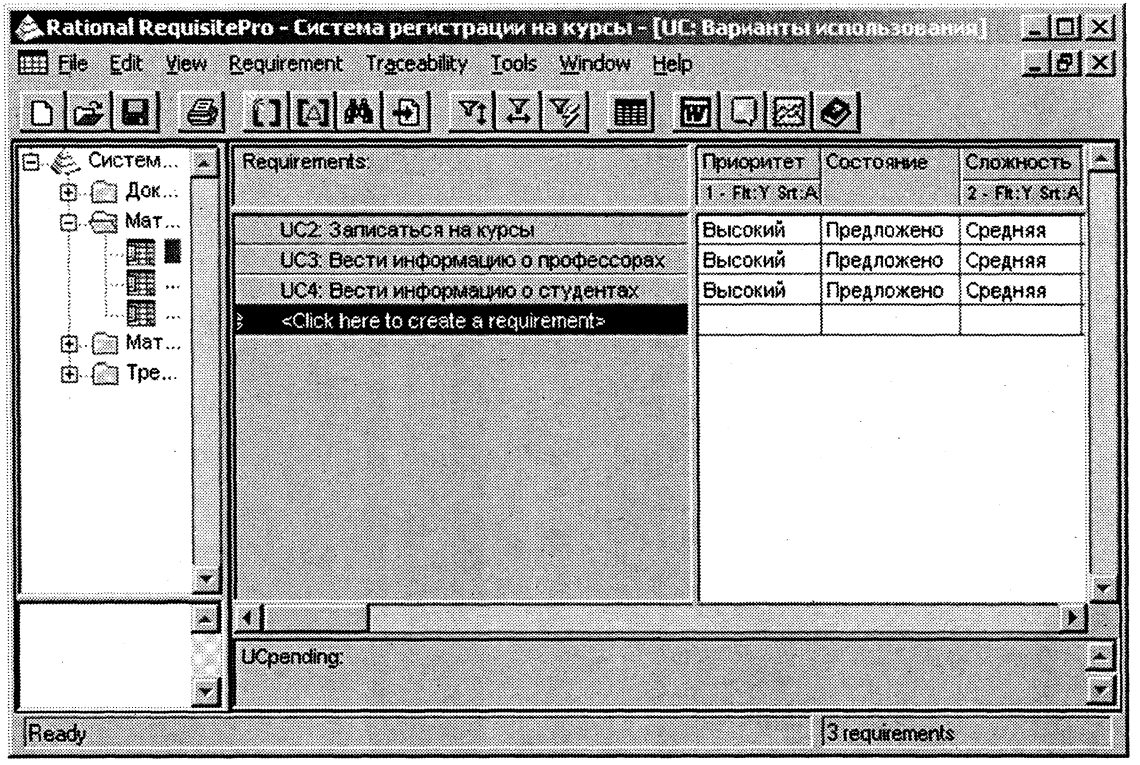


Рис. 3.42. Результат выполнения запроса

АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

4.1. АРХИТЕКТУРНЫЙ АНАЛИЗ

Целью объектно-ориентированного анализа является трансформация функциональных требований к ПО в предварительный системный проект и создание стабильной основы архитектуры системы. В процессе проектирования системный проект "погружается" в среду реализации с учетом всех нефункциональных требований.

Объектно-ориентированный анализ включает два вида деятельности – архитектурный анализ и анализ вариантов использования.

Архитектурный анализ выполняется архитектором системы и включает:

- утверждение общих стандартов (соглашений) моделирования и документирования системы;
- предварительное выявление архитектурных механизмов (механизмов анализа);
- формирование набора основных абстракций предметной области (классов анализа);
- формирование начального представления архитектурных уровней.

Соглашения моделирования определяют:

- используемые диаграммы и элементы модели;
- правила их применения;
- соглашения по именованию элементов модели;
- организацию модели (пакеты).

Пример набора соглашений моделирования:

- имена вариантов использования должны быть короткими глагольными фразами;
- имена классов должны быть существительными, соответствующими по возможности понятиям предметной области;

• имена классов должны писаться с прописной буквы;

• имена атрибутов и операций должны писаться со строчной буквы;

• составные имена должны быть сплошными, без подчеркиваний, каждое отдельное слово должно писаться с прописной буквы;

• все классы и диаграммы, описывающие предварительный системный проект, помещаются в пакет с именем Analysis Model;

• диаграммы классов, реализующих вариант использования, и диаграммы взаимодействия, отражающие взаимодействие объектов в процессе реализации сценариев варианта использования, помещаются в кооперацию с именем данного варианта использования и стереотипом <<use case realization>>. Все кооперации помещаются в пакет с именем Use Case Realizations. Связь между вариантом использования и его реализацией изображается на специальной диаграмме трассировки (рис. 4.1).

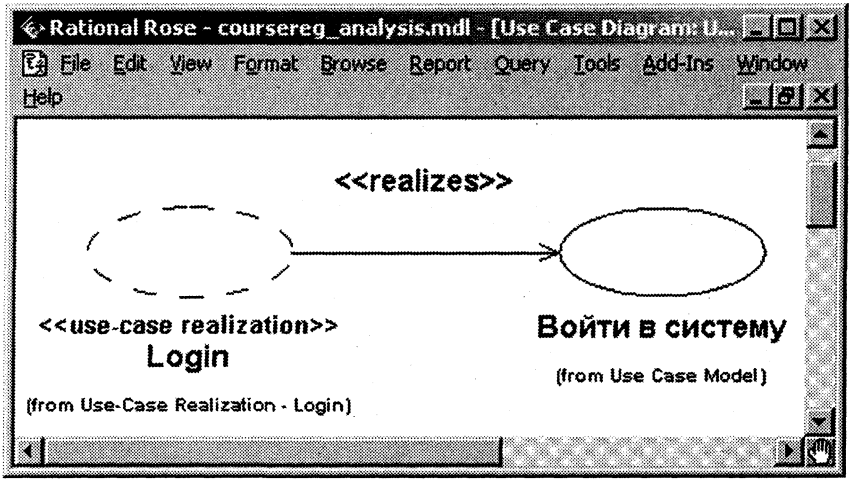


Рис. 4.1. Фрагмент диаграммы трассировки

Идентификация основных абстракций заключается в предварительном определении набора классов системы (классов анализа) на основе описания предметной области и спецификации требований к системе (в частности, глоссария). Способы идентифика-

ции основных абстракций аналогичны способам идентификации сущностей в модели "сущность-связь". Основной (неформальный) способ идентификации сущностей – это поиск абстракций, описывающих физические или материальные объекты, процессы и события, роли людей, организации и другие понятия. Единственным формальным способом идентификации сущностей является анализ текстовых описаний предметной области, выделение из описаний имен существительных и выбор их в качестве "кандидатов" на роль абстракций. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Следуя этим рекомендациям, определим для системы регистрации пять классов анализа (рис. 4.2.):

- Student (Студент);
- Professor (Профессор);
- Schedule (Учебный график);
- Course (Курс);
- CourseOffering (Предлагаемый курс).

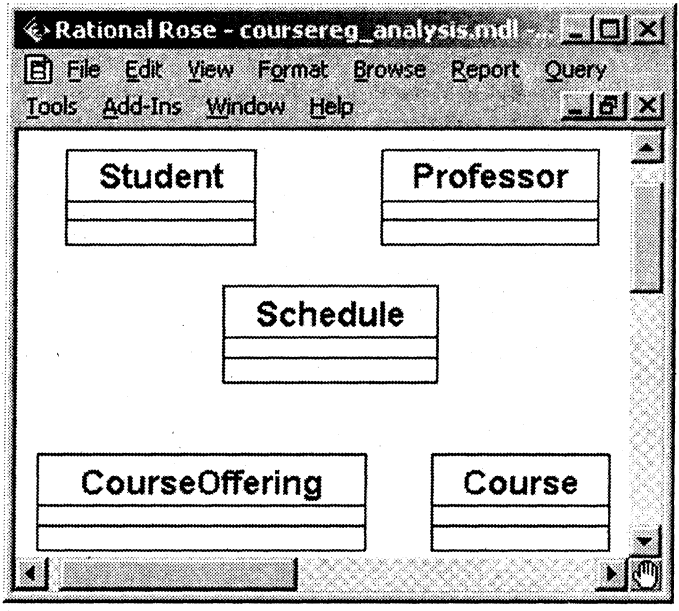


Рис. 4.2. Классы анализа системы регистрации

Упражнение 4.1.

Создание структуры модели и классов анализа в соответствии с требованиями архитектурного анализа

Для того чтобы создать пакеты и диаграммы трассировки:

1. Щелкните правой кнопкой мыши по пакету Use Case Realizations, входящему в пакет Design Model логического представления браузера.

2. Выберите пункт New > Package в открывшемся меню.

3. Создайте пакет с наименованием Use Case Realization – Register for Courses, затем таким же образом пакеты Use Case Realization – Close Registration и Use Case Realization – Login.

4. Создайте в каждом из пакетов типа Use Case Realization соответствующие кооперации Register for Courses, Close Registration и Login (каждая кооперация создается как вариант использования со стереотипом <<use case realization>>, который задается в окне спецификации варианта использования путем выбора нужного стереотипа из списка стереотипов).

5. Создайте в каждом из пакетов типа Use Case Realization новую диаграмму вариантов использования с названием Realize Dependency и постройте ее в соответствии с рис. 4.1, подставляя соответствующие кооперации и варианты использования.

Для того чтобы создать классы анализа и соответствующую диаграмму Key Abstractions:

1. Щелкните правой кнопкой мыши по пакету Analysis Model логического представления браузера.

2. Выберите пункт New > Class в открывшемся меню. Новый класс под названием NewClass появится в браузере.

3. Выделите его и введите имя Student.

4. Создайте аналогичным образом классы Professor, Schedule, Course и CourseOffering.

5. Щелкните правой кнопкой мыши по пакету Analysis Model.

6. Выберите пункт New > Class Diagram в открывшемся меню.

7. Назовите новую диаграмму классов Key Abstractions.

8. Откройте диаграмму классов и перетащите классы на открытую диаграмму мышью, для того чтобы расположить вновь созданные классы. Диаграмма классов должна выглядеть, как показано на рис. 4.2.

Архитектурные уровни образуют иерархию уровней представления системы. В практике разработки клиент-серверных систем

существует ряд типовых решений – архитектурных образцов, среди которых наиболее распространен образец "Уровни" (Layers). В соответствии с ним базовый вариант системы включает следующие уровни (сверху вниз):

- прикладной – набор компонентов, реализующих основную функциональность системы, отраженную в вариантах использования;
- бизнес-уровень – набор компонентов, специфичных для конкретной предметной области;
- промежуточный – различные платформо-независимые сервисы (библиотеки пользовательского интерфейса, брокеры запросов и др.);
- системный – ПО для вычислительной и сетевой инфраструктур (ОС, сетевые протоколы и др.).

Архитектурные уровни представляются в модели в виде пакетов со стереотипом <<layer>>. В рамках архитектурного анализа определяется начальная структура модели и рассматриваются только верхние уровни (прикладной и бизнес-уровень).

Для моделирования архитектурных уровней создайте в пакете Design Model два пакета с наименованиями "Application" и "Business Services"

4.2. АНАЛИЗ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Анализ вариантов использования включает:

- идентификацию классов, участвующих в реализации потоков событий варианта использования;
- распределение поведения, реализуемого вариантом использования, между классами (определение обязанностей классов);
- определение атрибутов и ассоциаций классов;
- унификацию классов анализа.

4.2.1. ИДЕНТИФИКАЦИЯ КЛАССОВ

В потоках событий варианта использования выявляются классы трех типов:

1. ***Граничные классы*** (Boundary) – посредники при взаимодействии внешних объектов с системой. Как правило, для каждой пары "действующее лицо – вариант использования" определяется один граничный класс. Типы граничных классов: пользова-

тельский интерфейс (обмен информацией с пользователем без деталей интерфейса – кнопок, списков, окон), системный интерфейс и аппаратный интерфейс (используемые протоколы без деталей их реализации).

2. ***Классы-сущности*** (Entity) – ключевые абстракции (понятия) разрабатываемой системы. Источники выявления классов-сущностей: ключевые абстракции, созданные в процессе архитектурного анализа, глоссарий, описание потоков событий вариантов использования.

3. ***Управляющие классы*** (Control) – обеспечивают координацию поведения объектов в системе. Могут отсутствовать в некоторых вариантах использования, ограничивающихся простыми манипуляциями с хранимыми данными. Как правило, для каждого варианта использования определяется один управляющий класс. Примеры управляющих классов: менеджер транзакций, координатор ресурсов, обработчик ошибок.

Классы анализа отражают функциональные требования к системе и моделируют объекты предметной области. Совокупность классов анализа представляет собой начальную концептуальную модель системы. Пример набора классов, участвующих в реализации варианта использования "Зарегистрироваться на курсы", приведен на рис. 4.3.

Упражнение 4.2.

Создание классов, участвующих в реализации варианта использования "Зарегистрироваться на курсы", и диаграммы классов

Для выполнения упражнения:

1. Щелкните правой кнопкой мыши по пакету Analysis Model.
2. Выберите пункт New > Class в открывшемся меню. Новый класс под названием NewClass появится в браузере.
3. Выделите его и введите имя RegisterForCoursesForm.
4. Щелкните правой кнопкой мыши по классу RegisterForCoursesForm.
5. Выберите пункт Open Specification в открывшемся меню.
6. Выберите стереотип <<Boundary>> в поле стереотипа и щелкните по кнопке OK.
7. Создайте аналогичным образом классы Course Catalog System со стереотипом <<Boundary>> и RegistrationController со стереотипом Control.

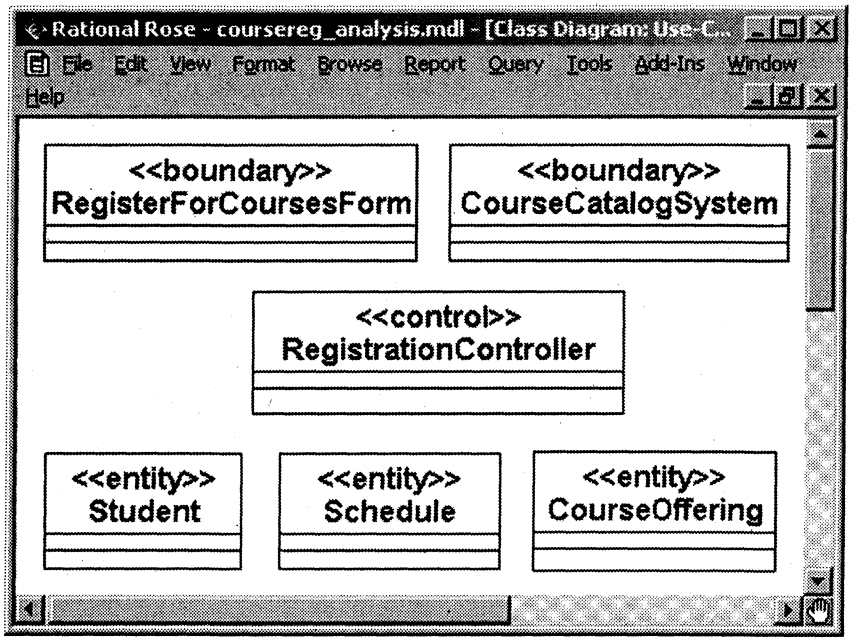


Рис. 4.3. Классы, участвующие в реализации варианта использования

8. Назначьте классам Schedule, CourseOffering и Student стереотип Entity.

9. Щелкните правой кнопкой мыши по кооперации Register for Courses в пакете Use Case Realization – Register for Courses.

10. Выберите пункт New > Class Diagram в открывшемся меню.

11. Назовите новую диаграмму классов Register for Courses – Participating Classes.

12. Откройте ее и перетащите классы на открытую диаграмму в соответствии с рис. 4.3.

4.2.2. РАСПРЕДЕЛЕНИЕ ОБЯЗАННОСТЕЙ МЕЖДУ КЛАССАМИ

Исходя из назначения трех выделенных типов классов, можно кратко охарактеризовать распределение обязанностей между ними:

- граничные классы отвечают за взаимодействие с внешней средой системы (действующими лицами);

• классы-сущности отвечают за хранение данных и манипулирование ими;

• управляющие классы координируют потоки событий варианта использования.

Более детальное распределение обязанностей (в виде операций классов) выполняется с помощью диаграмм взаимодействия (диаграмм последовательности и кооперативных диаграмм). В первую очередь строится диаграмма (одна или больше), описывающая основной поток событий и его подчиненные потоки. Для каждого альтернативного потока событий строится отдельная диаграмма. Примеры:

• обработка ошибок;

• контроль времени выполнения;

• обработка неправильных вводимых данных.

Нецелесообразно описывать тривиальные потоки событий (например, в потоке участвует только один объект).

Упражнение 4.3.

Создание диаграмм взаимодействия

Создадим диаграммы последовательности и кооперативные диаграммы для основного потока событий варианта использования "Зарегистрироваться на курсы". Готовые диаграммы последовательности должны иметь вид, показанный на рис. 4.4.

Настройка:

1. Выберите пункт Tools > Options в меню модели.

2. Перейдите на вкладку диаграмм.

3. Контрольные переключатели Sequence Numbering, Collaboration Numbering должны быть помечены, а Focus of Control – нет.

4. Щелкните по кнопке OK, чтобы выйти из окна параметров.

Создание диаграммы последовательности:

1. Щелкните правой кнопкой мыши по кооперации Register for Courses в пакете Use-Case Realization – Register for Courses.

2. Выберите пункт New > Sequence Diagram в открывшемся меню.

3. Назовите новую диаграмму Register for Courses – Basic Flow.

4. Щелкните по ней дважды, чтобы открыть ее.

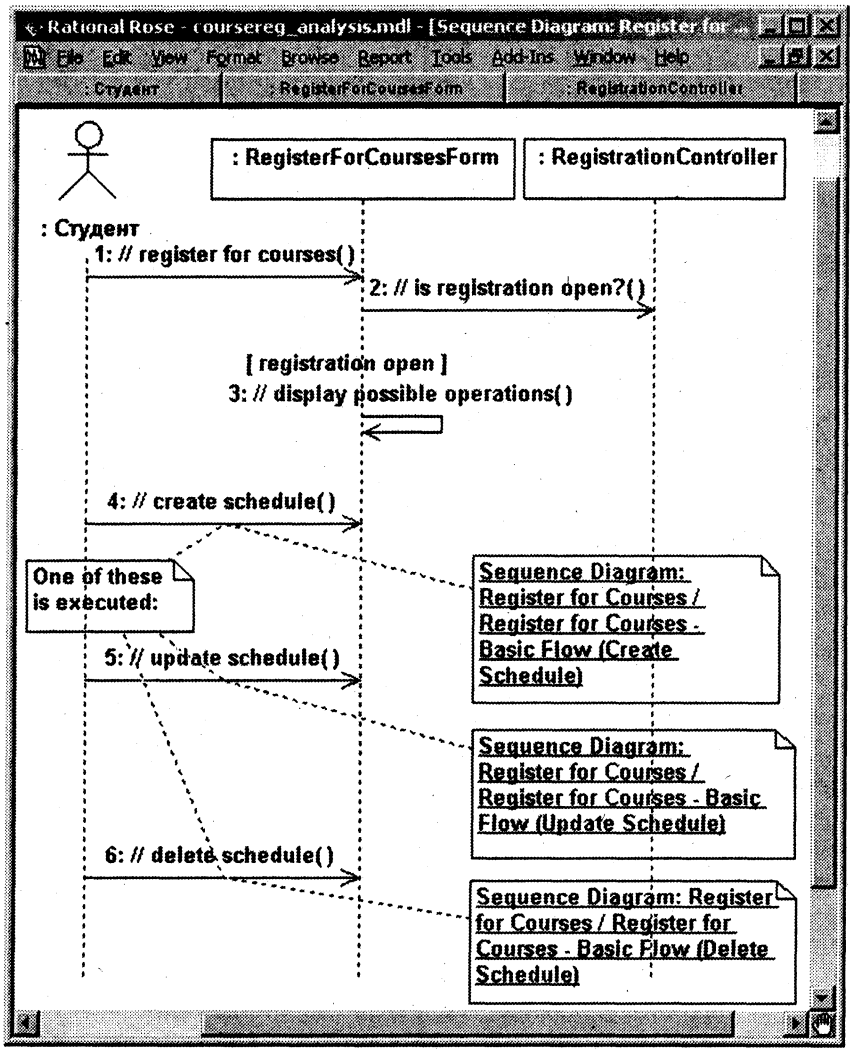


Рис. 4.4. Диаграмма последовательности Register for Courses – Basic Flow

Добавление на диаграмму действующего лица, объектов и сообщений:

1. Перетащите действующее лицо "Студент" из браузера на диаграмму.

2. Перетащите классы RegisterForCoursesForm и RegistrationController из браузера на диаграмму. Можно переместить объект из верхней части диаграммы в точку его создания. Для того чтобы расположить объект между двумя существующими объектами, достаточно щелкнуть мышью между ними.

3. Щелкните по кнопке Object Message на панели инструментов.

4. Проведите мышью от линии жизни действующего лица "Студент" к линии жизни объекта RegisterForCoursesForm.

5. Выделите сообщение и введите его имя: // register for courses.

6. Повторите действия п. 3–5, чтобы поместить на диаграмму остальные сообщения, как показано на рис. 4.4 (для рефлексивного сообщения 3 используется кнопка Message to Self).

Соотнесение сообщений с операциями:

1. Щелкните правой кнопкой мыши по тексту сообщения 1, // register for courses.

2. Выберите пункт new operation в открывшемся меню. Появится окно спецификации операции.

3. Оставьте имя сообщения – // register for courses в поле имени.

4. Щелкните по кнопке OK, чтобы закрыть окно спецификации операции и вернуться на диаграмму.

5. Повторяйте действия п. 1–4, пока не соотнесете с операциями все остальные сообщения.

Выполните аналогичные действия для создания диаграмм последовательности, показанных на рис. 4.5 – 4.8. Обратите внимание, что на диаграмме рис. 4.8 изображен объект нового класса PrimarySheduleOfferingInfo (ассоциации-класса, описывающего связь между классами Shedule и OfferingInfo), который нужно предварительно создать.

Для того чтобы добавить на диаграмму примечание:

1. Щелкните на панели инструментов по кнопке Note.

2. Щелкните мышью по тому месту диаграммы, куда собираетесь поместить примечание.

3. Выделите новое примечание и введите туда текст.

4. Щелкните по кнопке Anchor Notes To Item на панели инструментов, для того чтобы прикрепить примечание к элементу диаграммы.

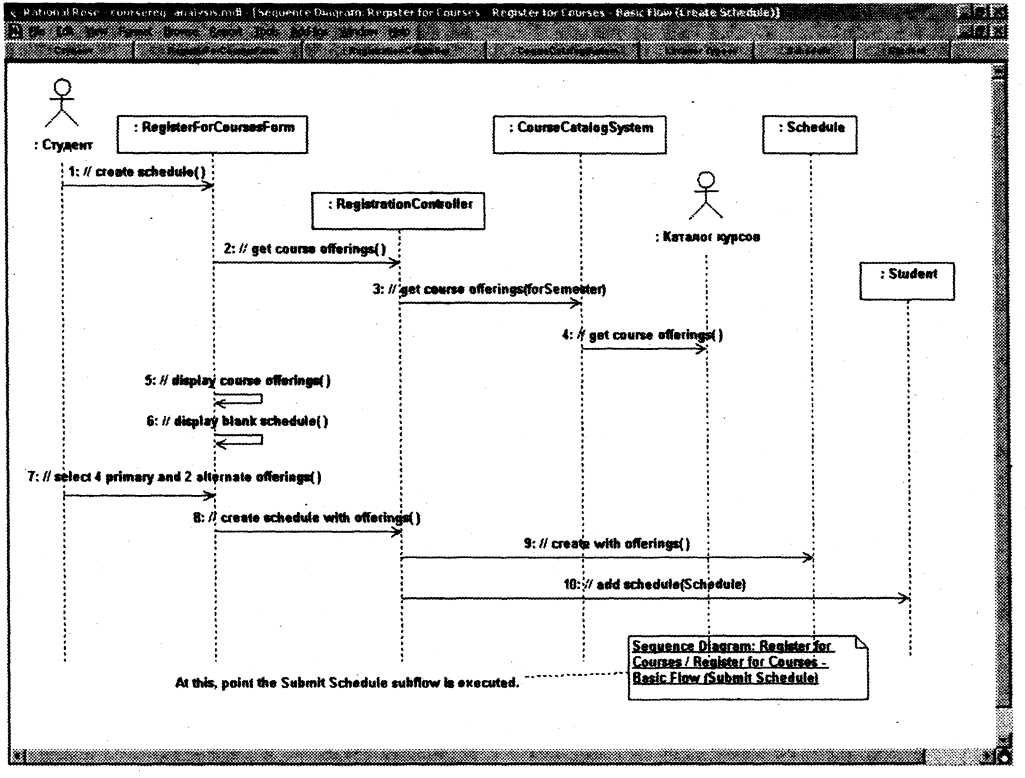


Рис. 4.5. Диаграмма последовательности Register for Courses – Basic Flow (Create Schedule)

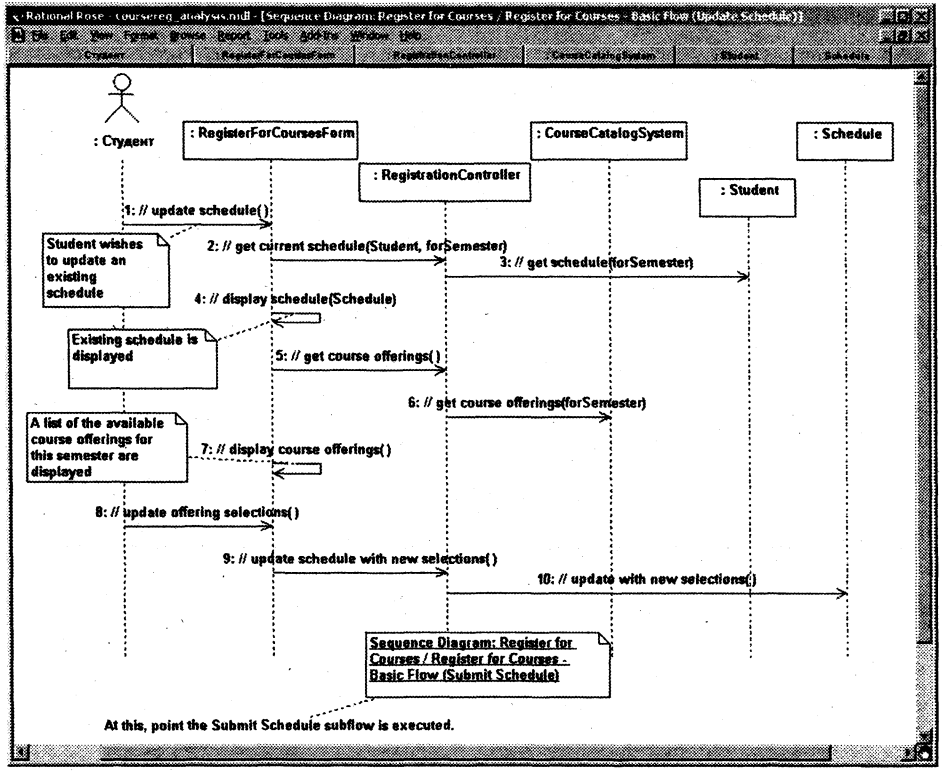


Рис. 4.6. Диаграмма последовательности Register for Courses – Basic Flow (Update Schedule)

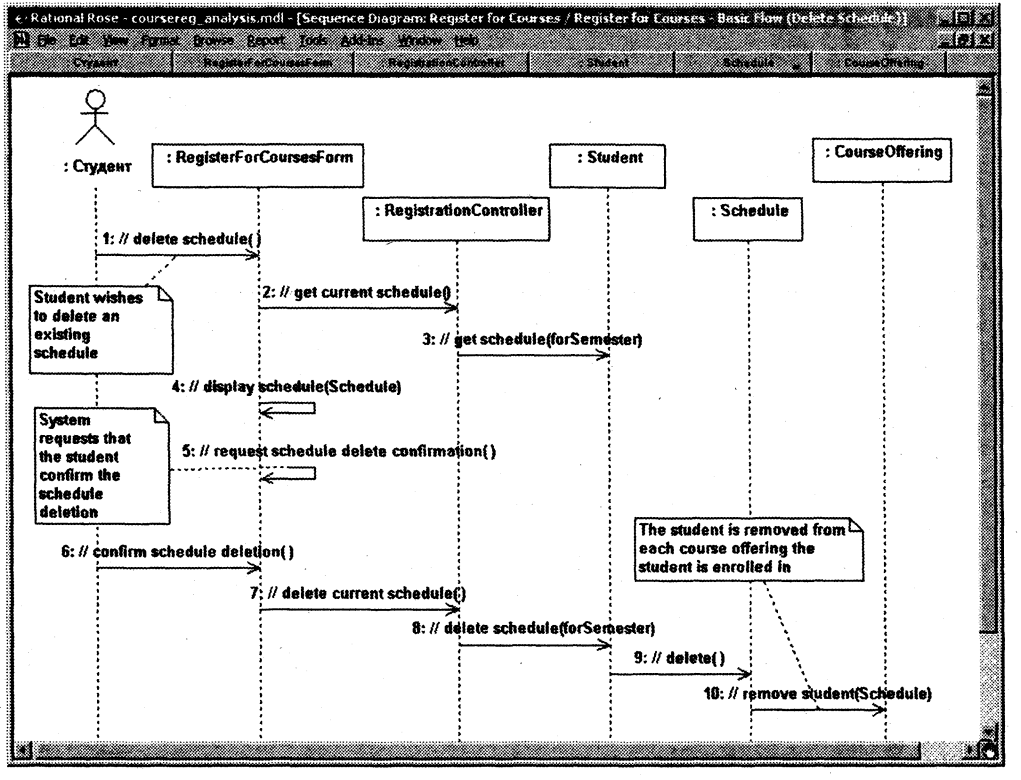


Рис. 4.7. Диаграмма последовательности Register for Courses – Basic Flow (Delete Schedule)

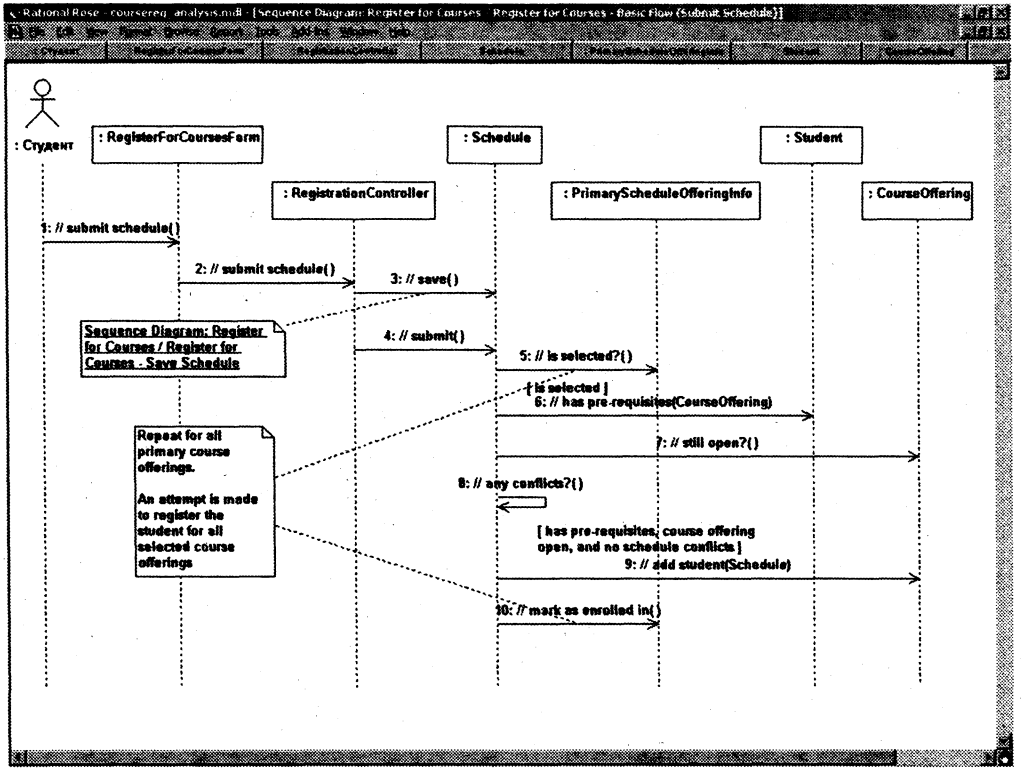


Рис. 4.8. Диаграмма последовательности Register for Courses – Basic Flow (Submit Schedule)

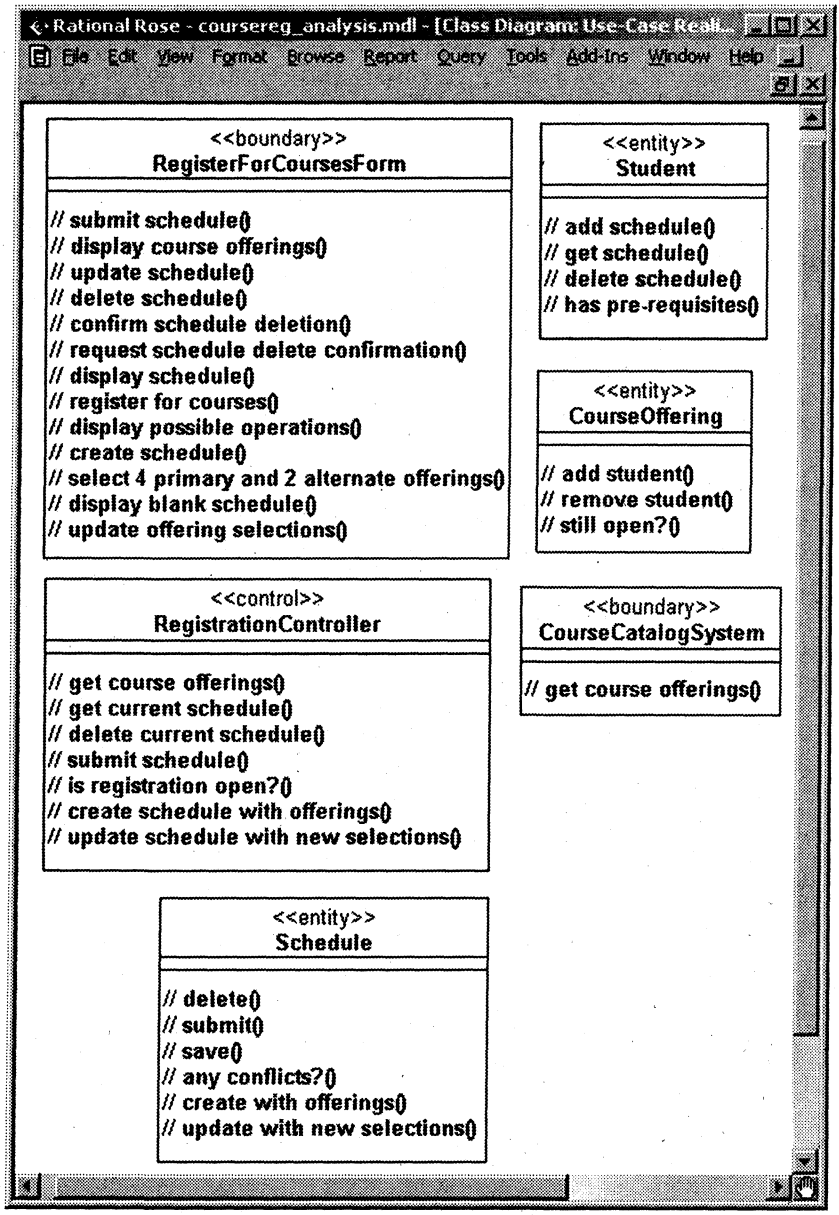


Рис. 4.9. Диаграмма классов Register for Courses – Participating Classes с операциями "анализа"

5. Щелкните по левой кнопке мыши и проведите указатель от примечания до элемента диаграммы, с которым оно будет связано. Между примечанием и элементом возникнет штриховая линия.

6. Создайте пустое примечание (без текста) и перетащите на него из браузера нужную диаграмму, для того чтобы создать примечание-ссылку на другую диаграмму (см. диаграмму рис. 4.4 и др.).

Кроме примечаний на диаграмму можно поместить также текстовую область и с ее помощью, например, добавить к сообщению условие, как [registration open] на рис. 4.4.

Для того чтобы поместить на диаграмму текстовую область:

1. Щелкните по кнопке Text Box на панели управления.

2. Щелкните мышью внутри диаграммы, чтобы поместить туда текстовую область.

3. Выделите эту область и введите в нее текст.

Для создания кооперативной диаграммы достаточно открыть диаграмму последовательности и нажать клавишу F5.

В процессе построения диаграмм взаимодействия (соотнесения сообщений с операциями) в классах автоматически появляются операции "анализа". Таким образом, диаграмма классов Register for Courses – Participating Classes (см. рис. 4.3) после построения диаграмм взаимодействия, описанного в упражнении 4.3, должна принять вид, показанный на рис. 4.9.

4.2.3. ОПРЕДЕЛЕНИЕ АТРИБУТОВ И АССОЦИАЦИЙ КЛАССОВ

Атрибуты классов анализа определяются исходя из знаний о предметной области, требований к системе и глоссария.

Упражнение 4.4.

Добавление атрибутов к классам

Настройка:

1. Выберите пункт Tools > Options в меню модели.

2. Перейдите на вкладку "Diagram".

3. Убедитесь, что переключатель Show All Attributes помечен.

4. Убедитесь, что переключатели Suppress Attributes и Suppress Operations не помечены.

Добавление атрибутов:

1. Щелкните правой кнопкой мыши по классу Student.

2. Выберите пункт New Attribute в открывшемся меню.
3. Введите новый атрибут Address.
4. Нажмите на клавишу Enter.
5. Повторите действия п. 1–4, добавив атрибуты name и studentID.
6. Добавьте атрибуты к классам CourseOffering и Shedule (рис. 4.10).

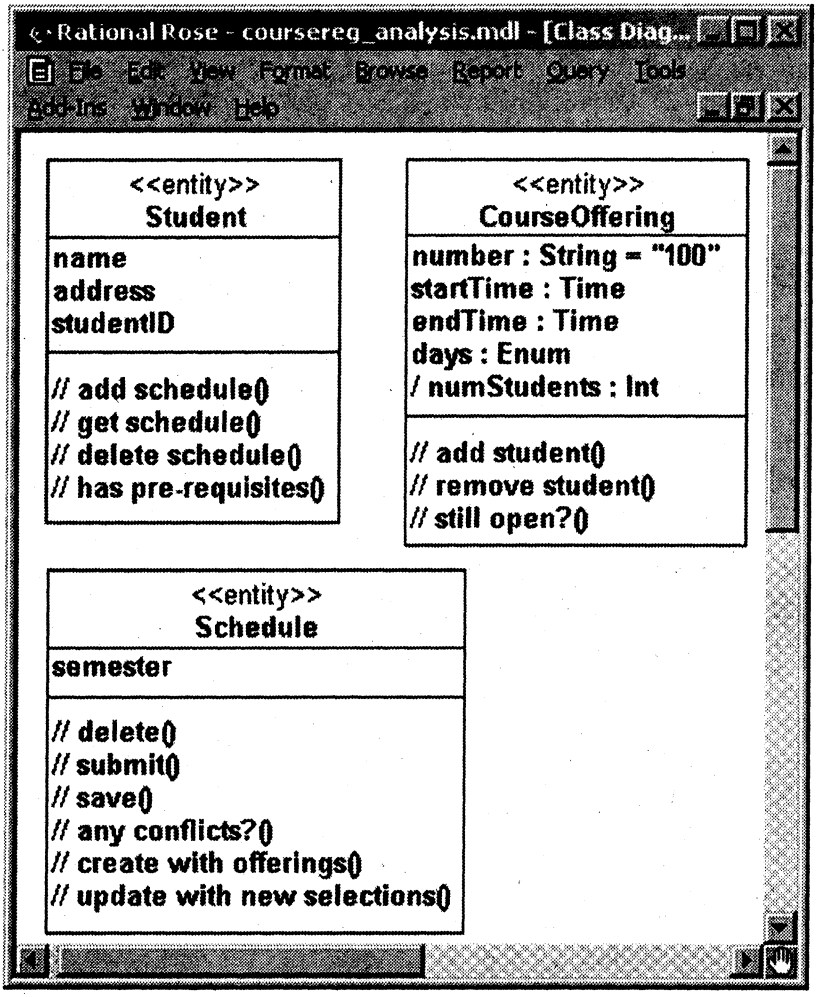


Рис. 4.10. Классы с операциями "анализа" и атрибутами

Связи между классами (ассоциации) определяются в два этапа:

Этап 1. Начальный набор связей устанавливается на основе анализа кооперативных диаграмм. Если два объекта взаимодействуют (обмениваются сообщениями), между ними на кооперативной диаграмме должна существовать связь (путь взаимодействия), которая преобразуется в двунаправленную ассоциацию между соответствующими классами. Если сообщения между некоторой парой объектов передаются только в одном направлении, то для соответствующей ассоциации вводится направление навигации.

Этап 2. Анализируются и уточняются ассоциации между классами-сущностями. Задаются мощности ассоциаций, могут использоваться множественные ассоциации, агрегации, обобщения и ассоциации-классы.

Упражнение 4.5.
Добавление связей

Добавим связи к классам, принимающим участие в варианте использования Register for Courses. Для отображения связей между классами построим три новых диаграммы классов в кооперации Register for Courses пакета Use Case Realization – Register for Courses (рис. 4.11 – 4.13).

На рис. 4.11 показаны только классы-сущности. Агрегация между классами Student и Schedule отражает тот факт, что каждый график является собственностью конкретного студента, принадлежит только ему. Предполагается также, что в системе будет храниться не только график текущего семестра, но и все графики студента за разные семестры. Между классами Schedule и CourseOffering введены две ассоциации, поскольку конкретный курс может входить в график студента в качестве основного (не больше четырех курсов) и альтернативного (не больше двух курсов). К классу Student добавлены два новых подкласса – FulltimeStudent (студент очного отделения) и ParttimeStudent (студент вечернего отделения).

На рис. 4.12 показаны ассоциации-классы, представляющие свойства связей между классами Schedule и CourseOffering. Ассоциация, связывающая график и альтернативный курс, имеет только один атрибут – статус курса в конкретном графике (status), который может принимать значения "включен в график", "отменен",

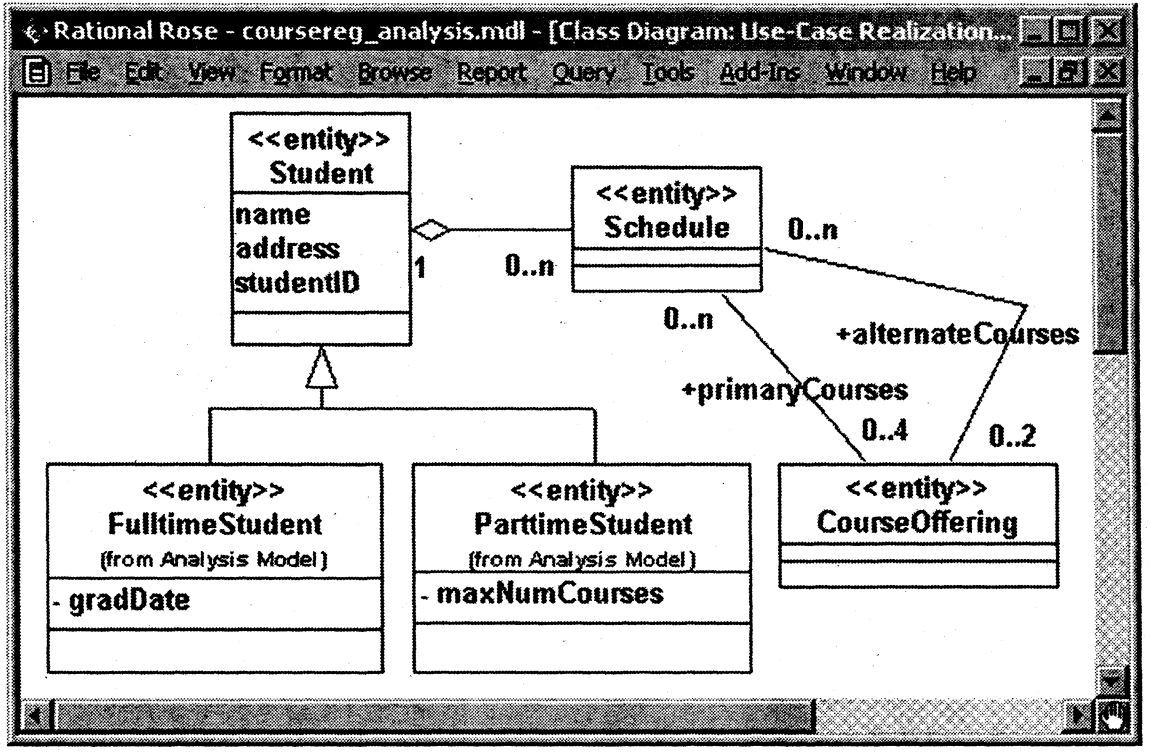


Рис. 4.11. Диаграмма Entity Classes (классы-сущности)

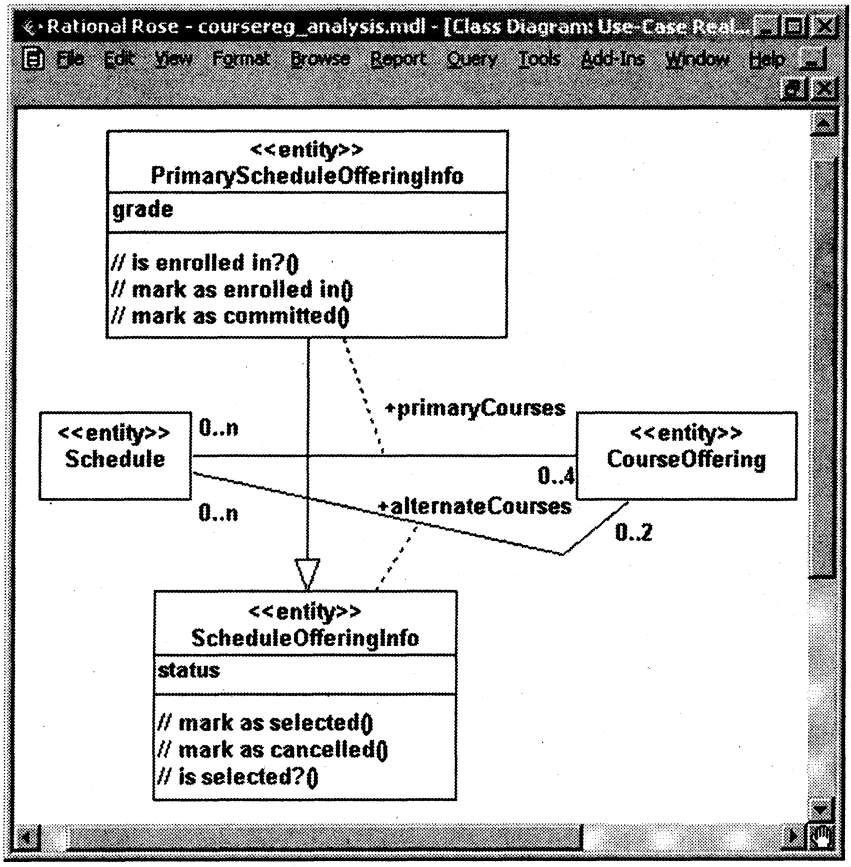


Рис. 4.12. Диаграмма CourseOfferingInfo (пример ассоциаций-классов)

"внесен в список курса" и "зафиксирован в графике". Если курс в процессе закрытия регистрации переходит из альтернативного в основные, то к соответствующей ассоциации добавляется атрибут "оценка" (grade). Таким образом, ассоциация-класс PrimaryScheduleOfferingInfo наследует свойства ассоциации-класса ScheduleOfferingInfo (атрибуты и операции, содержащиеся в этом классе, относятся как к основным, так и к альтернативным курсам) и добавляет свои собственные (оценка и окончательное включение курса в график могут иметь место только для основных курсов), что показано с помощью связи обобщения.

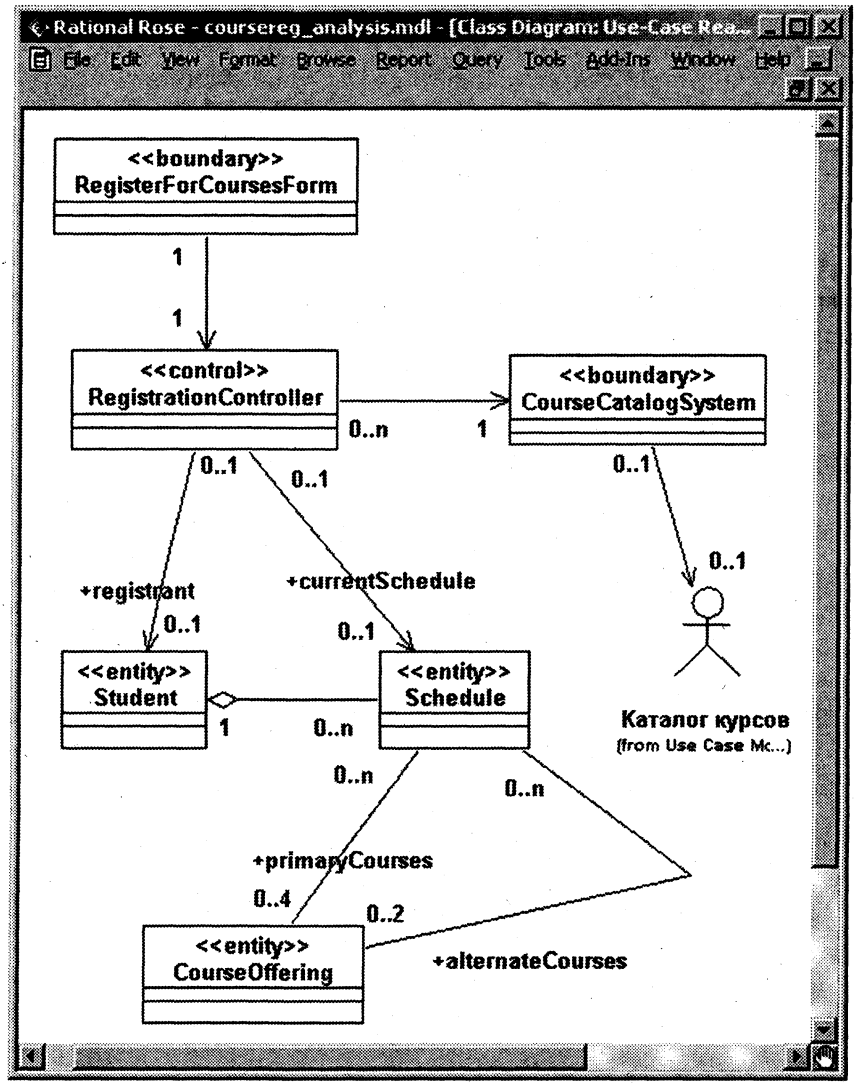


Рис. 4.13. Полная диаграмма классов Register for Courses – Participating Classes (без атрибутов и операций)

На рис. 4.13 показана полная диаграмма классов варианта использования "Зарегистрироваться на курсы" (без атрибутов и операций). Ассоциации между граничными и управляющими

классами, а также между управляющими классами и классами-сущностями введены на основе анализа кооперативных диаграмм и в отличие от устойчивых структурных (семантических) связей между сущностями отражают связи, динамически возникающие между соответствующими объектами в потоке управления (в процессе работы приложения). Поскольку для ассоциаций это не свойственно, в дальнейшем (в процессе проектирования) они могут быть преобразованы в зависимости.

Ассоциации создают непосредственно на диаграмме классов. Панель инструментов диаграммы классов содержит кнопки для создания как одно-, так и двунаправленных ассоциаций.

Для того чтобы создать ассоциацию на диаграмме классов:

1. Щелкните по кнопке Association на панели инструментов.
2. Проведите мышью линию ассоциации от одного класса к другому.

Для того чтобы задать возможности навигации по ассоциации:

1. Щелкните правой кнопкой мыши на связи по тому концу, на котором хотите показать стрелку.
2. Выберите пункт Navigable в открывшемся меню.

Для того чтобы создать рефлексивную ассоциацию:

1. Щелкните по кнопке Association на панели инструментов диаграммы.
2. Проведите линию ассоциации от класса до какого-нибудь места вне класса.
3. Отпустите кнопку мыши.
4. Проведите линию ассоциации назад к классу.

Для того чтобы создать агрегацию:

1. Щелкните по кнопке Aggregation панели инструментов.
2. Проведите линию агрегации от класса-части к целому.

Для того чтобы поместить на диаграмму классов рефлексивную агрегацию:

1. Щелкните по кнопке Aggregation на панели инструментов диаграммы.
2. Проведите линию агрегации от класса до какого-нибудь места вне класса.
3. Отпустите кнопку мыши.
4. Проведите линию агрегации назад к классу.

При создании обобщения может потребоваться перенести некоторые атрибуты или операции из одного класса в другой. Если,

например, понадобится перенести их из подкласса в суперкласс, в браузере достаточно просто перетащить атрибуты или операции из одного класса в другой. Не забудьте удалить копию атрибута из второго подкласса, если он имеется.

Для того чтобы поместить обобщение на диаграмму классов:

1. Щелкните по кнопке Generalization панели инструментов.
2. Проведите линию обобщения от подкласса к суперклассу.

Спецификации связей касаются имен ассоциаций, ролевых имен, мощности и ассоциаций-классов.

Для того чтобы задать мощность связи:

1. Щелкните правой кнопкой мыши по одному концу связи.
2. Выберите пункт Multiplicity в открывшемся меню.
3. Укажите нужную мощность.
4. Повторите п. 1–3 для другого конца связи.

Для того чтобы задать имя связи:

1. Выделите нужную связь.
2. Введите ее имя.

Для того чтобы задать связи ролевое имя:

1. Щелкните правой кнопкой мыши по ассоциации с нужного конца.
2. Выберите пункт role Name в открывшемся меню.
3. Введите ролевое имя.

Для того чтобы задать элемент связи (ассоциацию-класс):

1. Откройте окно спецификации требуемой связи.
2. Перейдите на вкладку "Detail".
3. Задайте элемент связи в поле Link Element.

Задание для самостоятельной работы

Выполните анализ варианта использования Close Registration и постройте соответствующие диаграммы взаимодействия и классов.

ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Целью объектно-ориентированного проектирования является адаптация предварительного системного проекта (набора классов "анализа"), составляющего стабильную основу архитектуры системы, к среде реализации с учетом всех нефункциональных требований.

Объектно-ориентированное проектирование включает два вида деятельности:

- проектирование архитектуры системы;
- проектирование элементов системы.

5.1. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СИСТЕМЫ

Проектирование архитектуры системы включает:

- идентификацию архитектурных решений и механизмов;
- анализ взаимодействий между классами "анализа", выявление подсистем и интерфейсов;
- формирование архитектурных уровней;
- проектирование структуры потоков управления;
- проектирование распределенной конфигурации системы.

5.1.1. ВЫЯВЛЕНИЕ ПОДСИСТЕМ И ИНТЕРФЕЙСОВ

Первым действием архитектора при выявлении подсистем является преобразование классов "анализа" в проектные классы (design classes). По каждому классу "анализа" принимается одно из двух решений:

- класс "анализа" отображается в проектный класс, если он простой или представляет единственную логическую абстракцию;
- сложный класс "анализа" может быть разбит на несколько классов, преобразован в пакет или подсистему.

Объединение классов в подсистемы проводится по следующим соображениям:

- функциональная связь: объединяются классы, участвующие в реализации варианта использования и взаимодействующие только друг с другом;
- обязательность: совокупность классов, реализующая функциональность, которая может быть удалена из системы или заменена на альтернативную;
- связанность: объединение в подсистемы сильно связанных классов;
- распределение: объединение классов, размещенных на конкретном узле сети.

Примеры возможных подсистем:

- совокупность классов, обеспечивающих сложный комплекс функций (например, безопасность и защиту данных);
- граничные классы, реализующие сложный пользовательский интерфейс или интерфейс с внешними системами;
- различные продукты: коммуникационное ПО, доступ к базам данных, общие утилиты (библиотеки), различные прикладные пакеты.

При создании подсистем в модели выполняются следующие преобразования:

- объединяемые классы помещаются в специальный пакет с именем подсистемы и стереотипом <<subsystem>>;
- спецификации операций классов, образующих подсистему, выносятся в интерфейс подсистемы – класс со стереотипом <<interface>>;
- описание интерфейса должно включать:

имя интерфейса, отражающее его роль в системе;

текстовое описание интерфейса размером с небольшой абзац, отражающее его обязанности;

описание операций интерфейса (имя, отражающее результат операции, алгоритм выполнения операции, возвращаемое значение, параметры с их типами);

характер использования операций интерфейса и порядок их выполнения документируются с помощью диаграмм взаимодействия, описывающих взаимодействие классов подсистемы при

реализации операций интерфейса, которые вместе с диаграммой классов подсистемы объединяются в кооперацию с именем интерфейса и стереотипом <<interface realization>>;

- в подсистеме создается класс-заместитель со стереотипом <<subsystem proxy>>, управляющий реализацией операций интерфейса.

Все интерфейсы подсистем должны быть полностью определены в процессе проектирования архитектуры, поскольку они будут служить в качестве точек синхронизации при параллельной разработке системы.

Как пример (для системы регистрации) приведем подсистему CourseCatalogSystem, которая создана вместо граничного класса CourseCatalogSystem. Структура и диаграммы пакета (подсистемы) CourseCatalogSystem (диаграммы классов и взаимодействия, описывающие данную подсистему и ее интерфейс) приведены на рис. 5.1 – 5.4. Классы DBCourseOfferring и CourseOfferringList отвечают за взаимодействие с БД каталога курсов в рамках JDBC.

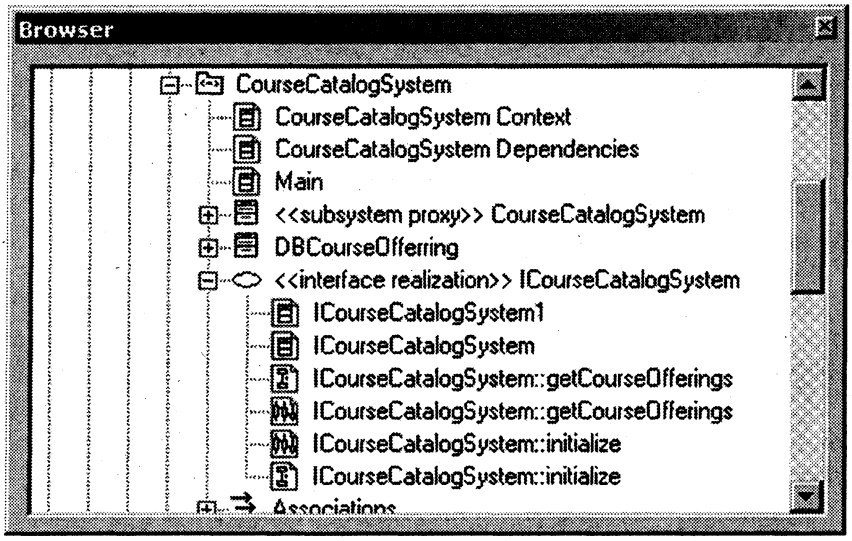


Рис. 5.1. Структура пакета CourseCatalogSystem

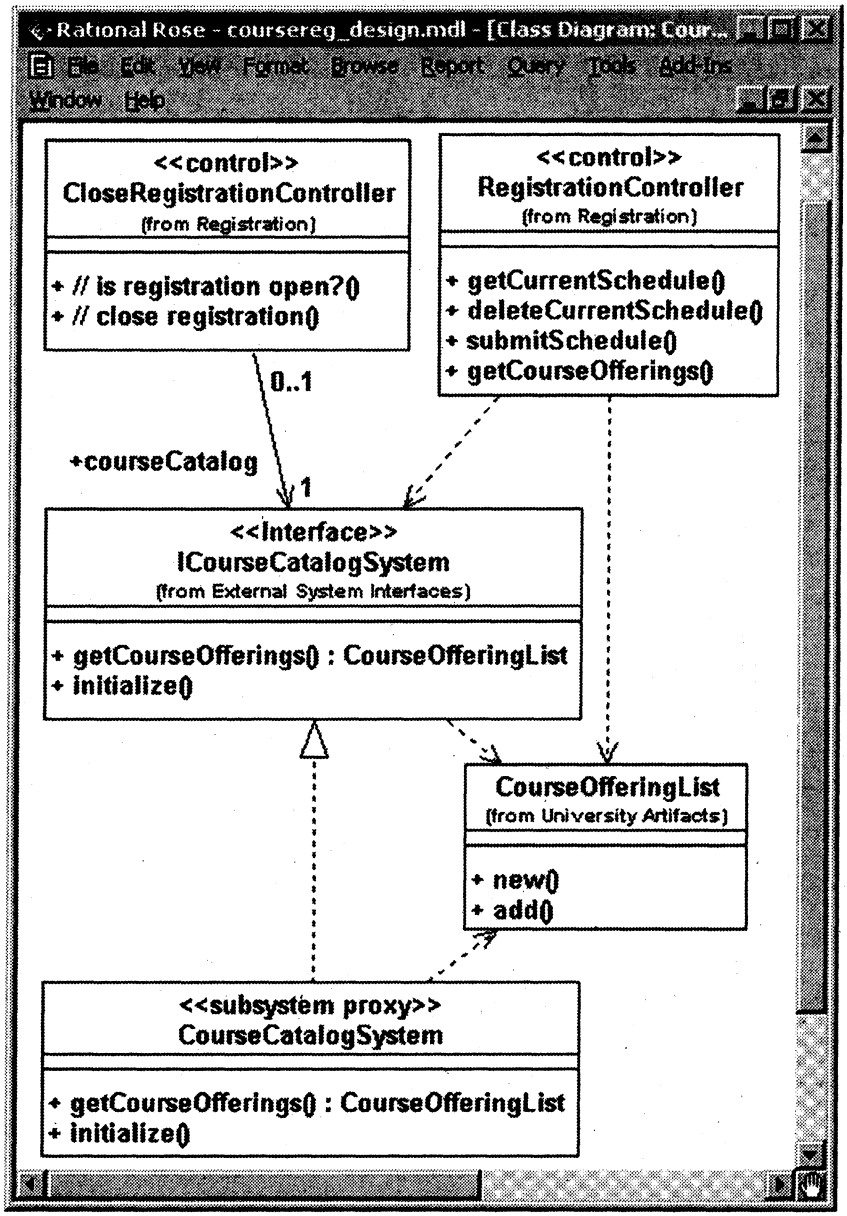


Рис. 5.2. Контекст подсистемы CourseCatalogSystem с точки зрения архитектора (диаграмма CourseCatalogSystem Context)

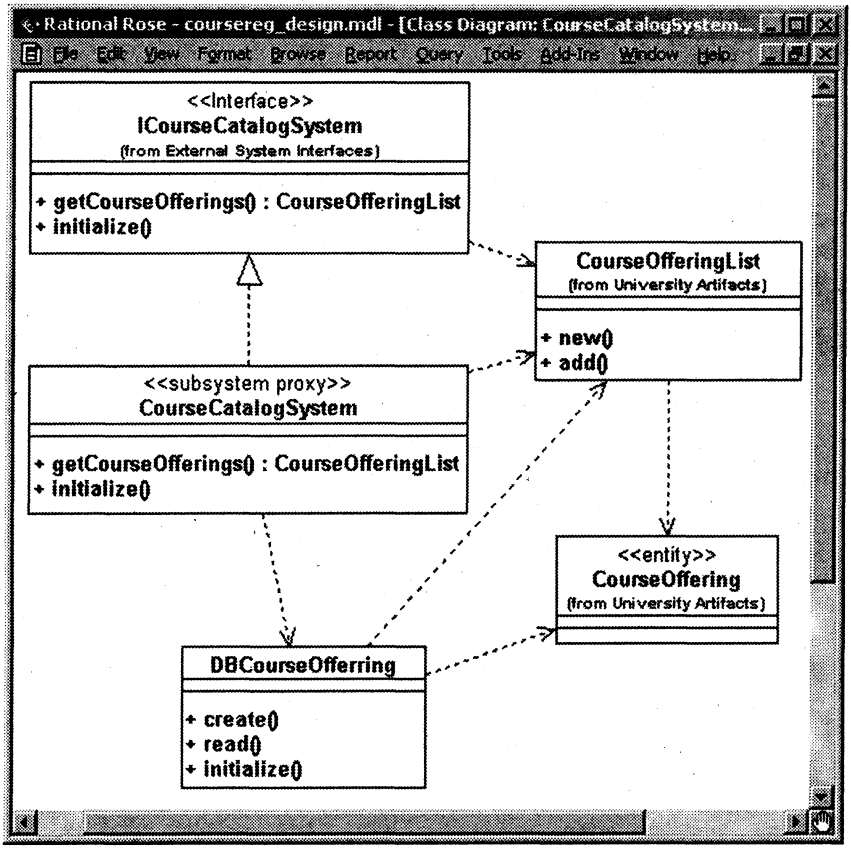


Рис. 5.3. Диаграмма классов подсистемы CourseCatalogSystem с точки зрения проектировщика (диаграмма ICourseCatalogSystem)

Объект CourseCatalogSystem Client (см. рис. 5.4) может принадлежать либо классу CloseRegistrationController, либо классу RegistrationController в зависимости от того, в каком из вариантов использования запрашивается каталог курсов.

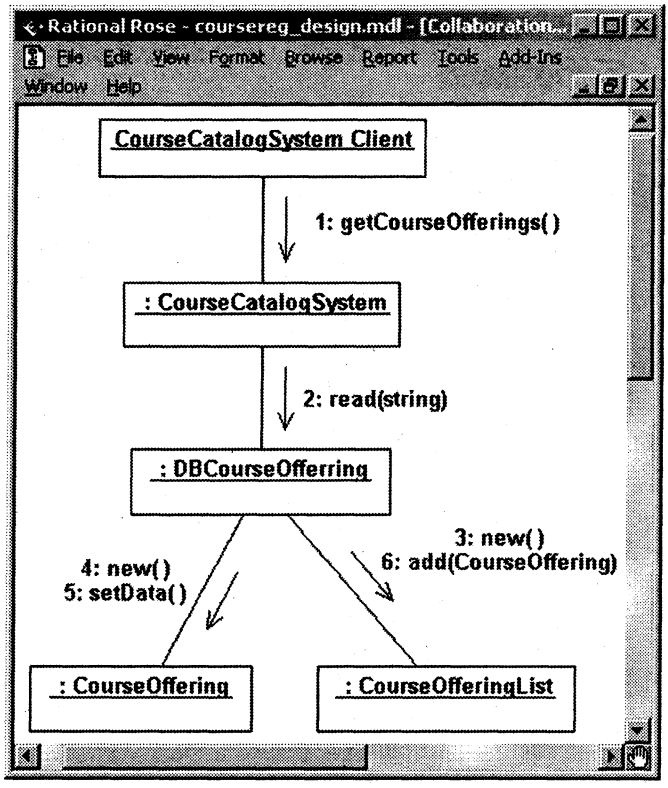


Рис. 5.4. Кооперативная диаграмма, описывающая реализацию операции интерфейса getCourseOfferings (диаграмма ICourseCatalogSystem::getCourseOfferings)

5.1.2. ФОРМИРОВАНИЕ АРХИТЕКТУРНЫХ УРОВНЕЙ

В процессе анализа было принято предварительное решение о том, чтобы выделить архитектурные уровни. В процессе проектирования все элементы системы должны быть распределены по данным уровням. С точки зрения модели это означает распределение проектных классов по пакетам, соответствующим архитектурным уровням (пакетам со стереотипом <<layer>>).

В сложной системе архитектурные уровни могут разбиваться на подуровни, количество и структура которых, как было сказано

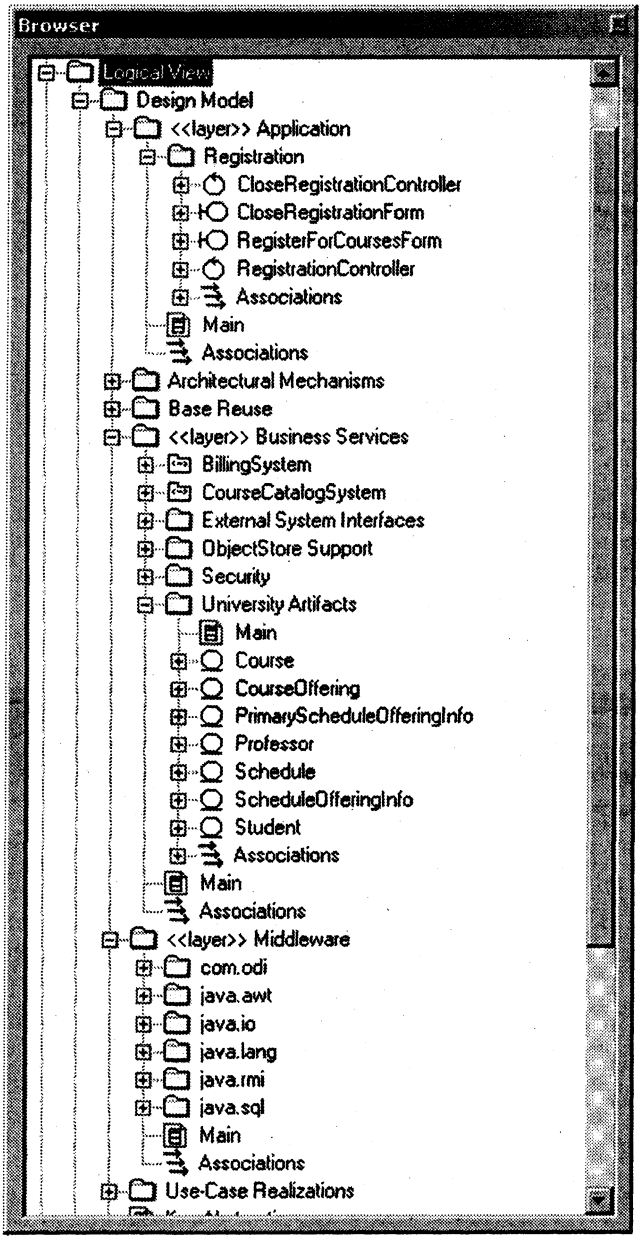


Рис. 5.5. Представление структуры модели в процессе проектирования

выше, зависят от сложности предметной области и среды реализации. Подуровни могут формироваться по следующим критериям:

- группировка элементов с максимальной связанностью;
- распределение в соответствии со структурой организации (обычно это касается верхних уровней архитектуры);
- распределение в соответствии с различными областями компетенции разработчиков (предметная область, сети, коммуникации, базы данных, безопасность и др.);
- группировка отдельных компонентов распределенной системы;
- распределение в соответствии с различной степенью безопасности и секретности.

Пример выделения архитектурных уровней и объединения элементов модели в пакеты для системы регистрации приведен на рис. 5.5.

Данное представление отражают следующие решения, принятые архитектором:

- выделены три архитектурных уровня (созданы три пакета со стереотипом <<layer>>);
- в пакете Application создан пакет Registration, куда включены граничные и управляющие классы;
- граничные классы BillingSystem и CourseCatalogSystem преобразованы в подсистемы;
- в пакет Business Services, помимо подсистем, включены еще два пакета: пакет External System Interfaces содержит интерфейсы подсистем (классы со стереотипом <<Interface>>), а пакет University Artifacts – все классы-сущности.

5.1.3. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ ПОТОКОВ УПРАВЛЕНИЯ

Проектирование структуры потоков управления выполняется при наличии в системе параллельных процессов (параллелизма). Цель проектирования – выявление процессов, существующих в системе, характера их взаимодействия, создания, уничтожения и отображения в среду реализации. Требование параллелизма возникает в следующих случаях:

- необходимо распределить обработку между различными процессорами или узлами;

- система управляется потоком событий (event-driven system);
- вычисления в системе обладают высокой интенсивностью;
- в системе одновременно работает много пользователей.

Например, система регистрации курсов обладает свойством параллелизма, поскольку она должна допускать одновременную работу многих пользователей (студентов и профессоров), каждый из которых порождает в системе отдельный процесс.

Процесс (process) – это ресурсоемкий поток управления, который может выполняться параллельно с другими процессами, а также в независимом адресном пространстве. В случае высокой сложности может разделяться на два потока или больше. Объект любого класса должен существовать внутри хотя бы одного процесса.

Поток (thread) – это облегченный поток управления, который может выполняться параллельно с другими потоками в рамках одного и того же процесса в общем адресном пространстве. Необходимость создания потоков в системе регистрации курсов диктуется следующими требованиями:

- если курс окажется заполненным в то время, когда студент формирует свой учебный график, включающий данный курс, то он должен быть извещен об этом (нужен независимый процесс, управляющий доступом к информации конкретных курсов);
- существующая база данных каталога курсов не обеспечивает требуемую производительность (необходим процесс промежуточной обработки – подкачки данных).

Реализация процессов и потоков обеспечивается средствами операционной системы.

Для моделирования структуры потоков управления используются так называемые активные классы – классы со стереотипами <<process>> и <<thread>>. Активный класс владеет собственным процессом или потоком и может инициировать управляющие воздействия. Связи между процессами моделируются как зависимости. Потоки могут существовать только внутри процессов, поэтому связи между процессами и потоками моделируются как композиции. Модель потоков управления помещается в пакет Process View логического представления. В качестве примера на рис. 5.6 – 5.8 приведены фрагменты диаграммы классов, описывающей структуру процесса регистрации студента на курсы. Ак-

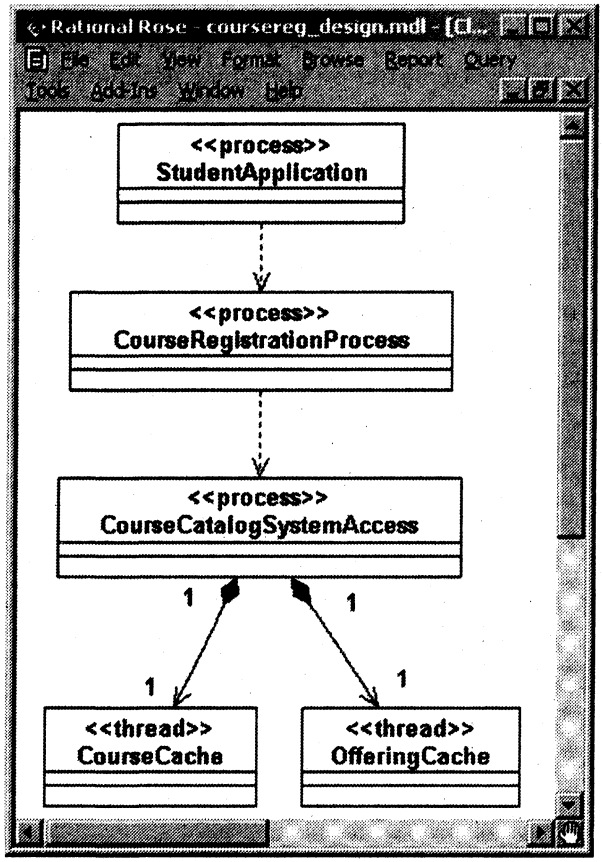


Рис. 5.6. Процессы и потоки

тивные классы, показанные на этих диаграммах, выполняют следующее назначение:

- StudentApplication – процесс, управляющий всеми функциями студента-пользователя в системе. Для каждого студента, который начинает регистрироваться на курсы, создается один объект данного класса;
- CourseRegistrationProcess – процесс, управляющий непосредственно регистрацией студента. Для каждого студента, который начинает регистрироваться на курсы, также создается один объект данного класса;

• CourseCatalogSystemAccess – управляет доступом к системе каталога курсов. Один и тот же объект данного класса используется всеми пользователями при доступе к каталогу курсов;

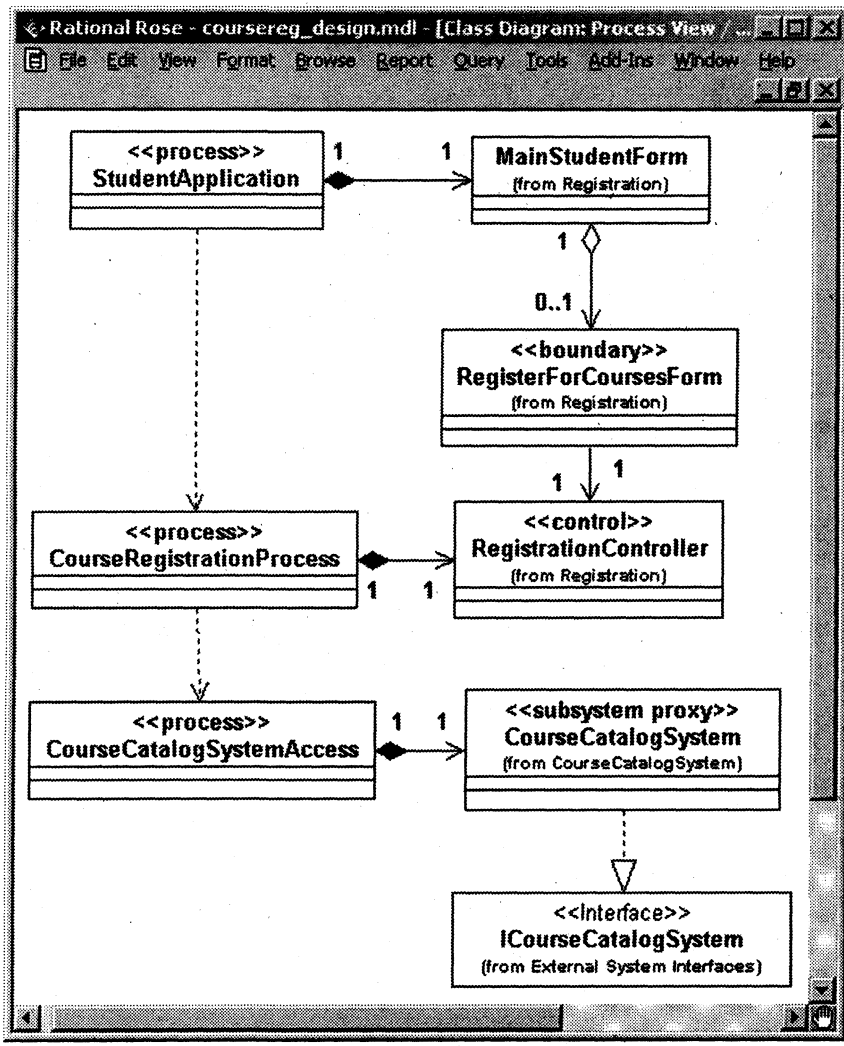


Рис. 5.7. Классы, связанные с процессами

• CourseCache и OfferingCache используются для асинхронного доступа к данным в базу данных с целью повышения производительности системы. Они представляют собой кэш для промежуточного хранения данных о курсах, извлеченных из БД.

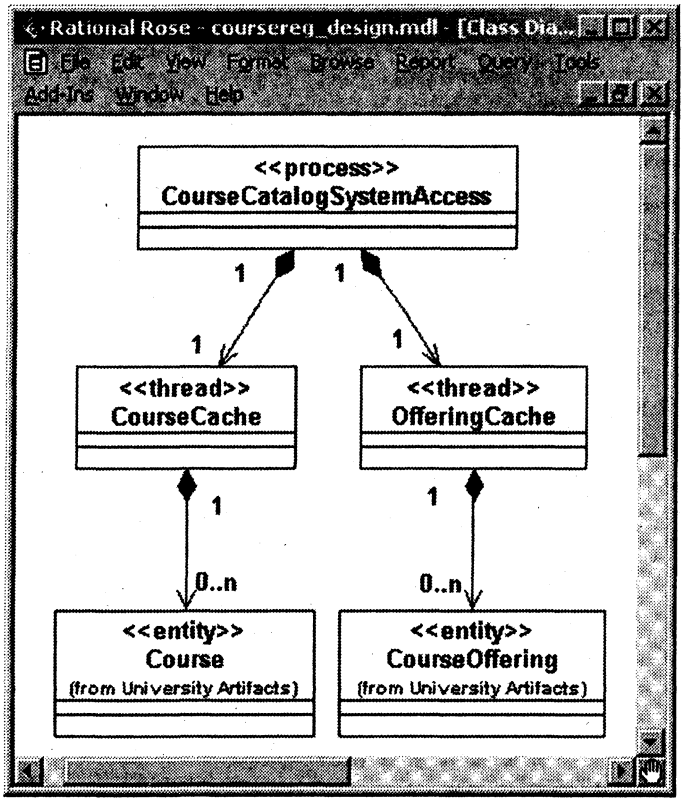


Рис. 5.8. Классы, связанные с потоками

5.1.4. МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЕННОЙ КОНФИГУРАЦИИ СИСТЕМЫ

Если создаваемая система является распределенной, необходимо спроектировать ее конфигурацию в вычислительной среде, т.е. описать вычислительные ресурсы, коммуникации между ними и использование ресурсов различными системными процес-

сами. Распределенная сетевая конфигурация системы моделируется с помощью диаграммы размещения. Ее основные элементы:

- узел (node) – вычислительный ресурс (процессор или другое устройство (дисковая память, контроллеры различных устройств и т.д.)). Для узла можно задать выполняющиеся на нем процессы;
- соединение (connection) – канал взаимодействия узлов (сеть).

Пример такой диаграммы для системы регистрации (без процессов) приведен на рис. 5.9.

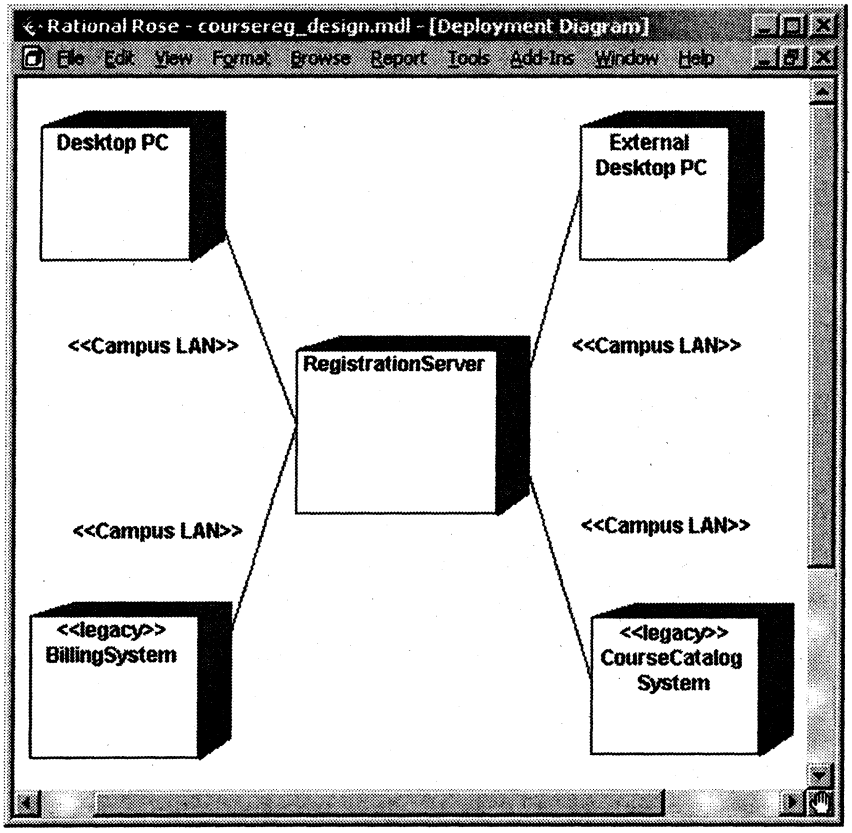


Рис. 5.9. Сетевая конфигурация системы регистрации

Распределение процессов по узлам сети (рис. 5.10) проводится с учетом следующих факторов:

- используемые образцы распределения (трехзвенная клиент-серверная конфигурация, "толстый" и "тонкий" клиенты, равноправные узлы (peer-to-peer) и т.д.);
- время отклика;
- минимизация сетевого трафика;
- мощность узла;
- надежность оборудования и коммуникаций.

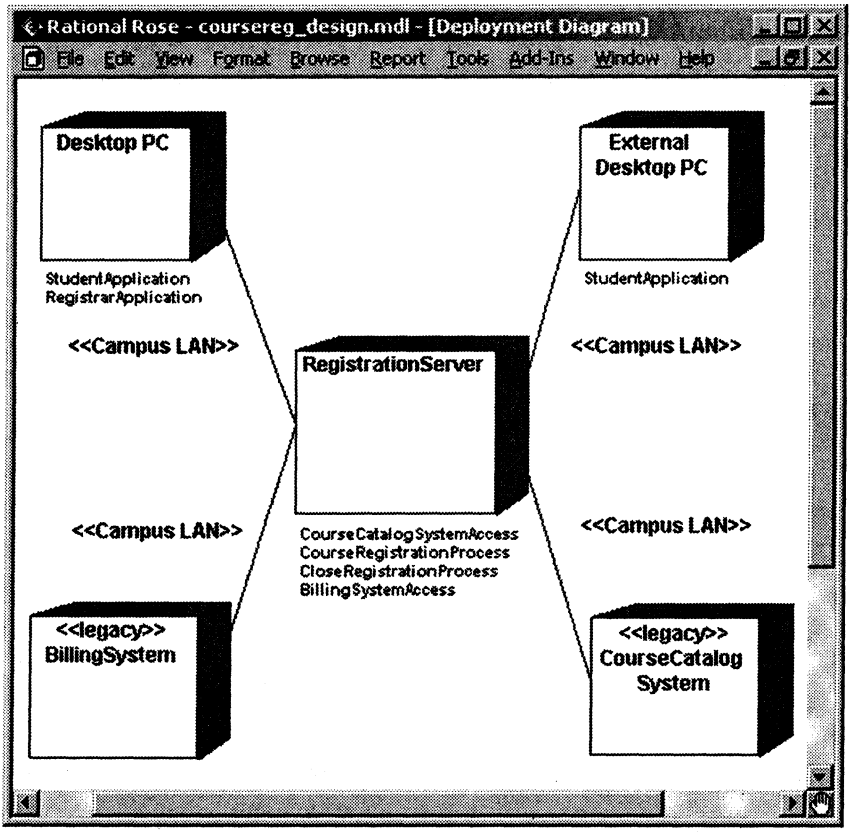


Рис. 5.10. Сетевая конфигурация системы регистрации с распределением процессов по узлам

Упражнение 5.1.

Создание диаграммы размещения системы регистрации

Для того чтобы открыть диаграмму размещения, надо дважды щелкнуть мышью по представлению Deployment View (представлении размещения) в браузере.

Для того чтобы поместить на диаграмму процессор:

1. Щелкните по кнопке на панели инструментов диаграммы Processor.
2. Щелкните по диаграмме размещения в том месте, куда хотите его поместить.
3. Введите имя процессора.

В спецификациях процессора можно ввести информацию о его стереотипе, характеристиках и планировании. Стереотипы применяются для классификации процессоров (например, компьютеров под управлением Unix или ПК).

Характеристики процессора могут, в частности, включать его скорость и объем памяти.

Поле планирования (scheduling) процессора содержит описание того, как осуществляется планирование его процессов:

1. Preemptive (с приоритетом). Высокоприоритетные процессы имеют преимущество перед низкоприоритетными.
2. Non preemptive (без приоритета). У процессов не имеется приоритета. Текущий процесс выполняется до его завершения, после чего начинается следующий.
3. Cyclic (циклический). Управление передается между процессами по кругу. Каждому процессу дается определенное время на его выполнение, затем управление переходит к следующему процессу.
4. Executive (исполнительный). Существует некий вычислительный алгоритм, который и управляет планированием процессов.
5. Manual (вручную). Процессы планируются пользователем.

Для того чтобы назначить процессору стереотип:

1. Откройте окно спецификации процессора.
2. Перейдите на вкладку "General".
3. Введите стереотип в поле Stereotype.

Для того чтобы ввести характеристики и планирование процессора:

1. Откройте окно спецификации процессора.

2. Перейдите на вкладку "Detail".
3. Введите характеристики в поле характеристик.
4. Укажите один из типов планирования.

Для того чтобы показать планирование на диаграмме:
1. Щелкните правой кнопкой мыши по процессору.
2. Выберите пункт Show Scheduling в открывшемся меню.

Для того чтобы добавить связь на диаграмму:
1. Щелкните по кнопке Connection на панели инструментов.
2. Щелкните по узлу диаграммы.
3. Проведите линию связи к другому узлу.

Для того чтобы назначить связи стереотип:
1. Откройте окно спецификации связи.
2. Перейдите на вкладку "General".
3. Введите стереотип в поле Stereotype (Стереотип).

Для того чтобы добавить процесс:
1. Щелкните правой кнопкой мыши по процессору в браузере.
2. Выберите пункт New > Process в открывшемся меню.
3. Введите имя нового процесса.

Для того чтобы показать процессы на диаграмме:
1. Щелкните правой кнопкой мыши по процессору в браузере.
2. Выберите пункт Show Processes в открывшемся меню.

5.2. ПРОЕКТИРОВАНИЕ ЭЛЕМЕНТОВ СИСТЕМЫ

Проектирование элементов системы включает:

- уточнение описания вариантов использования (заключается в модификации их диаграмм взаимодействия и диаграмм классов с учетом вновь появившихся на шаге проектирования классов и подсистем, а также проектных механизмов);
- проектирование классов;
- проектирование баз данных.

5.2.1. ПРОЕКТИРОВАНИЕ КЛАССОВ

Проектирование классов включает:

- детализацию проектных классов;
- уточнение операций и атрибутов;
- моделирование состояний для классов;
- уточнение связей между классами.

Детализация проектных классов

Каждый граничный класс преобразуется в некий набор классов в зависимости от своего назначения. Это может быть набор элементов пользовательского интерфейса, зависящий от возможностей среды разработки, или набор классов, реализующий системный или аппаратный интерфейс.

Классы-сущности с учетом соображений производительности и защиты данных могут разбиваться на ряд классов. Основанием для разбиения является наличие в классе атрибутов с различной частотой использования или видимостью. Такие атрибуты, как правило, выделяются в отдельные классы.

Управляющие классы, реализующие простую передачу информации от граничных классов к сущностям, могут быть удалены. Сохраняются классы, выполняющие существенную работу по управлению потоками событий (управление транзакциями, распределенная обработка и т.д.).

Полученные в результате уточнения классы подлежат непосредственной реализации в коде системы.

Уточнение операций и атрибутов

Обязанности классов, определенные в процессе анализа и документированные в виде операций "анализа", преобразуются в операции, которые будут реализованы в коде. При этом:

- каждой операции присваивается краткое имя, характеризующее ее результат;
- определяется полная сигнатура операции (в соответствии с нотацией, принятой в языке UML);
- создается краткое описание операции, включая смысл всех ее параметров;
- задается видимость операции: public, private или protected;
- определяется область действия операции: экземпляр (операция объекта) или классификатор (операция класса);
- может быть составлено описание алгоритма выполнения операции (с использованием диаграмм деятельности в виде блок-схем, а также диаграмм взаимодействия различных объектов при выполнении операции).

Уточнение атрибутов классов (рис. 5.11) заключается в следующем:

- кроме имени атрибута задаются его тип и значение по умолчанию (необязательное);

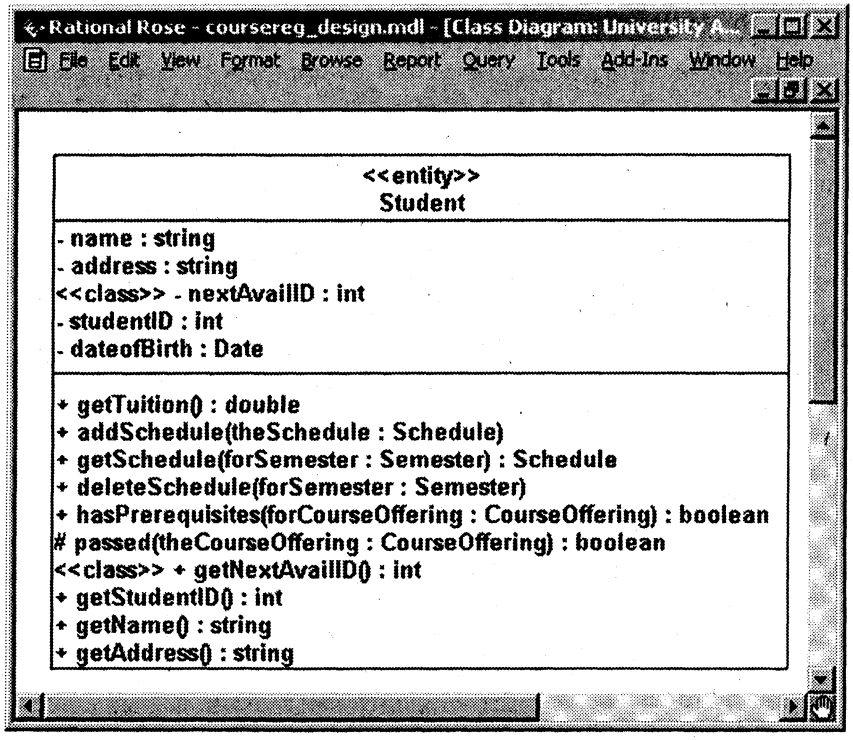


Рис. 5.11. Класс Student с полностью определенными атрибутами и операциями

- учитываются соглашения по именованию атрибутов, принятые в проекте и языке реализации;
- задается видимость атрибутов: public, private или protected;
- при необходимости определяются производные (вычисляемые) атрибуты.

Упражнение 5.2.

Определение атрибутов и операций для класса Student

Для того чтобы задать тип данных, значение по умолчанию и видимость атрибута:

1. Щелкните правой кнопкой мыши по атрибуту в браузере.
2. Выберите пункт Open Specification в открывшемся меню.

3. Укажите тип данных в раскрывающемся списке типов или введите собственный тип данных.

4. Введите значение атрибута по умолчанию в поле Initial Field (Первоначальное значение).

5. Выберите видимость атрибута: public, protected, private или implementation в поле Export Control. По умолчанию видимость всех атрибутов соответствует private.

Для того чтобы изменить нотацию для обозначения видимости:

1. Выберите пункт Tools > Options в меню модели.

2. Перейдите на вкладку "Notation".

3. Пометьте контрольный переключатель Visibility as Icons, чтобы использовать нотацию Rose, или снимите пометку, чтобы использовать нотацию UML.

Примечание. Изменение значения этого параметра приведет к смене нотации только для новых диаграмм и не затронет существующие диаграммы.

Для того чтобы задать тип возвращаемого значения, стереотип и видимость:

1. Щелкните правой кнопкой мыши по операции в браузере.

2. Откройте окно спецификации класса этой операции.

3. Укажите тип возвращаемого значения в раскрывающемся списке или введите свой тип.

4. Укажите стереотип в соответствующем раскрывающемся списке или введите новый.

5. Укажите значение видимости операции в поле Export Control: public, protected, private или implementation. По умолчанию видимость всех операций установлена в public.

Для того чтобы добавить к операции аргумент:

1. Откройте окно спецификации операции.

2. Перейдите на вкладку "Detail".

3. Щелкните правой кнопкой мыши по области аргументов, в открывшемся меню выберите Insert.

4. Введите имя аргумента.

5. Щелкните по колонке Data type и введите туда тип данных аргумента.

6. Щелкните по колонке default при необходимости и введите значение аргумента по умолчанию.

Моделирование состояний для классов

Если в системе присутствуют зависимые от состояния объекты со сложной динамикой поведения, то для них можно построить модель, описывающую состояния объектов и переходы между ними. Эта модель представляется в виде диаграмм состояний.

В качестве примера, связанного с системой регистрации, рассмотрим поведение объекта класса CourseOffering. Диаграмма состояний строится в несколько этапов.

Этап 1. Идентификация состояний. Признаками для выявления состояний являются изменение значений атрибутов объекта или создание и уничтожение связей с другими объектами. Так, объект CourseOffering может находиться в состоянии Open (прием на курс открыт) до тех пор, пока количество зарегистрировавшихся на него студентов не превышает 10, а как только оно станет равным 10, объект переходит в состоянии Closed (прием на курс закрыт). Кроме того, объект CourseOffering может находиться в состоянии Unassigned (его никто не ведет, т.е. отсутствует связь с каким-либо объектом Professor) или Assigned (такая связь существует).

Этап 2. Идентификация событий. События связаны, как правило, с выполнением некоторых операций. Так, в классе CourseOffering в результате распределения обязанностей при анализе варианта использования "Выбрать курсы для преподавания" определены две операции – addProfessor и removeProfessor, связанные с выбором курса некоторым профессором (созданием новой связи) и отказом от выбранного курса (разрывом связи). Этим операциям ставятся в соответствие два события – addProfessor и removeProfessor.

Этап 3. Идентификация переходов между состояниями. Переходы вызываются событиями. Таким образом, состояния Unassigned и Assigned соединяются двумя переходами (рис. 5.12).

Дальнейшая детализация поведения объекта CourseOffering приведет к построению диаграммы состояний (рис. 5.13). На данной диаграмме использованы такие возможности моделирования состояний, как композитные (composite state) и историческое со-

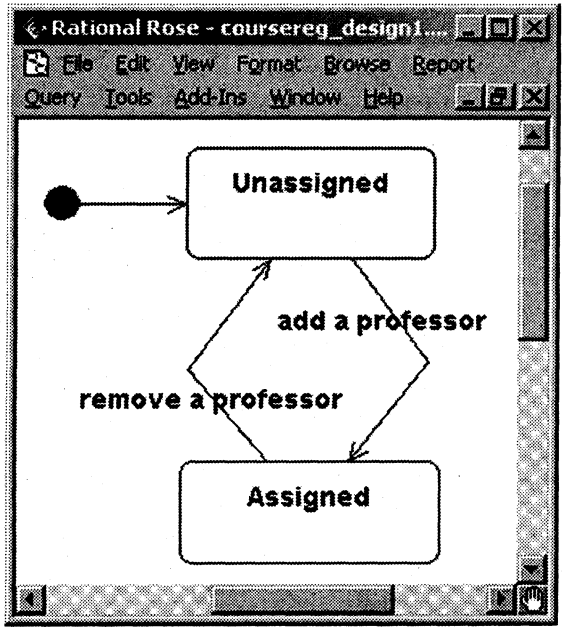


Рис. 5.12. Переходы между состояниями

стояния (history state). В данном случае композитными состояниями являются Open и Closed, а вложенными состояниями – Unassigned, Assigned, Cancelled (курс отменен), Full (курс заполнен) и Committed (курс включен в расписание). Композитные состояния позволяют упростить диаграмму, уменьшая количество переходов, поскольку вложенные состояния наследуют все свойства и переходы композитного состояния.

Историческое состояние (обозначенное на диаграмме окружностью с буквой H) – это псевдосостояние, которое восстанавливает предыдущее активное состояние в композитном состоянии. Оно позволяет композитному состоянию Open запоминать, какое из вложенных состояний (Unassigned или Assigned) было текущим в момент выхода из Open, для того чтобы любой из переходов в Open (add student или remove student) возвращался именно в это вложенное состояние, а не в начальное состояние.

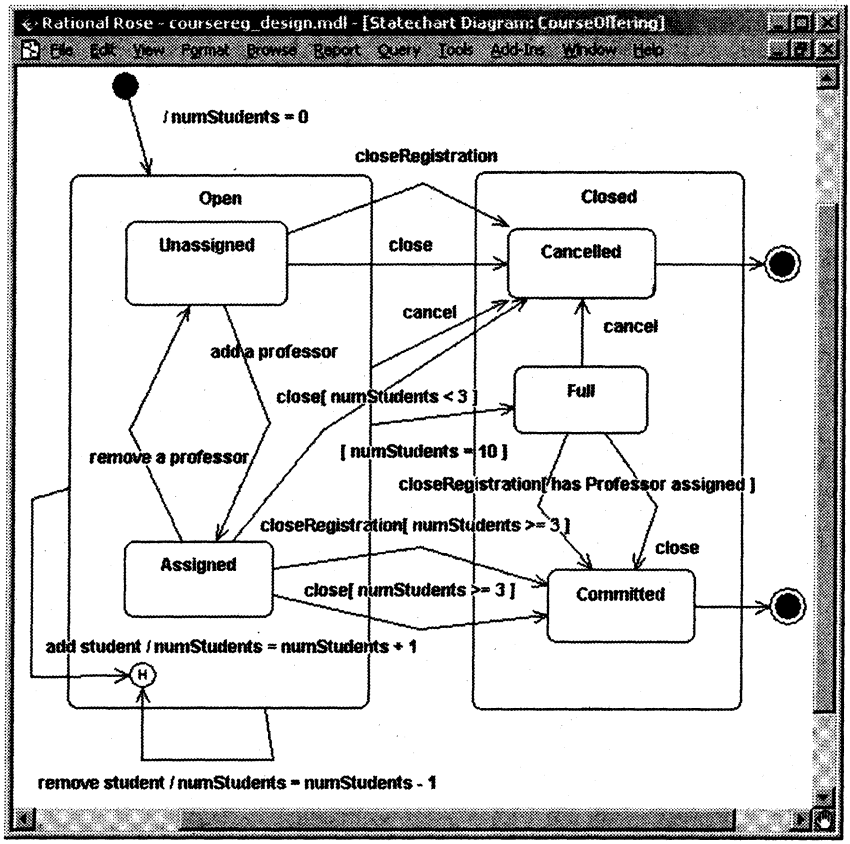


Рис. 5.13. Диаграмма состояний с композитными состояниями

Упражнение 5.3.

Создание диаграммы состояний для класса CourseOffering

Для того чтобы создать диаграммы состояний:

1. Щелкните правой кнопкой мыши по браузеру на нужном классе.

2. Выберите пункт New > Statechart Diagram в открывшемся меню.

Для того чтобы добавить состояние:

1. Щелкните по кнопке State на панели инструментов.

2. Щелкните мышью по диаграмме состояний в том месте, куда хотите поместить его.

Все элементы состояния можно добавить с помощью вкладки "Detail" окна спецификации состояния.

Для того чтобы добавить деятельность:

1. Откройте окно спецификации требуемого состояния.
2. Перейдите на вкладку "Detail".
3. Щелкните правой кнопкой мыши по окну "Actions".
4. Выберите пункт Insert в открывшемся меню.
5. Щелкните дважды по новому действию.
6. Введите действие в поле Actions.
7. Укажите Do в окне "When", чтобы сделать новое действие деятельностью.

Для того чтобы добавить входное действие, в окне "When" укажите On Entry.

Для того чтобы добавить выходное действие, в окне "When" укажите On Exit.

Для того чтобы послать событие:

1. Откройте окно спецификации требуемого состояния.
2. Перейдите на вкладку "Detail".
3. Щелкните правой кнопкой мыши по окну "Actions".
4. Выберите пункт Insert в открывшемся меню.
5. Щелкните дважды по новому действию.
6. Укажите Send Event в качестве типа действия.
7. Введите событие (event), аргументы (arguments) и целевой объект (Target) в соответствующие поля.

Для того чтобы добавить переход:

1. Щелкните по кнопке Transition панели инструментов.
2. Щелкните мышью по состоянию, откуда осуществляется переход.
3. Проведите линию перехода до того состояния, где он завершается.

Для того чтобы добавить рефлексивный переход:

1. Щелкните по кнопке Transition to Self панели инструментов.
2. Щелкните по тому состоянию, где осуществляется рефлексивный переход.

Для того чтобы добавить событие, его аргументы, ограждающее условие и действие:

1. Щелкните дважды по переходу, чтобы открыть окно его спецификации.

2. Перейдите на вкладку "General".
3. Введите событие в поле Event.
4. Введите аргументы в поле Arguments.
5. Введите ограждающее условие в поле Condition.
6. Введите действие в поле Action.

Для того чтобы отправить событие:

1. Щелкните дважды по переходу, чтобы открыть окно его спецификации.
2. Перейдите на вкладку "Detail"..
3. Введите событие в поле Send Event.
4. Введите аргументы в поле Send Arguments.
5. Задайте цель в поле Send Target.

Для того чтобы указать начальное или конечное состояние:

1. Нажмите кнопку Start State или End State на панели инструментов.
2. Щелкните мышью по диаграмме состояний в том месте, куда хотите поместить состояние.

Уточнение связей между классами

В процессе проектирования связи между классами (ассоциации, агрегации и обобщения) подлежат уточнению:

- ассоциации между граничными и управляющими классами отражают связи, динамически возникающие между соответствующими объектами в потоке управления. Для таких связей достаточно обеспечить видимость классов, поэтому они преобразуются в зависимости;
- если для некоторых ассоциаций нет необходимости в двунаправленной связи, то вводятся направления навигации;
- агрегации, обладающие свойствами композиции, преобразуются в связи композиции.

Пример преобразования связей в соответствии с данными рекомендациями для классов варианта использования "Зарегистрироваться на курсы" приведен на рис. 5.14. Ассоциация между управляющим и граничным классами преобразована в зависимость. Агрегация между классами Student и Schedule обладает свойствами композиции. Направления навигации на ассоциациях между классами Schedule и CourseOffering введены по следующим соображениям: нет необходимости в получении списка графиков, в которых присутствует какой-либо курс; количество графиков

относительно мало по сравнению с количеством конкретных курсов.

Для того чтобы преобразовать агрегацию в композицию:

1. Щелкните правой кнопкой мыши по тому концу агрегации, который упирается в класс-часть (на рис.5.14 – Schedule).
2. Выберите пункт Containment в открывшемся меню.
3. Укажите метод включения By Value.

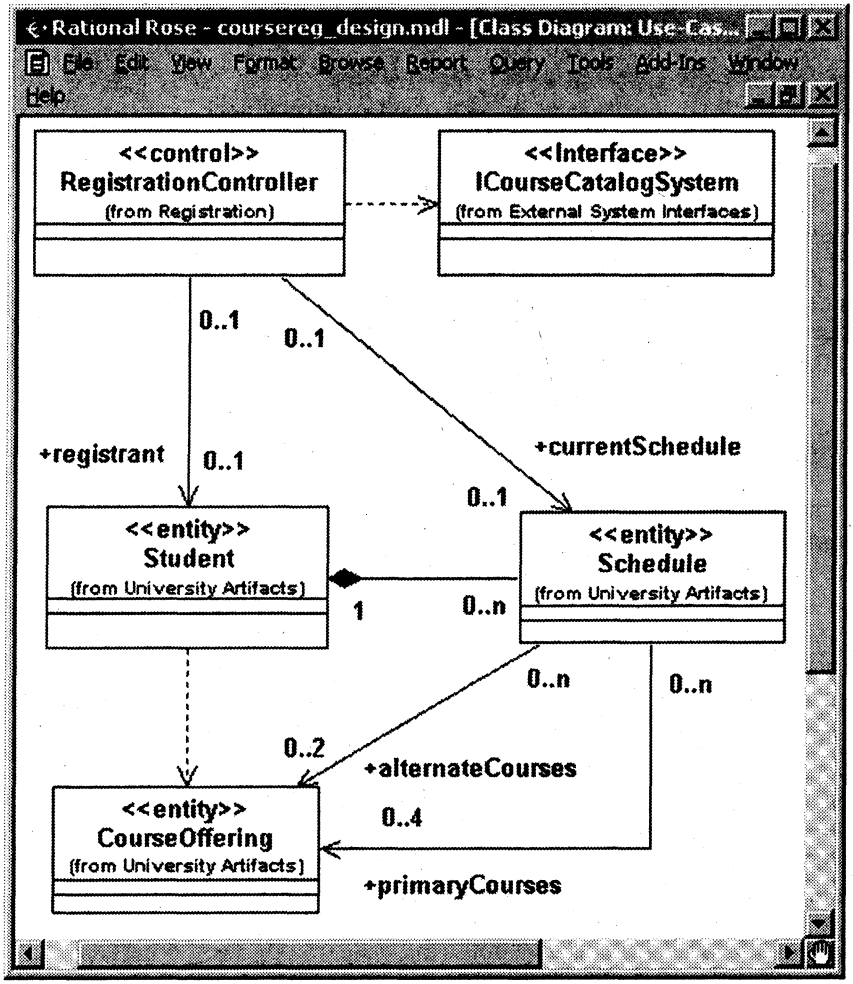


Рис. 5.14. Пример преобразования ассоциаций и агрегаций

Примечание. Значение By Value предполагает, что целое и часть создаются и разрушаются одновременно, что соответствует композиции. Агрегация (By Reference) предполагает, что целое и часть создаются и разрушаются в разное время.

Связи обобщения могут преобразовываться в ситуациях с так называемой метаморфозой подтипов. Например, в случае с системой регистрации студент может переходить с очной формы

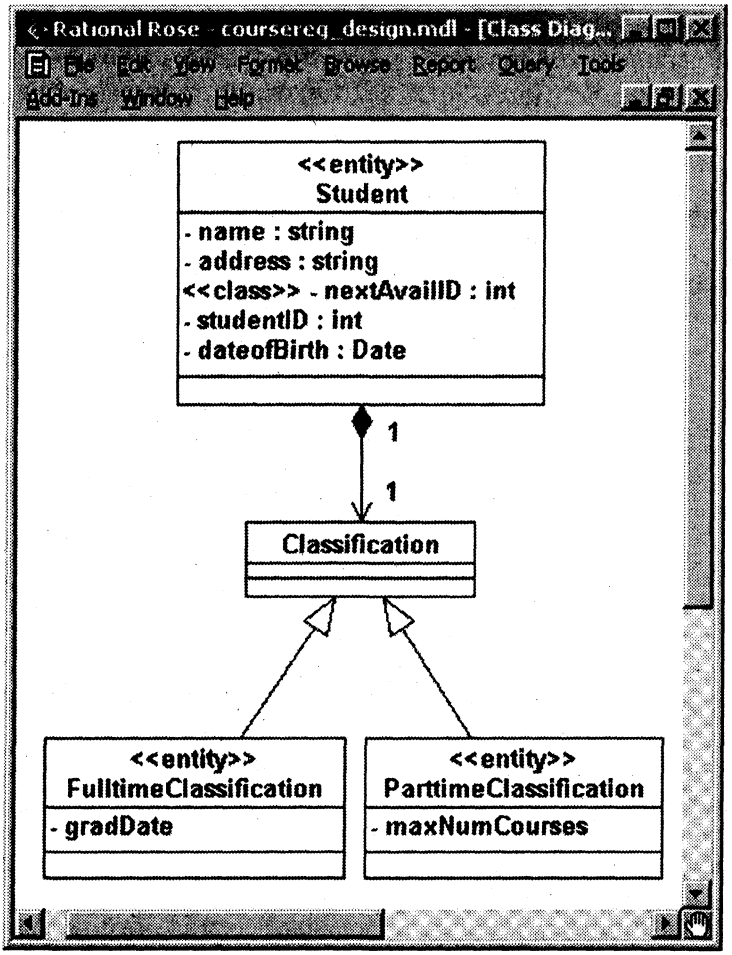


Рис. 5.15. Преобразование обобщения

обучения на вечернюю, т.е. объект Student может менять свой подтип. При таком изменении придется модифицировать описание объекта в системе. Для того чтобы избежать этой модификации и тем самым повысить устойчивость системы, иерархия наследования реализуется с помощью классификации (рис. 5.15).

5.2.2. ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Проектирование базы данных зависит от того, какой тип СУБД используется для хранения данных СУБД – объектная или реляционная. Для объектных БД никакого проектирования не требуется, поскольку классы-сущности непосредственно отображаются в БД. Для реляционных БД классы-сущности объектной модели должны быть отображены в таблицы реляционной БД. Совокупность таблиц и связей между ними может быть представлена в виде диаграммы классов, которая, по существу, является ER-диаграммой. Набор правил, применяемых при отображении классов в таблицы БД, фактически совпадает с правилами преобразования сущностей и связей в таблицы БД. В технологии RUP для такого отображения используется специальный инструмент – Data Modeler. Он выполняет преобразование классов-сущностей в классы-таблицы с последующей генерацией описания БД как на стандартном языке SQL (ANSI SQL), так и на его диалектах для различных СУБД (Oracle, IBM DB2, Sybase Adaptive Server, MS SQL Server). Для описания схемы БД применяется следующий набор элементов языка UML со своими стереотипами (профиль UML):

- таблица представляется в виде класса со стереотипом <<Table>>;
- представление изображается в виде класса со стереотипом <<View>>;
- столбец таблицы представляется в виде атрибута класса с соответствующим типом данных;
- обычная ассоциация и агрегация представляются в виде ассоциации со стереотипом <<Non-Identifying>> (в терминологии IDEF1X – неидентифицирующей связи);
- композиция представляется в виде ассоциации со стереотипом <<Identifying>> (в терминологии IDEF1X – идентифицирующей связи);

• схема БД представляется в виде пакета со стереотипом <<Schema>>, содержащего классы-таблицы;

• контейнер хранимых процедур представляется в виде класса со стереотипом <<SP Container>>;

• ограничения целостности, индексы и триггеры представляются в виде операций классов-таблиц со стереотипами <<PK>> (Primary key), <<FK>> (Foreign key), <<Unique>>, <<Check>>, <<Index>> и <<Trigger>>;

• физическая база данных представляется в виде компонента со стереотипом <<Database>>.

Упражнение 5.4.

Проектирование реляционной базы данных

Проектирование БД состоит из следующих шагов.

Шаг 1. Создание нового компонента – базы данных:

1. Щелкните правой кнопкой мыши по представлению компонентов.

2. Выберите пункт Data Modeler > New > Database в открывшемся меню.

3. Откройте окно спецификации вновь созданного компонента DB_0 и в списке Target выберите Oracle 8.x.

Шаг 2. Определение устойчивых (persistent) классов:

1. Откройте окно спецификации класса Student в пакете University Artifacts.

2. Перейдите на вкладку "Detail".

3. Установите значение переключателя Persistence в Persistent.

4. Проделайте такие же действия для классов Classification, FulltimeClassification и ParttimeClassification.

5. Откройте класс Student в браузере, нажав "+".

6. Щелкните правой кнопкой мыши по атрибуту studentID.

7. Выберите пункт Data Modeler > Part of Object Identity в открывшемся меню (указание атрибута в качестве части первичного ключа).

Шаг 3. Создание схемы БД:

1. Щелкните правой кнопкой мыши по пакету University Artifacts.

2. Выберите пункт Data Modeler > Transform to Data Model в открывшемся меню.

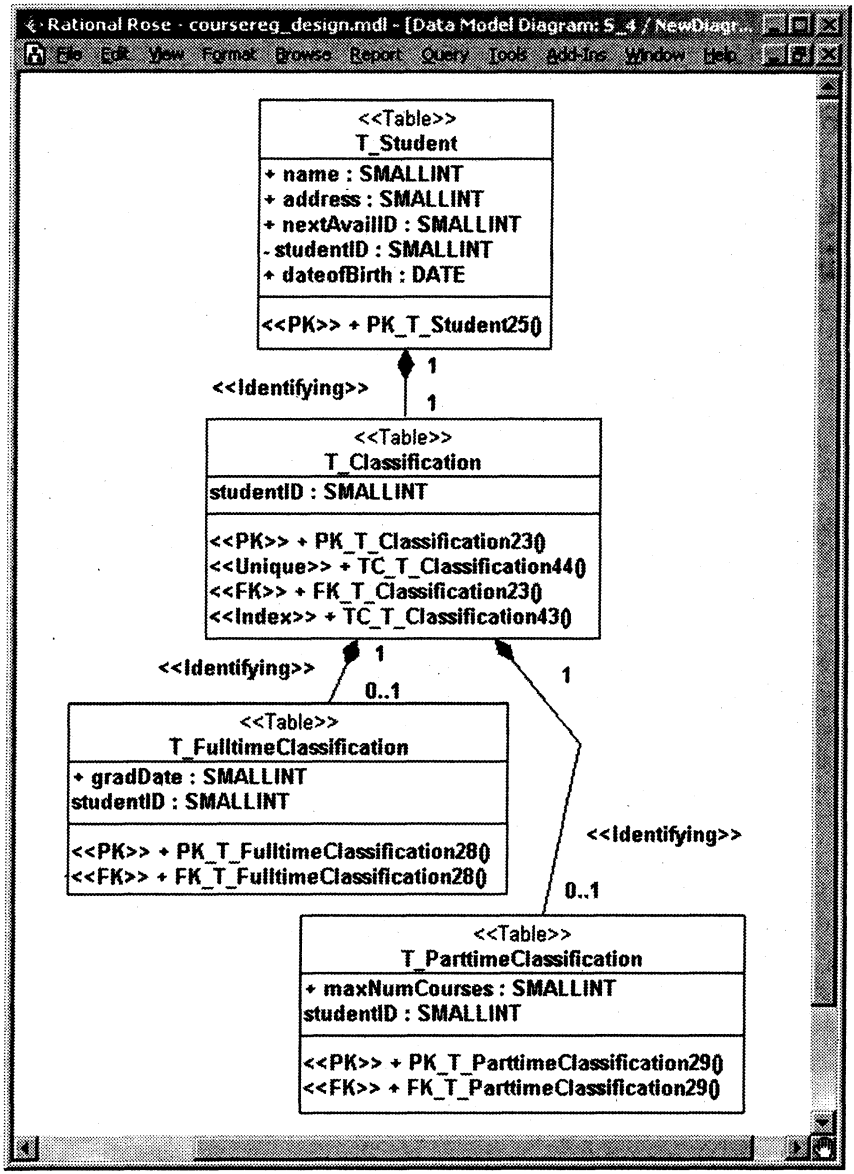


Рис. 5.16. Диаграмма "сущность-связь"

3. Укажите DB_0 и щелкните по кнопке OK в появившемся окне в списке Target Database. В результате в логическом представлении появится новый пакет Schemas.

4. Откройте пакет Schemas и щелкните правой кнопкой мыши по пакету <<Schema>> S_0.

5. Выберите пункт Data Modeler > New > Data Model Diagram в открывшемся меню.

6. Откройте пакет, затем вновь созданную диаграмму "сущность-связь" NewDiagram и перенесите на нее все классы-таблицы, находящиеся в пакете <<Schema>> S_0. Получившаяся диаграмма показана на рис. 5.16.

Глава 6

ГЕНЕРАЦИЯ КОДА

6.1. ГЕНЕРАЦИЯ ОПИСАНИЯ БАЗЫ ДАННЫХ НА ЯЗЫКЕ SQL

После завершения проектирования БД можно сгенерировать описание базы данных на языке SQL.

Упражнение 6.1.
Генерация описания базы данных

Для генерации описания БД:
1. Щелкните правой кнопкой мыши по пакету <<Schema>> S_0.

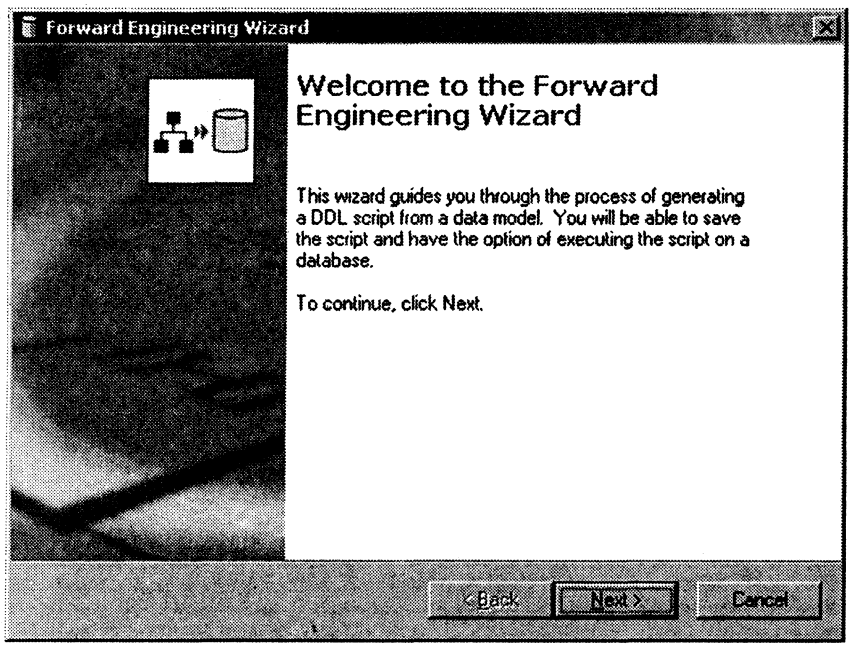


Рис. 6.1. Окно мастера "Forward Engineering Wizard"

2. Выберите пункт Data Modeler > Forward Engineer в открывшемся меню.

3. Щелкните по кнопке Next в открывшемся окне мастера "Forward Engineering Wizard" (рис. 6.1).

4. Оставьте все флажки генерации языка описания данных (DDL) отмеченными (рис. 6.2) и щелкните по кнопке Next.

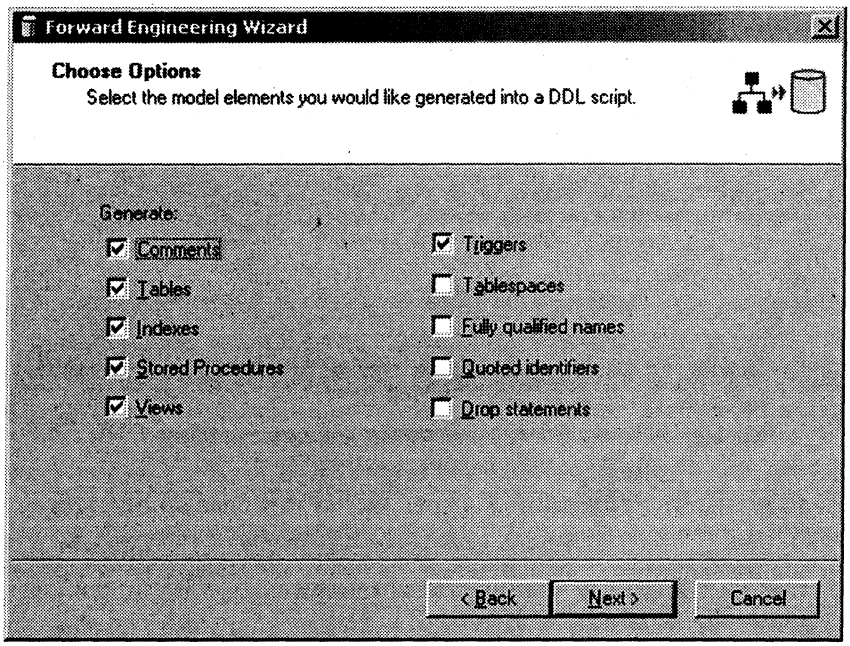


Рис. 6.2. Параметры генерации DDL

5. Укажите имя и расположение текстового файла с результатами генерации (рис. 6.3) и щелкните по кнопке Next.

6. Завершив генерацию, откройте созданный текстовый файл и просмотрите результаты.

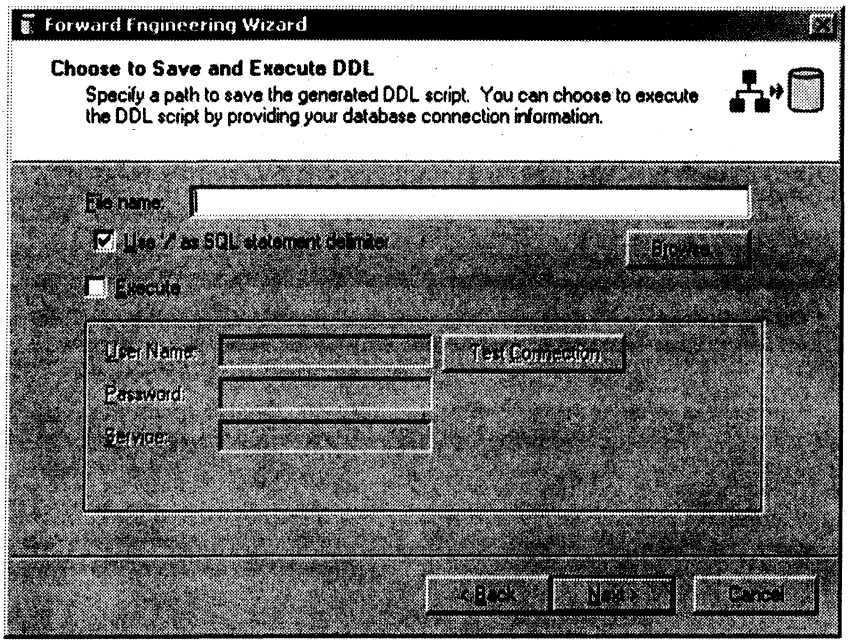


Рис. 6.3. Задание имени файла с результатами генерации DDL

Результаты генерации должны выглядеть следующим образом:

```
CREATE TABLE T_ParttimeClassification (
maxNumCourses SMALLINT NOT NULL,
studentID SMALLINT NOT NULL,
CONSTRAINT PK_T_ParttimeClassification29 PRIMARY KEY
(studentID)
)
/
CREATE TABLE T_Classification (
studentID SMALLINT NOT NULL,
CONSTRAINT PK_T_Classification23 PRIMARY KEY
(studentID),
CONSTRAINT TC_T_Classification44 UNIQUE (studentID)
)
/
```

```
    CREATE TABLE T_FulltimeClassification (
    gradDate SMALLINT NOT NULL,
    studentID SMALLINT NOT NULL,
    CONSTRAINT PK_T_FulltimeClassification28 PRIMARY KEY
(studentID)
    )
    /
    CREATE TABLE T_Student (
    name SMALLINT NOT NULL,
    address SMALLINT NOT NULL,
    nextAvailID SMALLINT NOT NULL,
    studentID SMALLINT NOT NULL,
    dateofBirth DATE NOT NULL,
    CONSTRAINT PK_T_Student25 PRIMARY KEY (studentID)
    )
    /
    CREATE INDEX TC_T_Classification43 ON T_Classification
(studentID)
    /
    ALTER TABLE T_ParttimeClassification ADD (CONSTRAINT
FK_T_ParttimeClassification29 FOREIGN KEY (studentID) REFER-
ENCES T_Classification (studentID))
    /
    ALTER TABLE T_Classification ADD (CONSTRAINT
FK_T_Classification23 FOREIGN KEY (studentID) REFERENCES
T_Student (studentID))
    /
    ALTER TABLE T_FulltimeClassification ADD (CONSTRAINT
FK_T_FulltimeClassification28 FOREIGN KEY (studentID) REFER-
ENCES T_Classification (studentID))
    /
```

6.2. ГЕНЕРАЦИЯ КОДА ПРИЛОЖЕНИЯ

Процесс генерации кода приложения состоит из следующих шагов:

Шаг 1. Проверка модели.

Шаг 2. Создание компонентов.

Шаг 3. Соотнесение классов с компонентами.

Шаг 4. Установка свойств генерации кода.

Шаг 5. Выбор класса, компонента или пакета.
Шаг 6. Генерация кода.

Упражнение 6.2.
Проверка модели

В Rose существует средство проверки моделей, не зависящее от языка и применяемое для обеспечения корректности модели перед генерацией кода. Рекомендуется всегда выполнять такую проверку, поскольку она помогает выявить в модели неточности и ошибки, не позволяющие генерировать код надлежащим образом.

Для проверки модели Rose:

1. Выберите в меню Tools > Check Model.
2. Проанализируйте все ошибки, появившиеся в окне журнала.

К наиболее распространенным ошибкам относятся, например, сообщения на диаграмме последовательности или кооперативной диаграмме, не отображенные на операцию, либо объекты этих диаграмм, не отображенные на класс.

С помощью пункта меню Check Model можно выявить большую часть неточностей и ошибок в модели. Пункт меню Access Violations позволяет обнаруживать нарушения правил доступа, возникающие тогда, когда существует связь между двумя классами разных пакетов, но связи между самими пакетами нет.

Для того чтобы обнаружить нарушение правил доступа:

1. Выберите в меню Report > Show Access Violations.
2. Проанализируйте все нарушения правил доступа в окне.

Упражнение 6.3.
Создание компонентов и соотнесение классов с компонентами

В Rose диаграммы компонентов создаются в представлении компонентов (рис. 6.4). Отдельные компоненты можно создавать непосредственно на диаграмме или перетаскивать их туда из браузера.

Выберем в качестве языка программирования C++ и создадим для класса Student соответствующие этому языку компоненты.

Для создания диаграммы компонентов:

1. Щелкните мышью дважды по главной диаграмме компонентов Main в представлении компонентов.

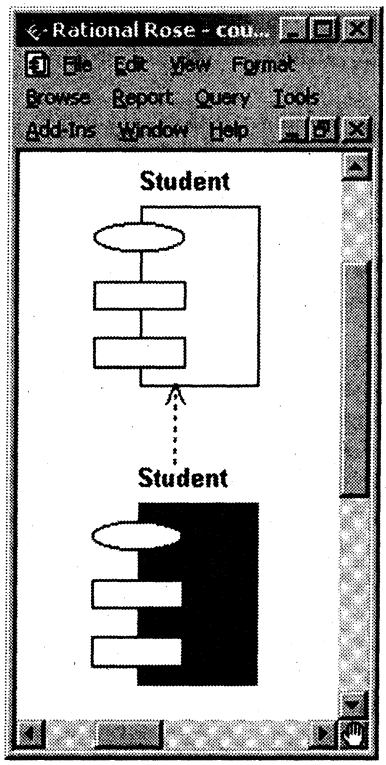


Рис. 6.4. Диаграмма компонентов

2. Щелкните по кнопке Package Specification на панели инструментов.

3. Поместите элемент Package Specification на диаграмму.

4. Откройте спецификацию элемента, введите имя Student и укажите в окне языка язык ANSI C++.

5. Щелкните по кнопке Package Body на панели инструментов.

6. Поместите элемент Package Body на диаграмму.

7. Откройте спецификацию элемента, введите имя Student и укажите в окне языка язык ANSI C++.

8. Щелкните по кнопке Dependency на панели инструментов.

9. Проведите линию зависимости от тела пакета к спецификации пакета.

Для соотнесения класса с компонентами:

1. Найдите класс Student в логическом представлении браузера.

2. Перетащите этот класс на спецификацию пакета компонента Student в представлении компонентов браузера. В результате класс Student будет соотнесен со спецификацией и телом пакета компонента Student.

Упражнение 6.4.
Генерация кода

Для каждого языка в Rose предусмотрен ряд определенных свойств генерации кода. Перед генерацией кода рекомендуется анализировать эти свойства и вносить необходимые изменения.

Для анализа свойств генерации кода выберите Tools > Options, а затем вкладку соответствующего языка. В окне списка можно выбрать класс, атрибут, операцию и другие элементы модели. Для каждого языка в этом списке указаны свои собственные элементы модели. При выборе разных значений на экране появляются разные наборы свойств. Любые изменения, вносимые в набор свойств в окне Tools > Options, воздействуют на все элементы модели, для которых используется данный набор*.

Во время генерации кода Rose выбирает информацию из логического и компонентного представлений модели и генерирует большой объем "скелетного" (skeletal) кода:

- **Классы.** Генерируются все классы модели.
- **Атрибуты.** Код включает атрибуты каждого класса, в том числе видимость, тип данных и значение по умолчанию.
- **Сигнатуры операций.** Код содержит определения операций со всеми параметрами, типами данных параметров и типом возвращаемого значения операции.
- **Связи.** Некоторые из связей модели вызывают создание атрибутов при генерации кода.
- **Компоненты.** Каждый компонент реализуется в виде соответствующего файла с исходным кодом.

Для генерации кода C++:

1. Откройте диаграмму компонентов системы.

2. Выберите все объекты на диаграмме компонентов.

---

* Более подробно генерация кода средствами Rose рассмотрена в книге: Боггс У., Боггс М. UML и Rational Rose 2002: Пер. с англ. – М.: ЛОРИ, 2004.

3. Выберите Tools > ANSI C++ > Generate Code в меню.
4. Выберите каталог для генерации кода.
5. Щелкните по кнопке OK и выполните генерацию в окне генерации кода.
6. Просмотрите результаты генерации (меню Tools > C++ > Browse Header и Tools > C++ > Browse Body).

Результаты генерации должны выглядеть следующим образом.

Файл Student.h:

```
#ifndef STUDENT_H_HEADER_INCLUDED_BE29599B
#define STUDENT_H_HEADER_INCLUDED_BE29599B

//##ModelId=35A6336C03DE
class Student
{
public:
//##ModelId=360EBEFA015E
double getTuition();

//##ModelId=374AFB93006B
addSchedule(Schedule theSchedule);

//##ModelId=374AFBDA0117
Schedule getSchedule(Semester forSemester);

//##ModelId=374B00540183
deleteSchedule(Semester forSemester);

//##ModelId=374B02690049
boolean hasPrerequisites(CourseOffering forCourseOffering);

//##ModelId=37812F3903C4
int getNextAvailID();

//##ModelId=379779F60364
int getStudentID();

//##ModelId=37BE859E00CA
string getName();
```

```
//##ModelId=37BF215800D8
string getAddress();

protected:
//##ModelId=37812764010E
boolean passed(CourseOffering theCourseOffering);

private:
//##ModelId=35E9A8EA00BE
string name;

//##ModelId=374D92DE019A
string address;

//##ModelId=37812F2301D8
int nextAvailID;

//##ModelId=378130B900EB
int studentID;

//##ModelId=378BE6A5015D
Date dateofBirth;

};

#endif /* STUDENT_H_HEADER_INCLUDED_BE29599B */
```

Файл Student.cpp:

```
#include "Student.h"

//##ModelId=360EBEFA015E
double Student::getTuition()
{
}

//##ModelId=374AFB93006B
Student::addSchedule(Schedule theSchedule)
{
}
```

```
//##ModelId=374AFBDA0117
Schedule Student::getSchedule(Semester forSemester)
{
}

//##ModelId=374B00540183
Student::deleteSchedule(Semester forSemester)
{
}

//##ModelId=374B02690049
boolean Student::hasPrerequisites(CourseOffering forCourseOffering)
{
}

//##ModelId=37812F3903C4
int Student::getNextAvailID()
{
}

//##ModelId=379779F60364
int Student::getStudentID()
{
}

//##ModelId=37BE859E00CA
string Student::getName()
{
}

//##ModelId=37BF215800D8
string Student::getAddress()
{
}

//##ModelId=37812764010E
boolean Student::passed(CourseOffering theCourseOffering)
{
}
```

1. ДОКУМЕНТЫ СИСТЕМЫ РЕГИСТРАЦИИ КУРСОВ

КОНЦЕПЦИЯ

1. Введение

1.1. Цель

Цель этого документа – определить требования высокого уровня к системе регистрации курсов (СРК) в терминах потребностей конечных пользователей.

1.2. Область применения

В данном документе рассматривается система СРК, разрабатываемая информационной службой университета. Клиент-серверная система СРК должна взаимодействовать с существующей базой данных каталога курсов.

Система регистрации курсов позволит студентам регистрироваться на курсы в онлайновом режиме, а профессорам – указывать курсы, которые они будут читать, и ставить оценки за курсы.

1.3. Определения, акронимы и сокращения

См. Глоссарий.

2. Основные положения

2.1. Возможности системы

Новый проект полностью заменит внешний интерфейс существующей системы регистрации курсов на современную диалоговую систему, которая позволит студентам и профессорам осуществлять доступ с персональных компьютеров.

Существующая система регистрации используется с 1985 г. и обладает недостаточной пропускной способностью. Кроме того, уста-

ревшая технология универсальной ЭВМ позволяет поддерживать доступ только через сотрудника деканата. Новая система позволит всем профессорам и студентам получить доступ к системе через ПК, связанные с университетом компьютерной сетью, и через любой персональный компьютер, подключенный к интернету.

Новая система обеспечит университету ведущую позицию в области компьютерных систем регистрации курсов и, таким образом, улучшит его репутацию, привлечет больше студентов и упростит процесс администрирования.

2.2. Формулировка проблемы

Проблема	Устаревший и в значительной степени неавтоматизированный процесс регистрации студентов в университете
Затрагивает	Студентов, профессоров и администрацию университета
Последствия	Медленный и дорогостоящий процесс регистрации, неудовлетворенность студентов и профессоров
Успешное решение позволит	Улучшить репутацию университета, привлечь больше студентов и упростить процесс администрирования

2.3. Формула продукта

Для	Студентов университета, профессоров и регистратора курсов
Которые	Посещают, преподают или администрируют курсы университета
СРК является	Инструментом
Который	Обеспечит диалоговую регистрацию на курсы и доступ к информации о курсах и оценках
В отличие от	Существующей устаревшей системы регистрации на базе универсальной ЭВМ
Наш продукт	Обеспечит получение своевременной информации относительно всех курсов, регистрации, преподавателей и оценок для всех пользователей любых ПК через ЛВС университета или интернет

3. Описание заинтересованных лиц и пользователей

В этом разделе описаны три существующих типа пользователей СРК: регистратор курсов, студенты и профессора.

3.1. Потенциальные потребители

Университетское сообщество пользователей – большое и сложное сообщество, которому требуются гибкие и высокопроизводительные системы диалоговой регистрации на курсы.

Пользователи обладают высоким уровнем образования, компьютерной грамотности и в большинстве случаев имеют домашние персональные компьютеры. Возможность регистрироваться на курсы посредством персональных компьютеров и просматривать свои оценки существенно упростила бы процесс регистрации. Студенты и профессора имеют свободный доступ к локальной сети университета через персональные компьютеры, расположенные в библиотеке университетского городка и студенческом корпусе.

Распространение первой версии СРК будет ограничено самим университетом. Возможность распространения последующих версий для других школ и университетов будет рассмотрена информационной службой университета. В этом случае СРК будет разработана с учетом возможности ее конфигурирования в процессе установки.

3.2. Заинтересованные лица

Наименование	Представляет	Роль
Директор информационной службы	Информационную службу и университет в целом	Отвечает за финансирование проекта и контроль за ходом работ
Регистратор (сотрудник деканата)	Службу регистрации, административный персонал и персонал, занимающийся подготовкой данных	Удостоверяет, что система соответствует потребностям регистратора, которые заключаются в управлении данными регистрации курса, включая базы данных профессоров и студентов
Студент	Студентов	Удостоверяет, что система соответствует потребностям студентов
Профессор	Профессоров	Представляет интересы факультета (профессоров)

3.3. Пользователи

Наименование	Описание
Регистратор	Управляет базой данных профессоров и студентов, открывает и закрывает процесс регистрации
Студент	Регистрируется на курсы, запрашивает оценки и другую информацию о курсах
Профессор	Выбирает курсы для преподавания. Ставит оценки студентам

3.4. Пользовательская среда

Пользовательская среда включает персональные компьютеры, Windows XP/2000.

3.5. Основные потребности заинтересованных лиц/пользователей

Группа представителей, включающая студентов, профессоров и регистратора, составила обзор, содержащий проблемы, которые возникают у пользователей существующей системы регистрации курса, и предложения по ее усовершенствованию. Резюме результатов обзора приведено ниже (в порядке их относительной важности):

Потребность	Приоритет	Проблема	Существующее решение	Предлагаемые решения
Регистрация студентов на курсы	Высокий	Регистрация студентов на курсы медленная и неэффективная	В настоящее время студенты должны заполнить форму регистрации на курсы и представить ее регистратору. Регистратору требуется около двух недель, чтобы обработать форму, и еще неделя, чтобы послать подтверждение студенту. Любые изменения в расписании из-за заполненности курсов или из-	Студенты хотели бы иметь диалоговый доступ к системе, чтобы быстро определять доступность курсов и назначенных профессоров

Продолжение

Потребность	Приоритет	Проблема	Существующее решение	Предлагаемые решения
			менений в предпочтениях студентов требуют, чтобы трехнедельный процесс был повторен. Это ограничивает возможности выбора курсов	
Быстрый доступ к оценкам студентов	Средний	Большие затраты времени на получение информации об оценках, постоянные обращения к профессорам	Заполненные табели успеваемости обычно отправляют студентам спустя восемь недель после начала экзаменационного периода. В это время студенты непрерывно звонят своим профессорам, пытаясь узнать оценки как можно быстрее	Диалоговый доступ к индивидуальным оценкам был рекомендован большинством студентов, участвовавших в обзоре
Низкие накладные расходы	Средний	Организационная работа занимает много времени и дорогостояща	Регистратору и двум-трем временным служащим требуется 400–500 ч в каждом семестре, чтобы обработать документы регистрации курсов. Бóльшая часть этого времени тратится на ввод информации в главную базу данных регистрации курсов и перерегистрацию студентов на другие курсы при разрешении конфликтов графика и проблем доступности курсов	Доступ студентов к системе регистрации курсов мог бы исключить эти расходы

4. Обзор продукта

Данный раздел содержит общее описание возможностей СРК, ее внешних интерфейсов с расчетной системой и базой данных каталога курсов, а также конфигурации.

4.1. Перспективы продукта

Система регистрации курсов заменит существующую систему регистрации курсов университета на базе универсальной ЭВМ. Новая система будет взаимодействовать с существующей расчетной системой и базой данных каталога курсов.

Система регистрации курсов будет состоять из клиентского и серверного компонентов.

Компонент клиента размещается на персональном компьютере и будет установлен на всех ПК университета. Любой ПК, который находится вне университета, должен загрузить программное обеспечение клиента с Unix-сервера через интернет. Как только компонент клиента будет установлен на ПК, пользователь может получить доступ к СРК через локальную сеть университета или интернет. Для доступа необходимо получить идентификатор и пароль.

Компонент сервера размещается на Unix-сервере университета и должен взаимодействовать с расчетной системой и базой данных каталога курсов на универсальной ЭВМ университета.

4.2. Возможности продукта

Ниже приведены основные возможности СРК в терминах ее свойств и достоинств с точки зрения потребителей. Свойства системы описаны в разд. 5.

Достоинство	Свойство системы
Своевременная информация о курсах	Система имеет доступ к базе данных каталога курса для получения своевременной информации относительно всех курсов, предлагаемых в университете. Студенты и профессора могут просмотреть описание каждого курса, требования к предшествующим курсам, назначенных преподавателей, местоположение классов и время занятий

Продолжение

Достоинство	Свойство системы
Своевременная информация о регистрации	Все регистрации на курсы немедленно фиксируются в базе данных регистрации, чтобы обеспечить своевременную информацию относительно заполненных или отмененных курсов
Простой и своевременный доступ к оценкам за курсы	Студенты могут просматривать свои оценки за любой курс, указывая свои идентификатор и пароль. Студенты могут получить доступ к системе регистрации с любого ПК университета или с домашнего ПК через интернет. Профессора вводят все оценки непосредственно в базу данных регистрации со своих ПК
Доступ с любого ПК университета	Студенты могут получить доступ к системе регистрации с любого ПК университета или со своего домашнего ПК через интернет. Установку компонента клиента системы регистрации на ПК легко выполнить через интернет
Простой и удобный доступ с домашнего ПК	Студенты могут получить доступ к системе регистрации с любого ПК университета или со своего домашнего ПК через интернет
Безопасность и конфиденциальность	Для доступа к системе регистрации необходимы пользовательский идентификатор и пароль. Студенческий табель успеваемости защищен от несанкционированного доступа
Мгновенное информирование о заполненных или отмененных курсах	Все регистрации на курсы немедленно фиксируются в базе данных регистрации, чтобы обеспечить своевременную информацию относительно заполненных или отмененных курсов

4.3. Проектные ограничения

Существующие расчетная система и база данных каталога курсов, которые размещаются на универсальной ЭВМ университета, будут поддерживаться по крайней мере до 2008 г.

Внешние интерфейсы расчетной системы и базы данных каталога курсов не будут изменены.

Предполагается, что университет продолжит использовать и поддерживать существующий Unix-сервер и универсальную ЭВМ по крайней мере до 2008 г.

Предполагается, что к 2008 г. будет получено дополнительное финансирование для замены расчетной системы и базы данных каталога курсов.

Реализация новой системы регистрации, запланированная на январь 2007 г., зависит от того, будет ли утверждено финансирование к 1 марта 2006 г.

4.4. Стоимость проекта

Затраты на разработку системы не должны превышать 1 200 000 долл. в связи с тем, что финансирование ограниченно.

Ожидается, что будут использоваться существующие компьютеры университета и не потребуется затрат на приобретение аппаратных средств.

4.5. Лицензирование и установка

К версии 1.0 системы не предъявляется никаких требований лицензирования, поскольку она будет использоваться только в университете.

Установка клиентского компонента должна быть доступна с дискеты, компакт-диска или из интернета.

Установка серверного компонента должна обеспечить варианты сохранения существующей базы данных регистрации (без потери данных) или генерации новой базы данных.

5. Функциональные возможности продукта

В данном разделе определены и описаны высокоуровневые функциональные возможности (свойства) системы регистрации, которые необходимы с точки зрения пользователей.

5.1. Вход в систему

Студенты, профессора и регистратор должны располагать идентификатором и паролем для входа в систему регистрации. На период приема в университет пользователи получают идентификатор и временный пароль. Система должна позволять пользователю изменить свой временный пароль.

5.2. Регистрация на курсы

Система должна выводить по запросу студента доступные курсы. Студент может указывать в запросе наименование курса, код курса и кафедру (факультет). Система должна регистрировать студентов на курсы в соответствии с их доступностью, конфликтами графика и предварительным прохождением необходимых курсов. Если регистрация на курс невозможна, система должна немедленно уведомить студента об этом.

Система должна позволять студенту изменить свой выбор курсов до окончания периода регистрации.

5.3. Отмена курсов

Система должна позволять регистратору отменять курсы. Регистратор обычно рассматривает все курсы в конце периода регистрации и отменяет курсы, которые никто не ведет или на которые записалось менее трех студентов. Регистратор уведомляет студентов об отмене курсов по телефону или почте.

5.4. Расчеты за курсы

Система СРК должна посылать уведомления расчетной системе после закрытия регистрации. Эти уведомления должны включать имя студента, его адрес, информацию о выбранных курсах и размер оплаты.

5.5. Ввод, обновление и просмотр информации о профессорах

Система должна вводить и обновлять информацию о профессорах, включая имя, адрес, телефон, факс и адрес электронной почты. Информация о профессорах должна быть доступна профессорам и регистратору для просмотра.

5.6. Просмотр оценок студентов

Система должна позволять студенту просматривать оценки по отдельным курсам или полный табель успеваемости. Система должна защищать информацию об оценках студента от доступа любого другого пользователя, кроме самого студента и профессоров.

5.7. Выбор курсов для преподавания

Система должна позволять профессорам выбирать курсы для преподавания до окончания периода регистрации.

5.8. Ввод, обновление и просмотр информации о студентах

Система должна вводить и обновлять информацию о студентах, включая студенческий ID, имя, адрес, телефон и адрес электронной почты. Информация о студентах должна быть доступна профессорам и регистратору для просмотра. Система должна гарантировать, что только студент имеет доступ к собственной информации. Регистратор ведет информацию о студентах.

5.9. Запись оценок студентов

Система должна сохранять оценки, введенные профессором.

5.10. Просмотр каталога курсов

Информация каталога курсов, содержащаяся в базе данных, должна выводиться по запросу пользователя, включающему наименование и код курса, имя профессора и наименование кафедры.

5.11. Просмотр графика курсов

Система должна выводить полный график курсов для конкретного студента по его запросу.

5.12. Мониторинг заполнения курсов

Система должна гарантировать, что ни на один курс не будет записано более 10 студентов.

6. Ограничения

Список, приведенный в подразд. 4.3, дополняют следующие ограничения:

- система не должна требовать никакого развития или приобретения аппаратных средств;
- доступная информация о курсах ограничена данными, поддерживаемыми в существующей базе данных каталога курсов.

7. Требования к качеству

В данном разделе определены требования к производительности, надежности, удобству использования и другим характеристикам качества системы регистрации.

Готовность: работоспособное состояние 24 ч в день 7 дней в неделю.

Удобство использования: должна быть простой в использовании для студентов и профессоров, а также включать диалоговую помощь и не требовать использования твердой копии руководств.

Сопровождаемость: должна обеспечивать простоту сопровождения. Все данные университета должны изменяться без перекомпиляции системы.

8. Приоритеты

Данный раздел определяет относительную важность предлагаемых функциональных возможностей системы, которые должны быть включены в первые два выпуска системы. Реализация всех возможностей, касающихся регистрации студентов на курсы, должна быть запланирована в выпуске 1.

Ожидается, что система регистрации начнет эксплуатироваться в университете через два-четыре основных выпуска.

Выпуск 1 должен содержать как минимум следующие основные функциональные возможности:

- вход в систему;
- регистрацию на курсы;
- интерфейс с базой данных каталога курсов;
- ведение информации о студентах;
- ведение информации о профессорах.

Выпуск 2 должен включать:

- проставление оценок студентам;
- просмотр оценок;
- выбор курсов для преподавания.

Функциональные возможности для выпуска 3 еще не определены. Ожидается, что этот выпуск будет содержать расширение существующих функциональных возможностей.

Замена существующих расчетной системы и базы данных каталога курсов намечается в выпуске 4 в 2008 г.

9. Прочие требования к продукту

9.1. Используемые стандарты

Пользовательский интерфейс должен быть Windows XP/2000-совместимым.

9.2. Системные требования

В системе должны быть предусмотрены интерфейсы с существующей базой данных каталога курсов и расчетной системой.

Серверный компонент системы должен функционировать на сервере университетского городка под управлением Unix.

Клиентский компонент системы должен функционировать на любом персональном компьютере под управлением Windows XP/2000.

Клиентский компонент системы не должен требовать дисковой памяти больше, чем 256 Мбайт RAM и 20 Мбайт.

9.3. Требования к производительности

Система должна поддерживать до 2000 пользователей, работающих с центральной базой данных в любое время, и до 500 пользователей, одновременно работающих с локальными серверами в любой момент.

Система должна обеспечивать доступ к базе данных каталога курсов в течение не более 10 с.

Система должна завершать 80 % всех транзакций в течение 2 мин.

9.4. Требования к окружающей среде

Отсутствуют.

10. Требования к документации

10.1. Руководство пользователя

Руководство пользователя должно описывать использование системы с точки зрения студентов, профессоров и регистратора и включать:

- минимальные системные требования;
- установку ПК-клиента;
- вход в систему;
- выход из системы;
- все функциональные возможности системы;
- информацию о поддержке пользователей.

Руководство пользователя должно иметь объем от 50 до 100 страниц с размером страницы 7×9 дюймов. Руководство пользователя должно быть доступно как в твердой копии, так и в онлайновом режиме.

10.2. Диалоговая помощь

Диалоговая помощь должна быть доступна пользователю для каждой функции системы.

10.3. Руководство по установке, конфигурированию и файл Read Me

Руководство по установке для серверной части должно включать:

- минимальные системные требования;
- инструкции по установке;
- конфигурирование под конкретные параметры университета;
- инициализацию базы данных регистрации;
- сохранение существующей базы данных регистрации;
- информацию о поддержке пользователей;
- информацию о получении обновлений.

Файл Read Me должен быть доступен во время установки в любое время и размещаться на диске. Файл Read Me должен включать:

- возможности нового выпуска;
- сведения об устраненных дефектах.

10.4. Маркировка и упаковка

Пользовательская документация и экранные заставки должны содержать логотип университета.

Поскольку начальные выпуски предназначены только для университета, маркетинговая литература и рекламные материалы разрабатываться не будут.

Вариант использования "Выбрать курсы для преподавания":

Краткое описание:

Данный вариант использования позволяет профессору выбрать из каталога курсов курсы, которые он желает вести в предстоящем семестре.

Основной поток событий:

Данный вариант использования начинает выполняться, когда профессор выбирает из каталога курсов курсы, которые он желает вести в предстоящем семестре.

1. Система выполняет поиск и выводит список предлагаемых курсов, которые профессор может вести в текущем семестре, а также список курсов, которые профессор выбрал ранее.
2. Профессор выбирает и/или отменяет курсы, которые он желает вести в предстоящем семестре.
3. Система удаляет связь между профессором и отмененным курсом.
4. Система подтверждает отсутствие конфликтов между выбранными курсами (совпадение даты и времени) и обновляет информацию для каждого выбранного курса (включает данные профессора).

Альтернативные потоки:

Отсутствуют доступные курсы:

Если во время выполнения основного потока обнаружится, что профессор не может вести никаких курсов в предстоящем семестре, система выводит сообщение об ошибке. Когда профессор подтверждает прием сообщения, выполнение варианта использования завершается.

Конфликт графика:

Если система обнаружит конфликт графика, будет выведено сообщение об ошибке. Система должна также указать, какие курсы конфликтуют. Профессор может либо разрешить конфликт (например, отменив выбор одного из курсов), либо отменить всю операцию, при этом информация о выборе не сохраняется и выполнение варианта использования завершается.

Недоступен каталог курсов:

Если система не может установить связь с каталогом курсов, выводится сообщение об ошибке. Когда профессор подтверждает

прием сообщения, выполнение варианта использования завершается.

Регистрация на курсы закончена:

Если в самом начале варианта использования выясняется, что регистрация на текущий семестр закончена, выводится сообщение и выполнение варианта использования завершается. После завершения регистрации профессор не может вносить изменения в курсы. Если такая необходимость возникнет, она реализуется за пределами данной системы.

Специальные требования:

Отсутствуют.

Предусловия:

Перед началом выполнения данного варианта использования профессор должен войти в систему.

Вариант использования "Проставить оценки"

Краткое описание:

Данный вариант использования позволяет профессору проставить оценки студенту за один курс или больше курсов, прослушанных в течение семестра.

Основной поток событий:

Данный вариант использования начинает выполняться, когда профессор хочет проставить студенту оценки за один курс или больше курсов, прослушанных в течение семестра.

1. Система выводит список курсов, которые профессор вел в течение семестра.
2. Профессор выбирает курс.
3. Система выполняет поиск и выводит список всех студентов, зарегистрировавшихся на данный курс. Для каждого студента выводятся все оценки, полученные ранее.
4. Для каждого студента в списке профессор вводит оценку: 2, 3, 4 или 5. Система записывает оценку. Если профессор хочет пропустить какого-либо студента, поле информации об оценке может быть оставлено незаполненным и будет заполнено позднее. Профессор также может изменить оценку и ввести новую оценку.

Альтернативные потоки:

Отсутствуют курсы:

Если во время выполнения основного потока обнаружится, что профессор не вел никаких курсов в данном семестре, система выводит сообщение об ошибке. Когда профессор подтверждает прием сообщения, выполнение варианта использования завершается.

Специальные требования:

Отсутствуют.

Предусловия:

Перед началом выполнения данного варианта использования профессор должен войти в систему.

Вариант использования "Просмотреть табель успеваемости"

Краткое описание:

Данный вариант использования позволяет студенту просмотреть свой табель успеваемости после завершения семестра.

Основной поток событий:

Данный вариант использования начинает выполняться, когда студент хочет просмотреть свой табель успеваемости после завершения семестра.

1. Система выполняет поиск и отображение информации об оценках за каждый из курсов, оконченных студентом в течение семестра.
2. Когда студент указывает, что он завершил просмотр оценок, выполнение варианта использования завершается.

Альтернативные потоки:

Информация об оценках недоступна:

Если во время выполнения основного потока система не может найти какую-либо информацию об оценках, выводится соответствующее сообщение. Когда студент подтверждает прием сообщения, выполнение варианта использования завершается.

Специальные требования:

Отсутствуют.

Предусловия:

Перед началом выполнения данного варианта использования студент должен войти в систему.

Вариант использования "Вести информацию о профессорах"

Краткое описание:

Данный вариант использования позволяет регистратору вести информацию о профессорах (добавлять, удалять и изменять данные).

Основной поток событий:

Данный вариант использования начинает выполняться, когда регистратор хочет добавить, удалить и/или изменить данные о профессорах.

1. Система запрашивает требуемое действие (создать, обновить или удалить данные о профессорах).
2. Когда регистратор указывает действие, выполняется один из подчиненных потоков.

Добавить данные профессора:

1. Система запрашивает у регистратора ввод информации о профессоре. Она включает: имя, дату рождения, номер страхового свидетельства, наименование должности, кафедры.
2. После ввода требуемой информации система формирует уникальный идентификационный номер профессора и сохраняет данные о нем.

Обновить данные профессора:

1. Система запрашивает у регистратора ввод идентификатора профессора.
2. Регистратор вводит идентификатор. Система выполняет поиск и выводит информацию о профессоре.
3. Регистратор выполняет требуемые изменения.
4. Система обновляет данные о профессоре.

Удалить данные о профессоре:

1. Система запрашивает у регистратора ввод идентификатора профессора.

2. Регистратор вводит идентификатор. Система выполняет поиск и выводит информацию о профессоре.

3. Система запрашивает у регистратора подтверждение того, что информация о профессоре удалена.

4. Регистратор подтверждает удаление

5. Система удаляет данные о профессоре.

Альтернативные потоки:

Профессор не найден:

Если во время выполнения подчиненных потоков "Обновить данные профессора" или "Удалить данные о профессоре" выясняется, что профессора с заданным идентификатором не существует, то выдается сообщение об ошибке. Регистратор может ввести другой идентификатор либо отменить операцию, после чего вариант использования завершится.

Профессор уже существует:

Если во время выполнения подчиненного потока "Добавить данные профессора" обнаруживается, что профессор с таким именем уже существует, то выдается сообщение об ошибке. Регистратор может изменить имя, ввести данные нового профессора с таким же именем или отменить операцию, после чего вариант использования завершится.

Специальные требования:

Отсутствуют.

Предусловия:

Перед началом выполнения данного варианта использования регистратор должен войти в систему.

Вариант использования "Вести информацию о студентах"

Краткое описание:

Данный вариант использования позволяет регистратору вести информацию о студентах (добавлять, удалять и изменять данные).

Основной поток событий:

Данный вариант использования начинает выполняться, когда регистратор хочет добавить, удалить и/или изменить данные о студентах.

1. Система запрашивает требуемое действие (создать, обновить или удалить данные о студентах).

2. Когда регистратор указывает действие, выполняется один из подчиненных потоков.

Добавить данные студента:

1. Система запрашивает у регистратора ввод информации о студенте. Она включает: имя, дату рождения, номер страхового свидетельства, дату поступления.

2. После ввода требуемой информации система формирует уникальный идентификационный номер студента и сохраняет данные о нем.

Обновить данные студента:

1. Система запрашивает у регистратора ввод идентификатора студента.

2. Регистратор вводит идентификатор. Система выполняет поиск и выводит информацию о студенте.

3. Регистратор выполняет требуемые изменения.

4. Система обновляет данные о студенте.

Удалить данные о студенте:

1. Система запрашивает у регистратора ввод идентификатора студента.

2. Регистратор вводит идентификатор. Система выполняет поиск и выводит информацию о студенте.

3. Система запрашивает у регистратора подтверждение того, что информация о студенте удалена.

4. Регистратор подтверждает удаление

5. Система удаляет данные о студенте.

Альтернативные потоки:

Студент не найден:

Если во время выполнения подчиненных потоков "Обновить данные студента" или "Удалить данные о студенте" обнаружится, что студента с заданным идентификатором не существует, то выдается сообщение об ошибке. Регистратор может ввести другой идентификатор либо отменить операцию, после чего вариант использования завершится.

Студент уже существует:

Если во время выполнения подчиненного потока "Добавить данные студента" обнаружится, что студент с таким именем уже сущест-

вует, то выдается сообщение об ошибке. Регистратор может изменить имя, ввести данные нового студента с таким же именем или отменить операцию, после чего вариант использования завершится.

Специальные требования:

Отсутствуют.

Предусловия:

Перед началом выполнения данного варианта использования регистратор должен войти в систему.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО КУРСОВОМУ ПРОЕКТИРОВАНИЮ

ТЕМАТИКА КУРСОВОГО ПРОЕКТА И ЗАДАНИЯ ПО ЕГО ВЫПОЛНЕНИЮ

Тематика курсового проекта ориентирована на разработку системы для решения конкретных прикладных задач в заданной предметной области.

Основное задание курсового проекта – это построение моделей ПО с помощью инструментального средства Rational Rose. Процесс создания модели состоит из нескольких этапов.

Этап 1. Составление глоссария проекта.

Этап 2. Создание модели вариантов использования.

Этап 3. Анализ вариантов использования.

Этап 4. Проектирование системы.

Процесс создания модели должен проходить так, как это описано в настоящем практикуме. Структура модели в браузере Rose должна соответствовать структуре, предусмотренной технологией Rational Unified Process.

После выполнения третьего этапа модель должна удовлетворять следующим требованиям:

- глоссарий проекта должен иметь вид таблицы и храниться в отдельном файле;
- на диаграммах вариантов использования каждое действующее лицо и вариант использования должны сопровождаться описанием.

Описание действующего лица должно коротко (в одну-две строки) характеризовать роль данного лица. Описание варианта использования должно включать пояснение, предусловие, потоки событий (основной и альтернативные, если таковые имеются) и постусловие. Описания представляют собой либо присоединенные текстовые файлы, либо текст, введенный в поле Documentation спецификации соответствующего элемента диаграммы;

• диаграммы взаимодействия, соответствующие потокам событий вариантов использования, должны содержать необходимые пояснения.

При проектировании системы требуется:

• создать иерархию классов системы;

• разместить классы по пакетам (в зависимости от постановки задачи);

• связать объекты с классами, сообщения на диаграммах взаимодействия – с операциями;

• сопроводить кратким описанием каждый класс (обязанности класса), описанием атрибутов в виде таблицы (имя, описание, тип) и таблицей с описанием операций (имя, описание, сигнатура);

• указать стереотипы для классов;

• построить диаграммы классов системы, отображающие связи между классами;

• построить диаграммы состояний для описания поведения объектов отдельных классов;

• разработать (если требуется) схему базы данных и отобразить ее на диаграмме "сущность-связь".

Следует также разработать диаграмму размещения. В зависимости от варианта задания диаграмма размещения должна показывать расположение компонентов в распределенном приложении.

Структура курсового проекта

Курсовой проект должен состоять из четырех глав и заключения.

Первая глава "Постановка задачи" должна содержать формулировку задания.

Вторая глава "Анализ требований" должна содержать глоссарий, диаграмму вариантов использования, описания действующих лиц и вариантов использования.

Третья глава "Анализ системы" должна содержать диаграммы взаимодействия между объектами (последовательности и коопера-

тивные), соответствующие потокам событий вариантов использования. При необходимости можно включить диаграммы деятельности и сопроводить их пояснениями, указывающими, какому потоку событий они соответствуют (если это не ясно из их названия), и комментариями.

Четвертая глава "Проектирование" должна содержать иерархию классов системы и описание пакетов. Для каждого класса системы дается описание, которое включает: ответственность класса, описание атрибутов в виде таблицы из трех столбцов: имя, описание, тип; таблицу с описанием операций (имя, описание, сигнатура). Должны быть приведены диаграммы классов системы, отображающие связи между классами, и диаграммы состояний, описывающие поведение экземпляров отдельных классов. Также приводится диаграмма размещения с комментариями. Если вариант предполагает создание схемы базы данных, то такая схема также должна быть включена в отчет.

В заключение должен быть подведен итог и оценены результаты работы.

3. ТЕМЫ КУРСОВОГО ПРОЕКТА

Система начисления заработной платы

Перед информационной службой компании поставлена задача создания новой системы начисления заработной платы взамен морально устаревшей существующей системы. Новая система должна предоставлять служащим возможность записывать электронным способом информацию из карточки учета рабочего времени и автоматически формировать чеки на оплату, учитывающие количество отработанных часов и общий объем продаж (для служащих, получающих комиссионное вознаграждение).

Новая система должна предоставлять служащим возможность вводить информацию из карточки учета рабочего времени и заказы на поставку, изменять свои параметры (такие, как способ оплаты за работу) и формировать различные отчеты. Система должна работать на персональных компьютерах служащих всей компании. В целях

обеспечения безопасности и аудита служащие должны иметь возможность доступа и редактирования только своих собственных карточек учета рабочего времени и заказов на поставку.

В системе должна храниться информация обо всех служащих компании, в том числе работающих в различных странах. Система должна обеспечивать правильную и своевременную оплату труда каждого служащего в соответствии с указанным им способом. Компания из соображений экономии расходов желает сохранить без изменений одну из существующих баз данных (БД управления проектами), которая содержит всю информацию относительно проектов и тарифов. База данных управления проектами функционирует в среде DB2 на мейнфрейме IBM. Новая система может читать данные из БД управления проектами, но не может обновлять их.

Некоторые служащие получают почасовую заработную плату, которая начисляется на основе карточек учета рабочего времени, каждая из которых содержит дату и количество часов, отработанных в соответствии с конкретным тарифом. Если какой-либо служащий отработал в день больше 8 ч, сверхурочное время оплачивается с коэффициентом 1,5. Служащие-почасовики получают заработную плату каждую пятницу.

Некоторые служащие получают фиксированный оклад, однако они тоже представляют свои карточки учета рабочего времени. Благодаря этому система может вести учет количества часов, отработанных в соответствии с конкретными тарифами. Такие служащие получают заработную плату в последний рабочий день месяца.

Некоторые из служащих, получающих фиксированный оклад, также получают комиссионное вознаграждение, учитывающее объем продаж. Они представляют заказы на поставку, отражающие дату и объем продаж. Процент комиссионного вознаграждения определяется индивидуально для каждого служащего и может составлять 10%, 15, 25 или 35%.

Одной из наиболее часто используемых возможностей новой системы является формирование различных отчетов: запросить количество отработанных часов, суммарную заработную плату, оставшееся время отпуска и т.д.

Служащие могут выбирать способ оплаты за работу и получать свои чеки на оплату по почте, на счет в банке или на руки в офисе.

Администратор системы ведет информацию о служащих. В его обязанности входят ввод данных о новых служащих, удаление данных и изменение любой информации о служащем, такой, как имя,

адрес и способ оплаты, а также формирование различных отчетов для руководства.

Приложение начисления заработной платы запускается автоматически каждую пятницу и в последний рабочий день месяца для расчета в эти дни заработной платы соответствующих служащих. Начисление заработной платы должно проводиться автоматически, без ручного вмешательства.

Интернет-магазин

Производитель компьютеров предлагает приобретать свою продукцию через интернет. Клиент может выбрать компьютер на Web-странице производителя. Компьютеры подразделяются на серверы, настольные и портативные. Заказчик может выбрать стандартную конфигурацию или построить требуемую в диалоговом режиме. Компоненты конфигурации (такие, как оперативная память) представляются в виде списка для выбора из доступных альтернатив. Для каждой новой конфигурации система может подсчитать цену.

Для того чтобы оформить заказ, клиент должен заполнить информацию по доставке и оплате. В качестве платежных средств допускается использование кредитных карточек или чеков. После ввода заказа система отправляет клиенту по электронной почте сообщение, содержащее подтверждение получения заказа вместе с относящимися к нему деталями. Пока клиент ожидает доставку компьютера, он может проверить состояние заказа в любое время в диалоговом режиме. Серверная часть обработки заказа состоит из заданий, необходимых для проверки кредитоспособности и способа расчета клиента за покупку, истребования заказанной конфигурации со склада, печати счета и подачи заявки о доставке компьютера клиенту на склад.

Дополнительные требования:

- для знакомства со стандартной конфигурацией выбираемого сервера, настольного или портативного компьютера клиент использует Web-страницу интернет-магазина. При этом также приводится цена конфигурации;
- клиент выбирает детали конфигурации, с которыми он хочет ознакомиться и, возможно, купить готовую или составить более подходящую конфигурацию. Цена для каждой конфигурации может быть подсчитана по требованию пользователя;

• клиент может выбрать вариант заказа компьютера по интернету либо попросить, чтобы продавец связался с ним для объяснения деталей заказа, договорился о цене и тому подобном, прежде чем заказ будет фактически размещен;

• для размещения заказа клиент должен заполнить электронную форму с адресами для доставки товара и отправки счет-фактуры, а также деталями, касающимися оплаты (кредитная карточка или чек);

• после ввода заказа клиента в систему продавец отправляет на склад электронное требование, содержащее подробное описание заказанной конфигурации;

• детали сделки, включая номера заказа и счета клиента, отправляются по электронной почте клиенту, так что заказчик может проверить состояние заказа по интернету;

• склад получает счет-фактуру от продавца и отгружает компьютер клиенту.

Магазин проката видеопродукции

Магазин проката видеопродукции нуждается в компьютеризованной системе учета, так как его ассортимент составляют около 1000 видеокассет и 500 видеодисков. Запас уже заказан у поставщика, однако директор намерен прибегать к услугам большего числа поставщиков. Все видеокассеты и диски снабжены штрих-кодом, так что сканер, интегрированный в систему, может поддерживать операции выдачи напрокат и возврата видеофильмов. Членские карточки клиентов также снабжены штрих-кодом.

Клиенты имеют возможность резервировать видео таким образом, чтобы комплект видеофильмов был собран к определенной дате. Система должна обладать поисковым механизмом для ответов на запросы клиентов, включая вопросы, касающиеся фильмов, которых нет в ассортименте магазина (но которые он может заказать по просьбе клиента).

Для каждого фильма установлен конкретный период проката (исчисляемый в днях) с соответствующей платой за прокат за этот период.

Видеомагазин должен быть в состоянии немедленно дать ответ на любой запрос по наличию фильмов в запасе, а также количеству кассет или дисков (текущие условия по каждой ленте и диску должны быть известны и зафиксированы).

Плата за прокат отличается в зависимости от видеоносителя: кассета или диск.

Хотя магазин держит в запасе видеодиски только одного формата DVD, пользователи желали бы расширить в будущем систему проката и на диски других форматов.

Работники видеомагазина стремятся запомнить коды наиболее популярных лент. Зачастую при идентификации фильма они используют именно код фильма, а не его название (поскольку фильм с одним названием мог выпускаться разными режиссерами).

Дополнительные требования:

• за кассеты и диски, возвращенные позже указанного срока, взимается дополнительная плата за период, превышающий срок проката. Каждый видеоноситель обладает уникальным идентификационным номером.

Фильмы заказываются у поставщика, который может поставить кассеты и диски в течение недели. Обычно делается один заказ на несколько фильмов.

Забронировать можно те фильмы, которые заказаны у поставщика и/или все копии которых находятся в прокате, а также фильмы, которых нет в запасе и которые не заказаны у поставщика; при этом с клиента требуется задаток за один период проката.

Клиент может также сделать несколько предварительных заказов, однако для каждого забронированного фильма нужно подготовить отдельный запрос на бронирование. Бронирование может быть отменено из-за того, что клиент не проявил никакой реакции в течение недели, прошедшей с момента, когда ему сообщили о возможности взять фильм напрокат. Если за фильм был уплачен задаток, он записывается на счет клиента.

База данных хранит традиционную информацию о поставщиках и клиентах, т.е. адреса, телефонные номера и т.д. В каждом заказе поставщику указываются заказываемые фильмы, их количество, формат кассеты/диска, а также дата ожидаемой доставки, отпускная цена, возможные скидки и т.д.

Когда кассета возвращается клиентом или поступает от поставщика, работники магазина в первую очередь обслуживают клиентов, сделавших предварительный заказ. Для правильной обработки бронирования фильмов информация, связанная с бронированием, обновляется дважды: после установления контакта с клиентом, когда ему сообщается, что "забронированный фильм пришел", и после сдачи фильма клиенту напрокат. Эти шаги гарантируют правильное проведение операции бронирования.

Клиент может взять несколько кассет или дисков, однако каждому взятому видеоносителю ставится в соответствие отдельная запись. Для каждого выдаваемого напрокат фильма фиксируются дата и время выдачи, установленный и фактический срок возврата. Позже запись о прокате обновляется, чтобы отразить факт возврата видеофильма и факт окончательного платежа (или возврата денег). Кроме того, запись хранит информацию о продавце, отвечающем за прокат фильма. Детальная информация о клиенте и по прокату хранится в течение года, чтобы можно было легко определить уровень доверия к клиенту. Старая информация по прокату сохраняется в течение года в целях проведения аудита.

Все операции выполняются с использованием наличности, электронного перевода денег или кредитных карточек. От клиентов требуется внести плату за прокат при выдаче кассет/дисков.

Если кассета/диск возвращены позже установленного срока (или не могут быть возвращены по каким-либо причинам), плата снимается либо со счета клиента, либо принимается непосредственно от клиента.

Если кассета/диск задержаны более чем на два дня, клиенту отправляется уведомление о задержке. После отправки двух уведомлений о задержке одной и той же кассеты/диска клиент получает предупреждение о том, что он является "нарушителем" и при следующем обращении его в магазин руководство будет рассматривает вопрос о снятии с него статуса "нарушителя".

Служба занятости в рамках вуза

Система предназначена для того, чтобы помочь студенту устроиться на работу уже в процессе обучения его в вузе. Подав заявление в систему, студент становится ее клиентом и начинает обслуживаться на протяжении всего обучения в вузе. Заявление представляет собой анкету. Система предлагает профессиональные (основанные на изучаемых предметах) психологические тестирования, проводимые регулярно (раз в семестр (полгода)). Особое внимание уделяется обучению студента, по итогам успеваемости составляются экспертные оценки. На основе собранной информации составляется резюме, представляющее собой полную характеристику человека, и рассылается всем организациям, имеющим необходимые вакансии.

Основным назначением системы являются автоматизация ввода и хранения отчетных данных о студентах, составление характерис-

тик и резюме, поиск вакансий в фирмах. Система позволяет изменять, дополнять, вести поиск и просмотр информации о студентах, накладывать ограничения доступа к системе, хранить списки студентов, окончивших обучение, в виде архива, контролировать выдачу студенту заданий на курсовые работы и проекты, связывать институт с фирмами, заинтересованными в поиске сотрудников.

Данная система также может быть использована для составления отдельных списков групп, печати зачетных ведомостей и полной базы данных, для статистики.

Система состоит из четырех подсистем:

- контроля за успеваемостью студентов;
- профессиональных и психологических тестов;
- обработки запросов, определения категорий полномочий пользователей;
- экспертных оценок.

Подсистема "Контроль успеваемости студентов" отвечает за статистическую отчетность по успеваемости отдельного студента, группы или целого факультета, а также за хранение и правильность ее ввода.

Входными данными подсистемы являются: оценки, даты сдачи экзаменов, имена студентов, номера групп, факультет. На выходе подсистема выдает обработанные данные: средний балл по студенту, группе или факультету, процентное соотношение оценок у студента в группе или на факультете, имена и количество стипендиатов в группе или на факультете. Подсистема "Контроль успеваемости студентов" может функционировать отдельно от всей системы, что дает возможность установить и использовать ее независимо, если это необходимо.

Подсистема "Контроль успеваемости студентов" включает следующие функции:

- ввод, вывод и редактирование информации по информационным объектам подсистемы;
- сохранение информации, поступившей от подсистемы "Контроль успеваемости студентов";
- расчет процентного соотношения оценок у студента в группе или на факультете и вывод его в виде таблиц, графиков и диаграмм;
- расчет среднего балла по студенту, группе или факультету;
- формирование данных по студенту, группе или факультету;
- выявление сильнейших и слабейших студентов в группе или на факультете;

- расчет количества стипендиатов в группе или на факультете;
- проверку правильности ввода данных.

Подсистема обработки запросов, определения категорий пользователей предназначена для определения категории, полномочий и обработки запросов пользователей службы занятости. В частности, она выполняет следующие функции:

- регистрацию новых фирм;
- регистрацию новых студентов;
- определение прав доступа зарегистрированного пользователя;
- обработку запросов;
- прием регистрационных данных от фирм, студентов и обслуживающего персонала;
- составление резюме;
- запись данных в БД студентов, фирм и зарегистрированных пользователей.

В соответствии с выполняемыми функциями система работает со следующими сведениями:

- регистрационными данными студентов и фирм;
- личными данными студентов;
- информацией о студентах (получаемой фирмами);
- информацией о фирмах (получаемой студентами);
- идентификационными данными пользователей;
- информацией о системе;
- запросом;
- служебной информацией (для обслуживающего персонала);
- результатами психологического и профессионального тестов;
- экспертными оценками.

Система складского учета

Система складского учета – программная система, затрагивающая все аспекты, связанные с движением товара на склад и со склада. По результатам анализа можно выделить семь основных функций системы.

Функция системы	Описание
Учет заказов	Прием заказов от клиентов и ответы на запросы клиентов о состоянии заказов
Ведение счетов	Направление счетов клиентам и отслеживание платежей. Прием счетов от поставщиков и отслеживание платежей, направляемых поставщикам

Функция системы	Описание
Отгрузка со склада	Составление спецификаций на комплектацию товаров, отправляемых со склада клиентам
Складской учет	Постановка прибывающих товаров на учет и снятие товаров с учета при отправке заказов
Закупки	Заказ товаров поставщикам и отслеживание поставок
Получение	Принятие на склад товаров от поставщиков
Планирование	Выпуск отчетов, в том числе отражающих тенденции спроса на отдельные виды товаров и активность поставщиков

В качестве части стратегии компании, занимающейся торговлей по каталогам, по проникновению на новые участки рынка было решено создать ряд относительно автономных региональных складов продукции. Каждый такой склад несет ответственность за учет товаров и выполнение заказов. В целях повышения эффективности своей работы склад обязан сам поддерживать ту номенклатуру товаров, которая в наилучшей степени соответствует потребностям местного рынка. Номенклатура может быть разной для каждого региона и должна оперативно меняться в соответствии с потребностями клиентов. Головная компания хотела бы иметь на всех складах одинаковую систему учета.

Основными функциями системы являются:

- учет товаров, приходящих от разных поставщиков, при приеме их на склад;
- учет заказов по мере поступления их из центральной удаленной организации; заказы также могут приниматься по почте. Их обработка ведется на местах;
- генерация указаний персоналу, в частности, об упаковке товаров;
- генерация счетов и отслеживание оплат;
- генерация запросов о поставке и отслеживание платежей поставщикам.

Кроме автоматизации стандартных складских операций система также должна предоставлять богатые возможности по генерации различных форм отчетности, в том числе отражающих тенденции развития рынка, списков наиболее надежных и ненадежных поставщиков и клиентов, материалов для рекламных кампаний.

Web-сайт авиакомпании

Коммерческий отдел авиакомпании предложил расширить свой Web-сайт, чтобы пользователи смогли:

- узнать о выполнении рейсов текущего дня;
- запросить информацию о расписании рейсов, стоимости билетов и наличии мест;
- купить билеты.

Постоянные клиенты авиакомпании могут использовать также следующие функции:

- получить текущую информацию о состоянии своего личного счета (количество километров, проведенных в воздухе с начала года на данное число, количество налетанных километров для получения вознаграждения (бесплатного перелета) и т.д.);
- купить билеты, используя либо информацию о налетанных километрах (для постоянных клиентов), либо кредитную карточку.

Для того чтобы гарантировать конфиденциальность частной информации и предотвратить несанкционированное использование данных о постоянных клиентах, необходимо требовать, чтобы пользователь при доступе к личным счетам зарегистрировался, введя номер счета и личный идентификационный номер владельца карточки (PIN). После регистрации пользователь должен увидеть начальную страницу с учетом его предпочтений и привычек, полученных из базы данных, хранящей информацию о перелетах постоянных клиентов. Постоянные клиенты могут оперативно обновлять сведения о себе.

Для того чтобы сэкономить деньги, руководство компании приняло решение использовать ряд существующих систем:

- систему управления счетами, хранящую информацию о постоянных клиентах и балансе "премиальных километров";
- маркетинговую базу данных, которая отслеживает данные о выполненных рейсах, классе оплаты и др. Эти данные используются для формирования специальных уведомлений, которые включаются в ежемесячные выписки из лицевого счета постоянных клиентов;
- базу данных тарифов;
- базу данных наличия билетов.

4. РЕКОМЕНДАЦИИ ПО УСТАНОВКЕ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

При выполнении упражнений, приведенных в практикуме, используются следующие версии инструментальных средств:

- IBM Rational Rose Enterprise Edition 2003.
- IBM Rational RequisitePro 2003.

Для того чтобы обеспечить совместную работу данных инструментальных средств, в процессе установки их с компакт-диска Rational Solutions for Windows нужно выполнить следующие действия:

- в окне выбора продукта для установки (рис. П.4.1) отметить строку Rational Suite DevelopmentStudio или Rational Suite Enterprise;

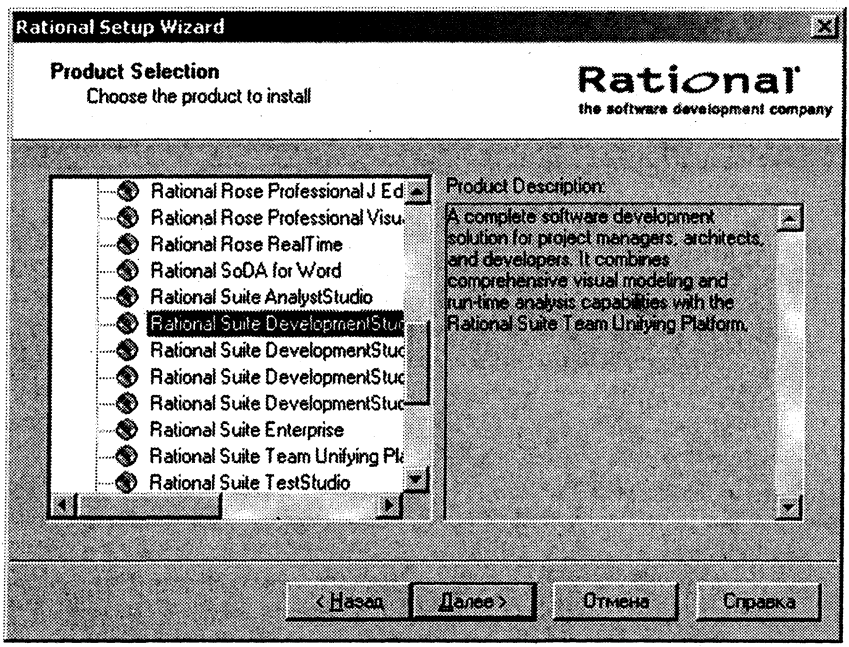


Рис. П.4.1. Окно выбора продукта для установки

• в окне выбора программ (рис. П.4.2) оставить отмеченными строку Rational Rose Enterprise Edition и Rational RequisitePro.

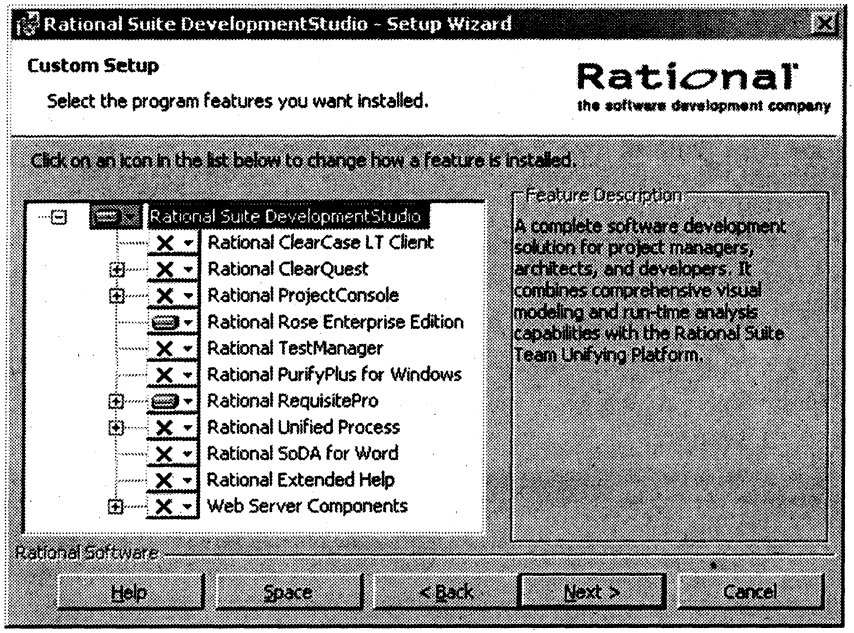


Рис. П.4.2. Окно выбора программ

ЛИТЕРАТУРА

1. *Боггс У., Боггс М.* UML и Rational Rose 2002: Пер. с англ. – М.: ЛОРИ, 2004.

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

3. *Кватрани Т.* Визуальное моделирование с помощью Rational Rose 2002 и UML.: Пер. с англ. – М.: Вильямс, 2003.

4. *Коберн А.* Современные методы описания функциональных требований к системам.: Пер. с англ. – М.: ЛОРИ, 2002.

5. *Крачтен Ф.* Введение в Rational Unified Process.: Пер. с англ. – М.: Вильямс, 2002.

6. *Ларман К.* Применение UML и шаблонов проектирования. – 2-е изд.: Пер. с англ.: – М.: Вильямс, 2002.

7. *Мацяшек Л.* Анализ требований и проектирование систем. Разработка информационных систем с использованием UML.: Пер. с англ.: – М.: Вильямс, 2002.

ОГЛАВЛЕНИЕ

Учебное издание

Вендров Александр Михайлович

ПРАКТИКУМ ПО ПРОЕКТИРОВАНИЮ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

Заведующая редакцией *Л. А. Табакова*
Ведущий редактор *Н. А. Кузнецова*
Младший редактор *Н. А. Федорова*
Художественный редактор *Ю. И. Артюхов*
Технический редактор *В. Ю. Фотиева*
Корректоры *Н. Б. Вторушина, Н. П. Сперанская*
Компьютерная верстка *Н. А. Пиминовой*
Оформление художника *Е. К. Самойлова*

ИБ № 5003

Подписано в печать 29.08.2006 г.
Формат 60×88/16. Печать офсетная.
Гарнитура "Таймс". Усл п. л. 11,76. Уч.-изд. л. 10,39.
Тираж 3000 экз. Заказ 81. "С" 167.

Издательство "Финансы и статистика"
101000, Москва, ул. Покровка, 7
Телефон (495) 625-35-02, факс (495) 625-09-57
E-mail: mail@finstat.ru http://www.finstat.ru

ООО «Великолукская городская типография»
182100, Псковская область, г. Великие Луки, ул. Полиграфистов, 78/12
Тел./факс: (811-53) 3-62-95
E-mail: zakaz@veltip.ru