А.М. Вендров

Объектно-ориентированный анализ и проектирование с использованием языка UML и Rational Rose

Практикум

Содержание

1. Основные сведения о работе в среде Rational Rose

1.1 Элементы экрана

Пять основных элементов интерфейса Rose - это браузер, окно документации, панели инструментов, окно диаграммы и журнал (log). 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) - применяются для быстрого доступа к наиболее распространенным командам
- Окно диаграммы (diagram window) - используется для просмотра и редактирования одной или нескольких диаграмм UML
- Журнал (log) - применяется для просмотра ошибок и отчетов о результатах выполнения различных команд

На рис. 1.1 показаны различные части интерфейса Rose.

Браузер

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
- просматривать существующие связи между элементами модели
- перемещать элементы модели
- переименовывать эти элементы
- добавлять элементы модели к диаграмме
- связывать элемент с файлом или адресом Интернет
- группировать элементы в пакеты
- работать с детализированной спецификацией элемента
- открывать диаграмму

Браузер поддерживает четыре представления (view): представление вариантов использования, компонентов, размещения и логическое представление. Все они и содержащиеся в них элементы модели описаны ниже в подразд. 1.2.

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

Окно документации

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

Панели инструментов

Панели инструментов Rose обеспечивают быстрый доступ к наиболее распространенным командам. В этой среде существует 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, затем выберите вкладку Toolbars.

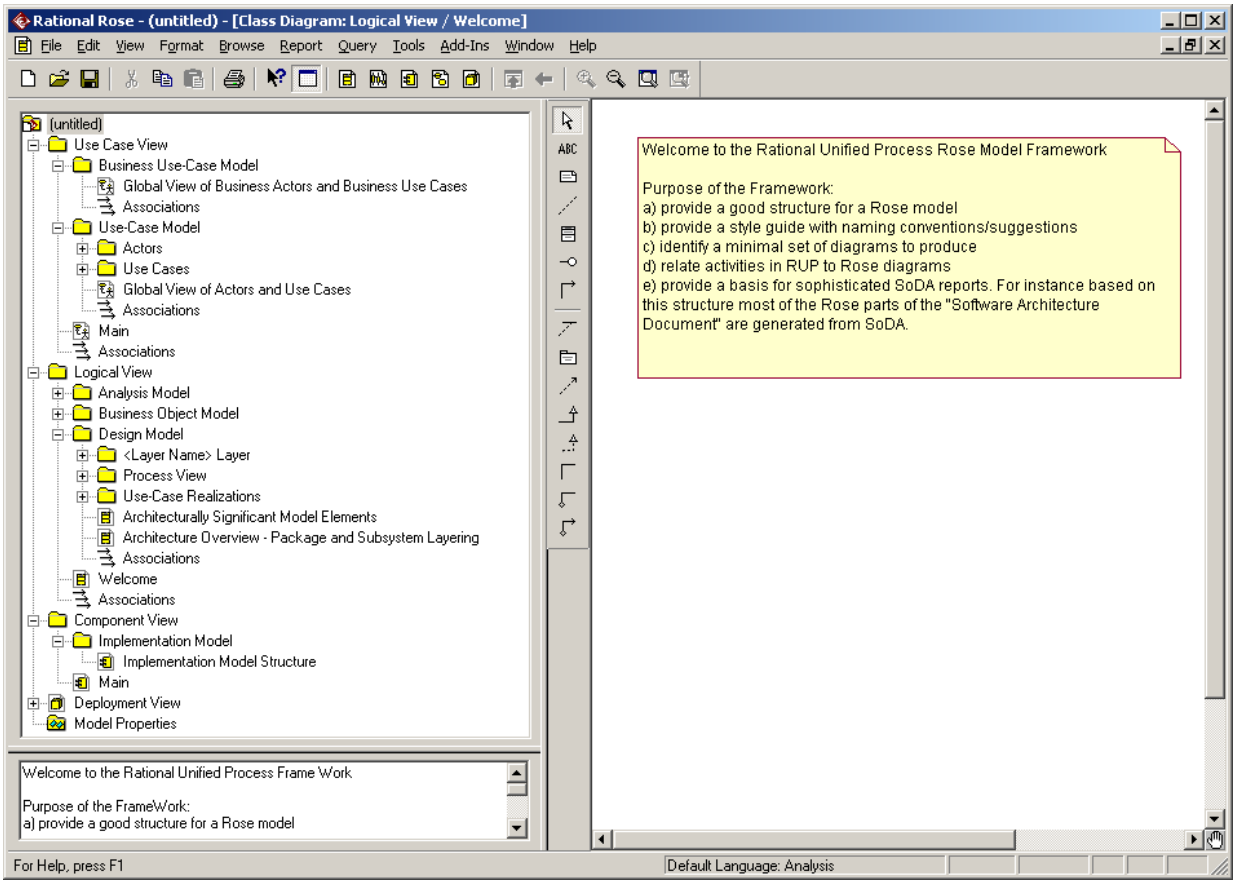

Рис. 1.1 Интерфейс Rose.

Чтобы показать или скрыть стандартную панель инструментов (или панель инструментов диаграммы):

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

Чтобы увеличить размер кнопок на панели инструментов:

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

Чтобы настроить панель инструментов:

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

Существует два способа удалить элемент модели – из одной диаграммы или из всей модели. Чтобы удалить элемент модели из диаграммы:

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

Чтобы удалить элемент из модели:

1. Выделите элемент на диаграмме.

2. Выберите пункт меню Edit > Delete from Model или нажмите сочетание клавиш CTRL + D.

Окно диаграммы

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

Журнал

По мере работы над вашей моделью определенная информация будет направляться в окно журнала. Например, туда помещаются сообщения об ошибках, возникающих при генерации кода. Не существует способа закрыть журнал совсем, но его окно может быть минимизировано.

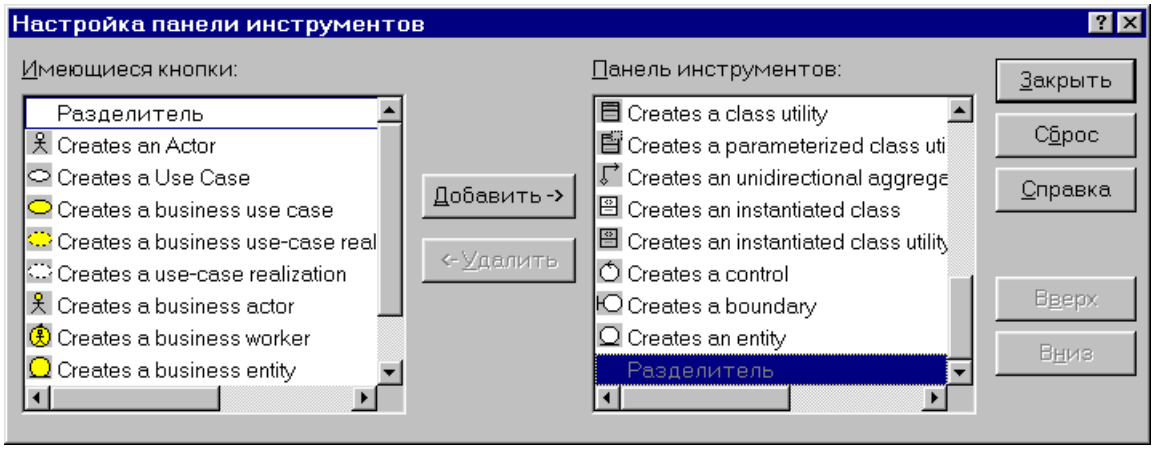

Рис. 1.2. Настройка стандартной панели инструментов.

1.2 Четыре представления модели Rose

В модели Rose поддерживается четыре представления (views) - представление вариантов использования, логическое представление, представление компонентов и представление размещения. Каждое из них предназначено для своих целей.

Представление вариантов использования

Это представление содержит модель бизнес-процессов и модель вариантов использования. На рис. 1.1 показано, как выглядит представление вариантов использования в браузере Rose.

Логическое представление

Логическое представление концентрируется на том, как система будет реализовывать поведение, описанное в вариантах использования. Оно дает подробную картину составных частей системы и описывает взаимодействие этих частей. Логическое представление включает в основном классы и диаграммы классов. С их помощью конструируется детальный проект создаваемой системы.

Логическое представление содержит:

- Классы.
- Диаграммы классов. Как правило, для описания системы используется несколько диаграмм классов, каждая из которых отображает некоторое подмножество всех классов системы.
- Диаграммы взаимодействия, применяемые для отображения объектов, участвующих в одном потоке событий варианта использования.
- Диаграммы состояний.
- Пакеты, являющиеся группами взаимосвязанных классов.

Представление компонентов

Представление компонентов содержит:

- Компоненты, являющиеся физическими модулями кода.
- Диаграммы компонентов.

– Пакеты, являющиеся группами связанных компонентов.

Представление размещения

Последнее представление Rose - это представление размещения. Оно соответствует физическому размещению системы, которое может отличаться от ее логической архитектуры.

В представление размещения входят:

– Процессы, являющиеся потоками (threads), исполняемыми в отведенной для них области памяти.
– Процессоры, включающие любые компьютеры, способные обрабатывать данные. Любой процесс выполняется на одном или нескольких процессорах.
– Устройства, то есть любая аппаратура, не способная обрабатывать данные. К числу таких устройств относятся, например, терминалы ввода-вывода и принтеры.
– Диаграмма размещения.

1.3 Параметры настройки отображения

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.

Существует два способа изменения параметров представления атрибутов на диаграмме. Можно установить нужные значения у каждого класса индивидуально. Можно также изменить значения нужных параметров по умолчанию до начала создания диаграммы классов. Внесенные таким образом изменения повлияют только на вновь создаваемые диаграммы.

Чтобы показать все атрибуты класса:

1. Выделите на диаграмме нужный класс.
2. Щелкните на нем правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. В нем выберите Options > Show All Attributes.

Чтобы показать у класса только избранные атрибуты:

1. Выделите на диаграмме нужный вам класс.
2. Щелкните на нем правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. В нем выберите Options > Select Compartment Items.
4. Укажите нужные вам атрибуты в окне Edit Compartment.

Чтобы подавить вывод всех атрибутов класса диаграммы:

1. Выделите на диаграмме нужный вам класс.
2. Щелкните на нем правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. В нем выберите Options > Suppress Attributes.

Чтобы изменить принятый по умолчанию вид атрибута:

1. В меню модели выберите пункт Tools > Options.
2. Перейдите на вкладку Diagram.
3. Для установки значений параметров отображения атрибутов по умолчанию воспользуйтесь контрольными переключателями Suppress Attributes и Show All Attributes.

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.

- 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. На диаграмме будет представлено не только имя операции, но и все ее параметры, типы данных параметров и тип возвращаемого значения операции.

Чтобы показать все операции класса:

1. Выделите на диаграмме нужный вам класс.
2. Щелкните на нем правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. В нем выберите Options > Show All Operations.

Чтобы показать только избранные операции класса:

1. Выделите на диаграмме нужный вам класс.
2. Щелкните на нем правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. В нем выберите Options > Select Compartment Items.
4. Укажите нужные вам операции в окне Edit Compartment.

Чтобы подавить вывод всех операций класса диаграммы:

1. Выделите на диаграмме нужный вам класс.
2. Щелкните на нем правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. В нем выберите Options > Suppress Operations.

Чтобы показать на диаграмме классов сигнатуру операции:

1. Выделите на диаграмме нужный вам класс.
2. Щелкните на нем правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. В нем выберите Options > Show Operation Signature.

Чтобы изменить принятый по умолчанию вид операции:

1. В меню модели выберите пункт Tools > Options.
2. Перейдите на вкладку Diagram.
3. Для установки значений параметров отображения операций по умолчанию воспользуйтесь контрольными переключателями Suppress Operations, Show All Operations и Show Operation Signatures.

Чтобы показать видимость атрибута или операции класса:

1. Выделите на диаграмме нужный вам класс.
2. Щелкните на нем правой кнопкой мыши, чтобы открыть контекстно-зависимое меню.
3. В нем выберите Options > Show Visibility.

Чтобы изменить принятое по умолчанию значение параметра показа видимости:

1. В меню модели выберите пункт Tools > Options.
2. Перейдите на вкладку Diagram.
3. Для установки параметров отображения видимости по умолчанию воспользуйтесь контрольным переключателем Show Visibility.

Для переключения между нотациями видимости Rose и UML:

1. В меню модели выберите пункт Tools > Options.
2. Перейдите на вкладку Notation.
3. Для переключения между нотациями воспользуйтесь переключателем Visibility as Icons. Если этот переключатель помечен, будет использоваться нотация Rose. Если нет, то нотация UML. Изменение этого параметра повлияет только на новые диаграммы. Существующие диаграммы классов останутся прежними.

2. Выполнение учебного проекта

2.1 Моделирование бизнес-процессов

Постановка задачи

Перед руководителем информационной службы университета ставится задача разработки автоматизированной системы регистрации студентов на дополнительные платные курсы. 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 системе, чтобы указать курсы, которые они будут читать, и проставить оценки за курсы.

В настоящее время в университете функционирует база данных, содержащая всю информацию о курсах (каталог курсов). Регистрация на курсы происходит следующим образом: в начале каждого семестра студенты могут запросить у регистратора каталог курсов, содержащий список курсов, предлагаемых в данном семестре. Информация о каждом курсе должна включать имя профессора, наименование кафедры и требования к предварительному уровню подготовки (прослушанным курсам).

Студент может выбрать 4 курса в предстоящем семестре. В дополнение к этому каждый студент может указать 2 альтернативных курса на тот случай, если какой-либо из выбранных им курсов окажется уже заполненным или отмененным. На каждый курс может 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 (добавить или отказаться от выбранных курсов). После того, как процесс регистрации некоторого студента завершен, регистратор 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 окончательного формирования его личного учебного плана. В конце семестра студенты могут просмотреть свои табели успеваемости.

2.2 Создание модели вариантов использования

Действующие лица (business actors):

- Студент – записывается на курсы и просматривает свой табель успеваемости.
- Профессор – выбирает курсы для преподавания и ставит оценки за курсы.
- Расчетная система – получает информацию по оплате за курсы.
- Каталог курсов – база данных, содержащая информацию о курсах.

Упражнение 1. Создание действующих лиц в среде Rational Rose

При запуске Rational Rose в окне Create New Model выберите вариант Rational Unified Process (рис. 2.1). В результате экран Rose примет вид, показанный на рис. 1.1.

Чтобы поместить действующее лицо в браузер:

1. Щелкните правой кнопкой мыши на пакете Business Use Case Model представления Use Case View 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.
5. Щелкните правой кнопкой мыши на действующем лице.
6. В открывшемся меню выберите пункт Open Specification.
7. В поле стереотипа выберите Business Actor и нажмите на кнопку ОК.

8. После создания действующих лиц сохраните модель под именем coursereg(analysis) с помощью пункта меню File > Save.

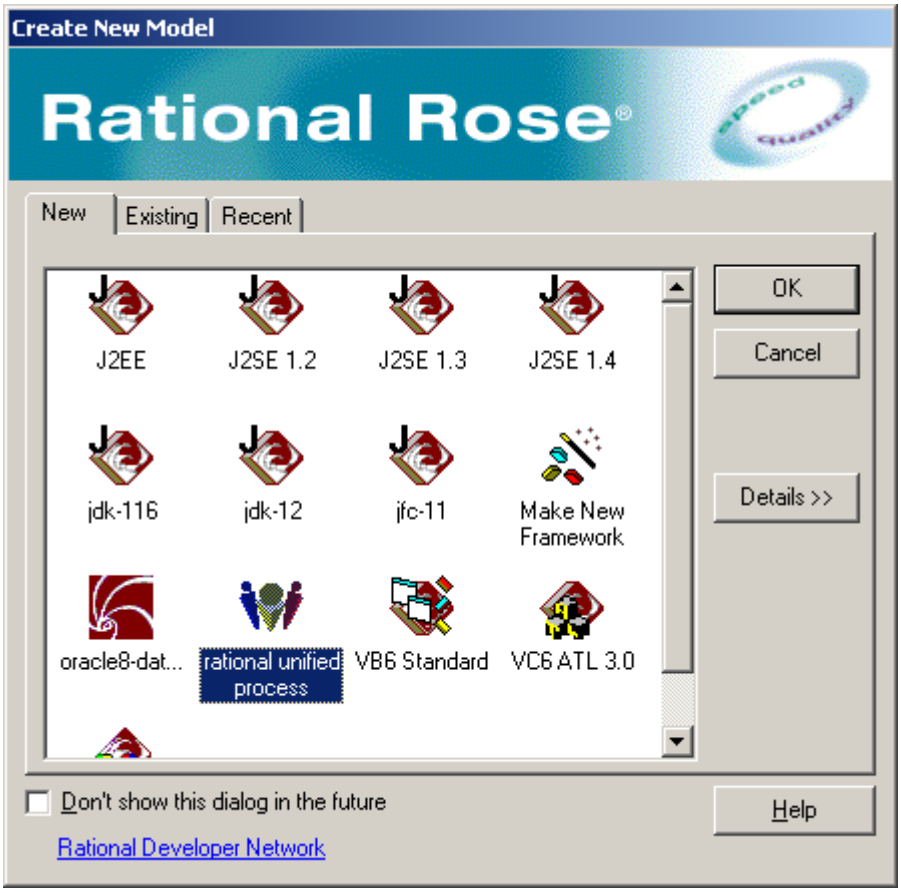

Рис. 2.1. Окно выбора шаблона модели

Варианты использования:

Исходя из потребностей действующих лиц, выделяются следующие варианты использования (Business Use Case):

- Зарегистрироваться на курсы;
- Просмотреть табель успеваемости;
- Выбрать курсы для преподавания;
- Проставить оценки.

Упражнение 2. Создание вариантов использования в среде Rational Rose

Чтобы поместить вариант использования в браузер:

1. Щелкните правой кнопкой мыши на пакете 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. Выделив новый вариант использования, введите его название.
5. Щелкните правой кнопкой мыши на варианте использования.
6. В открывшемся меню выберите пункт Open Specification.
7. В поле стереотипа выберите Business Use Case и нажмите на кнопку OK.

Диаграмма вариантов использования:

Создайте диаграмму вариантов использования для бизнес-модели системы регистрации. Требуемые для этого действия подробно перечислены далее. Готовая диаграмма вариантов использования должна выглядеть как на рис. 2.2.

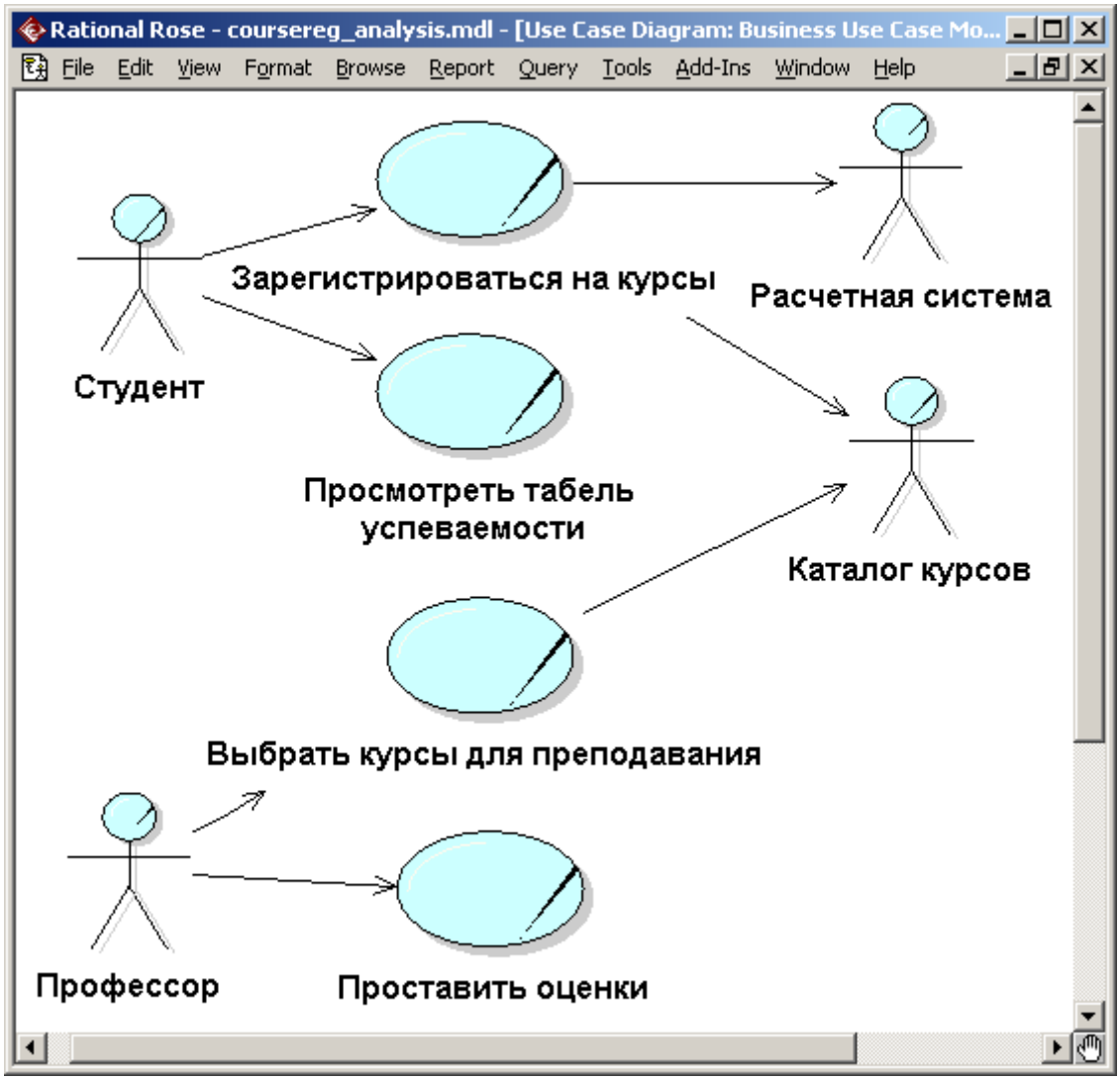

Рис. 2.2. Диаграмма вариантов использования для системы регистрации.

Упражнение 3. Построение диаграммы вариантов использования

Для создания новой диаграммы вариантов использования:

1. Щелкните правой кнопкой мыши на пакете Business Use Case Model представления Use Case View в браузере.
2. Из всплывающего меню выберите пункт New > Use Case Diagram.
3. Выделив новую диаграмму, введите ее имя (Business Use Case Diagram).
4. Дважды щелкните на названии этой диаграммы в браузере, чтобы открыть ее.
5. Чтобы поместить действующее лицо или вариант использования на диаграмму, перетащите его мышью из браузера на диаграмму вариантов использования.
6. С помощью кнопки Unidirectional Association (Однонаправленная ассоциация) панели инструментов нарисуйте ассоциации между действующими лицами и вариантами использования.

Упражнение 4. Добавление описаний к вариантам использования

1. Выделите в браузере вариант использования «Зарегистрироваться на курсы».
2. В окне документации введите следующее описание к этому варианту использования: «Данный Business Use Case позволяет студенту зарегистрироваться на конкретные курсы в текущем семестре. Студент может изменить свой выбор, если изменение выполняется в установленное время в начале семестра».
3. Создайте с помощью MS Word текстовый файл с 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. Студент может изменить свой выбор, если изменение выполняется в установленное время в начале семестра.

Основной сценарий:

1. Студент приходит к регистратору и просит зарегистрировать его на предлагаемые курсы или изменить свой график курсов.
2. В зависимости от запроса студента, выполняется один из подчиненных сценариев (создать график или изменить график).

Подчиненный сценарий «Создать график»:

1. Регистратор выполняет поиск в каталоге курсов доступных в настоящий момент курсов и выдает студенту их список.
2. Студент выбирает из списка 4 основных курса и 2 альтернативных курса.
3. Регистратор формирует график студента.
4. Выполняется подчиненный сценарий «Принять график».

Подчиненный сценарий «Изменить график»:

1. Регистратор находит текущий график студента.
2. Регистратор выполняет поиск в каталоге курсов доступных в настоящий момент курсов и выдает студенту их список.
3. Студент может изменить свой выбор курсов, удаляя или добавляя предлагаемые курсы.
4. После выбора регистратор обновляет график.
5. Выполняется подчиненный сценарий «Принять график».

Подчиненный сценарий «Принять график»:

1. Для каждого выбранного студентом курса регистратор подтверждает выполнение студентом предварительных требований (прохождение определенных курсов), факт открытия предлагаемого курса и отсутствие конфликтов графика.
2. Регистратор вносит студента в список каждого выбранного предлагаемого курса. Курс фиксируется в графике.

Альтернативные сценарии:

Не выполнены предварительные требования, курс заполнен или имеют место конфликты графика:

Если во время выполнения подчиненного сценария «Принять график» регистратор обнаружит, что студент не выполнил необходимые предварительные требования, или выбранный им предлагаемый курс заполнен (уже записалось 10 студентов), или имеют место конфликты графика (два или более курсов с совпадающим расписанием), то он предлагает студенту изменить свой выбор курсов, либо отменить формирование графика и вернуться к нему позже.

Система каталога курсов недоступна:

Если во время поиска в каталоге курсов окажется, что невозможно установить связь с системой каталога курсов, то регистрацию придется прервать и дождаться восстановления связи.

Регистрация на курсы закончена:

Если в самом начале выполнения регистрации окажется, что регистрация на текущий семестр уже закончена, то процесс завершится.

Упражнение 5. Прикрепление файла к варианту использования

1. Щелкните правой кнопкой мыши на варианте использования.
2. В открывшемся меню выберите пункт Open Specification
3. Перейдите на вкладку Files.
4. Щелкните правой кнопкой мыши на белом поле и из открывшегося меню выберите пункт Insert File.
5. Укажите созданный ранее файл и нажмите на кнопку Open, чтобы прикрепить файл к варианту использования.

2.3 Создание модели бизнес-анализа

Исполнители:

1. Регистратор – формирует учебный план и каталог курсов, записывает студентов на курсы, ведет все данные о курсах, профессорах, успеваемости и студентах.

Сущности:

2. Студент.
3. Профессор.
4. График студента (список курсов)
5. Курс (в программе обучения).
6. Предлагаемый курс (курс в расписании).

Упражнение 6. Создание классов, участвующих в реализации бизнес-процесса «Зарегистрироваться на курсы», и кооперации, описывающей реализацию бизнес-процесса

1. Щелкните правой кнопкой мыши на пакете 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 на данном классе.
5. В открывшемся меню выберите пункт Open Specification.
6. В поле стереотипа выберите Business Worker и нажмите на кнопку ОК.
7. Создайте аналогичным образом классы-сущности со стереотипом Business Entity.
8. Щелкните правой кнопкой мыши на пакете Object Model представления Logical View в браузере.
9. В открывшемся меню выберите пункт New > Package
10. Назовите новый пакет Business Use-Case Realizations.
11. В пакете Business Use-Case Realizations создайте кооперацию «Зарегистрироваться на курсы» (кооперация представляет собой вариант использования со стереотипом «business use-case realization», который задается в спецификации варианта использования).
12. Щелкните правой кнопкой мыши на созданной кооперации.
13. В открывшемся меню выберите пункт New > Class Diagram.
14. Назовите новую диаграмму классов VOPC.
15. Откройте ее и перетащите классы на открытую диаграмму в соответствии с рис. 2.3.

Диаграмма классов для модели бизнес-анализа, описывающей Business Use Case «Зарегистрироваться на курсы», приведена на рис. 2.3 (для данных классов использовано

изображение стереотипа 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.

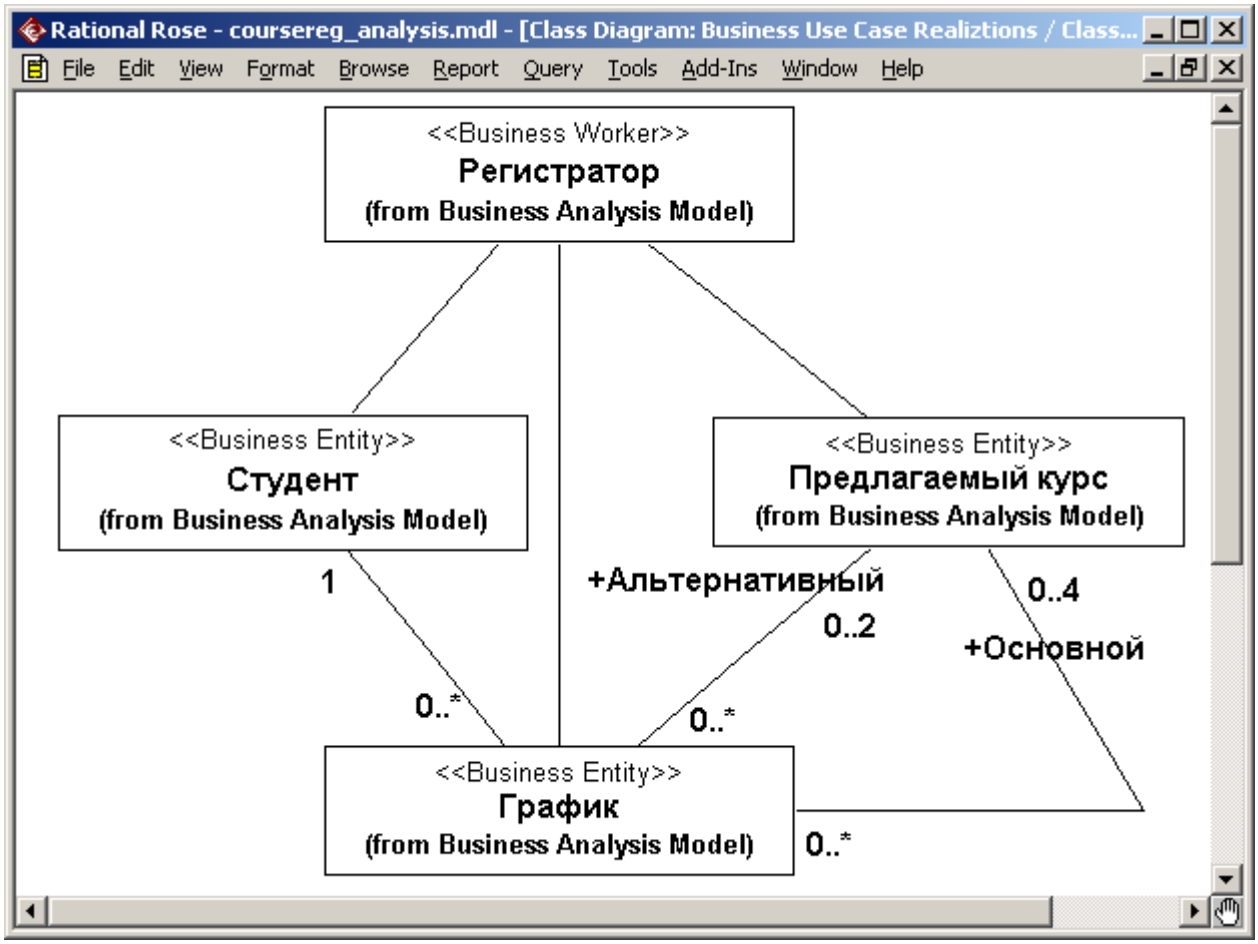

Рис. 2.3. Диаграмма классов модели бизнес-анализа

2.4 Спецификация требований к ПО

Уточненная постановка задачи для системы

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

Из-за недостатка средств университет не в состоянии заменить сразу всю существующую систему. Остается функционировать в прежнем виде база данных, содержащая всю информацию о курсах (каталог курсов). Эта база данных 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 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 4 курса в предстоящем семестре. В дополнение, каждый студент может указать 2 альтернативных курса на тот случай, если какой-либо из выбранных им курсов окажется уже заполненным или отмененным. На каждый курс может 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 окончательного формирования его личного учебного плана.

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.

2.5 Составление глоссария проекта

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

Курс	Учебный курс, предлагаемый университетом
Предлагаемый курс (Course Offering)	Предлагаемое чтение данного курса в конкретном семестре (один и тот же курс может вестись в нескольких параллельных сессиях). Включает точные 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 (Report Card)	Все оценки за все курсы, полученные студентом в данном семестре.
Список курса (Roster)	Список всех студентов, записавшихся на предлагаемый курс.
Студент	Личность, проходящая обучение в университете.
Учебный график (Schedule)	Курсы, выбранные студентом в текущем семестре.

2.6 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 надежность, удобство использования, производительность, сопровождаемость, а также ряд функциональных требований, являющихся общими для нескольких вариантов использования.

Функциональные возможности

Система должна обеспечивать многопользовательский режим работы.

Если предлагаем курс оказывается заполненным в то время, когда студент формирует свой учебный график, включающий данный курс, то система должна известить его об этом.

Удобство использования

Пользовательский интерфейс должен быть Windows 95/98-совместимым.

Надежность

Система должна быть в работоспособном состоянии 24 часа в день 7 дней в неделю, время простоя – не более 10%.

Производительность

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

Безопасность

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

Только профессора имеют право ставить студентам оценки.

Только регистратор может изменять любую информацию о студентах.

Проектные ограничения

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

2.7 Создание начальной версии модели вариантов использования

Действующие лица:

- Регистратор – формирует учебный план и каталог курсов, записывает студентов на курсы, ведет все данные о курсах, профессорах, успеваемости и студентах.
- Расчетная система – получает от данной системы информацию по оплате за курсы.
- Каталог курсов – база данных, содержащая информацию о курсах.

Упражнение 7. Создание действующих лиц в среде Rational Rose

Чтобы поместить действующее лицо в браузер:

1. Щелкните правой кнопкой мыши на пакете Use Case Model представления Use Case View 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.

Варианты использования:

Исходя из потребностей действующих лиц, выделяются следующие варианты использования:

- Войти в систему;
- Зарегистрировать студента на курсы;
- Вывести табель успеваемости;
- Назначить курсы для преподавания;
- Проставить оценки;
- Вести информацию о профессорах;
- Вести информацию о студентах;
- Закрыть регистрацию.

Начальная версия диаграммы вариантов использования показана на рис. 2.4.

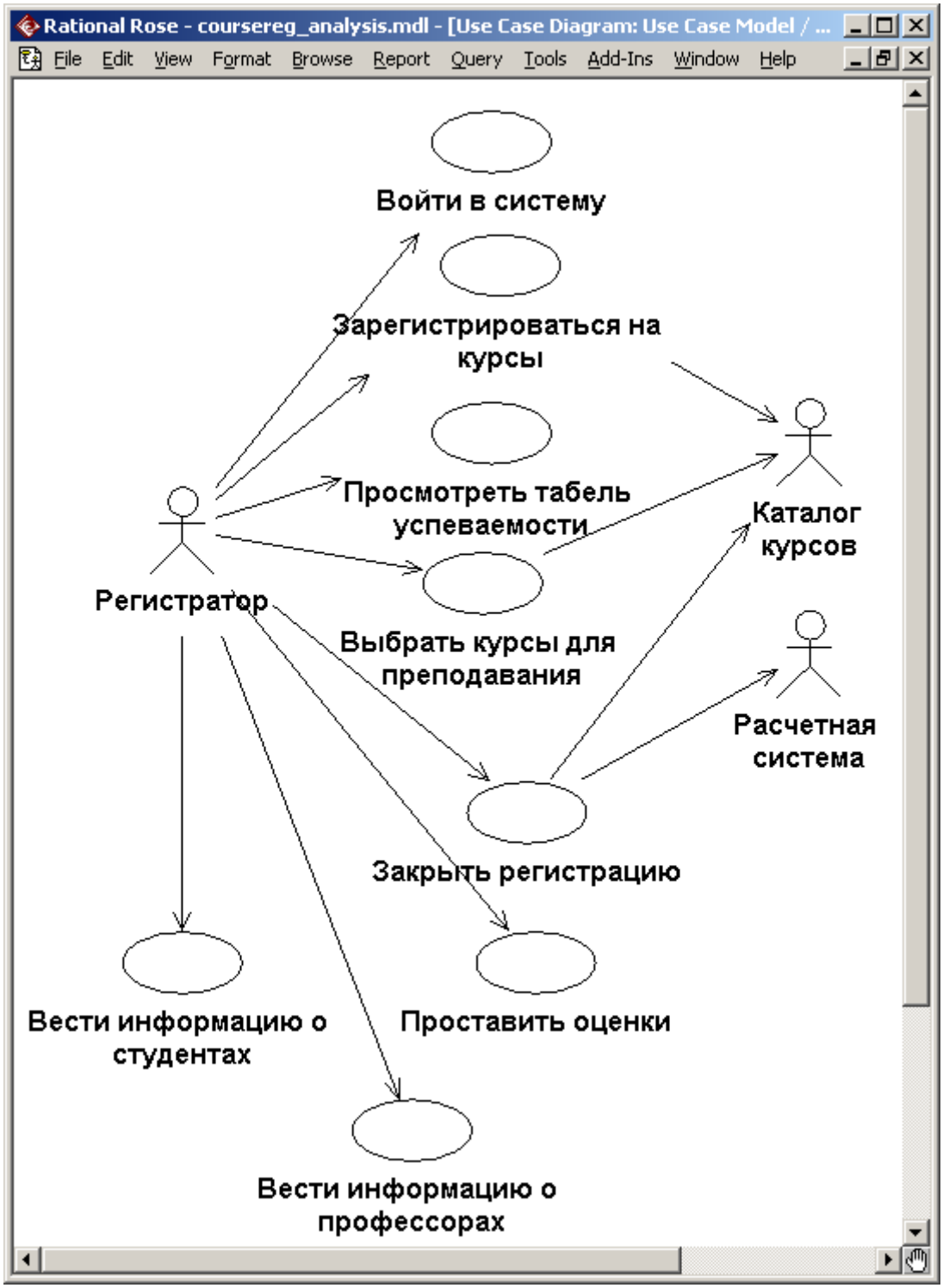

Рис. 2.4. Начальная версия диаграммы вариантов использования

Упражнение 8. Создание вариантов использования в среде Rational Rose
Чтобы поместить вариант использования в браузер:

1. Щелкните правой кнопкой мыши на пакете 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. Выделив новый вариант использования, введите его название.

Диаграмма вариантов использования:

Создайте диаграмму вариантов использования для системы регистрации. Требуемые для этого действия подробно перечислены далее. Готовая диаграмма вариантов использования должна выглядеть как на рис. 2.4.

В среде Rose диаграммы вариантов использования создаются в представлении вариантов использования. Главная диаграмма (Main) предлагается по умолчанию. Для моделирования системы можно затем разработать столько дополнительных диаграмм, сколько необходимо.

Чтобы получить доступ к главной диаграмме вариантов использования:

1. Рядом с представлением вариантов использования в браузере щелкните на значке "+", это приведет к открытию данного представления.
2. Дважды щелкните на главной диаграмме Main, чтобы открыть ее. Строка заголовка изменится, включив фразу [Use Case Diagram: Use Case view / Main].

Для создания новой диаграммы вариантов использования:

1. Щелкните правой кнопкой мыши на пакете представления вариантов использования в браузере.
2. Из всплывающего меню выберите пункт New > Use Case Diagram.
3. Выделив новую диаграмму, введите ее имя.
4. Дважды щелкните на названии этой диаграммы в браузере, чтобы открыть ее.

Упражнение 9. Построение диаграммы вариантов использования

1. Откройте диаграмму вариантов использования Main.
2. Чтобы поместить действующее лицо или вариант использования на диаграмму, перетащите его мышью из браузера на диаграмму вариантов использования.
3. С помощью кнопки Unidirectional Association (Однонаправленная ассоциация) панели инструментов нарисуйте ассоциации между действующими лицами и вариантами использования.

2.8 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. 2.5. Поскольку вход в систему полностью одинаков для регистратора, студента и профессора, их поведение можно обобщить и ввести новое действующее «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.

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.

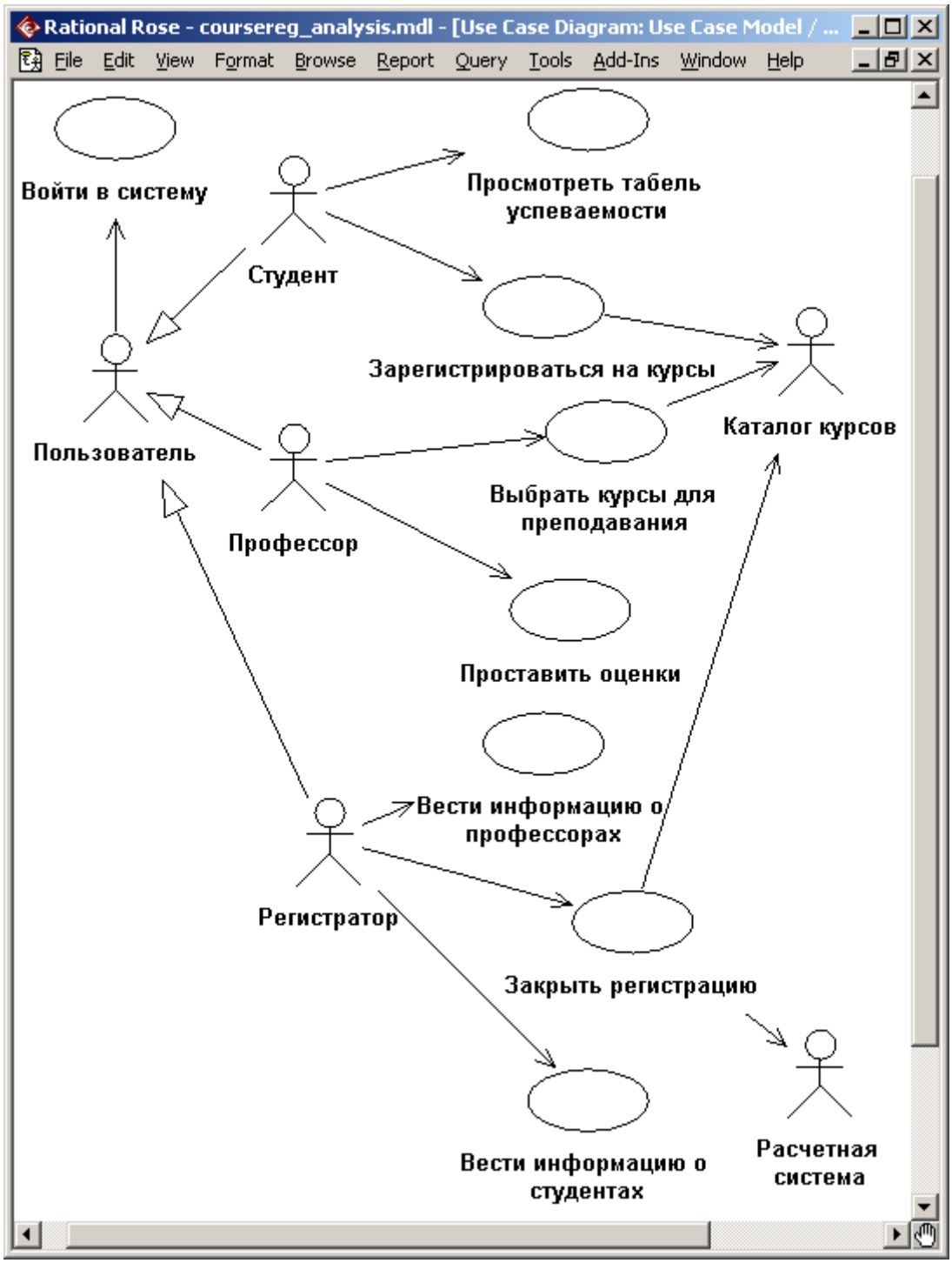

Рис. 2.5. Модифицированная диаграмма вариантов использования для системы регистрации

Упражнение 10. Добавление описаний к вариантам использования

1. Выделите в браузере вариант использования «Зарегистрироваться на курсы».
2. В окне документации введите 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 описаниями вариантов использования «Войти в систему», «Зарегистрироваться на курсы» и «Закрыть регистрацию».

<u>Спецификации вариантов использования</u>

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.

Предусловия:

Отсутствуют.

Постусловия:

Если вариант использования выполнен успешно, пользователь входит в систему. В противном случае состояние системы не изменяется.

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. Студент выбирает из списка 4 основных курса и 2 альтернативных курса.
3. После выбора система создает график студента.
4. Выполняется подчиненный поток «Принять график».

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. После выбора система обновляет график.
5. Выполняется подчиненный поток «Принять график».

Удалить график:

1. Система выводит текущий график студента.
2. Система запрашивает у студента подтверждения 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.

Принять график:

Для каждого выбранного, но еще не «зафиксированного» предлагаемого курса в графике система проверяет выполнение студентом предварительных требований (прохождение определенных курсов), факт открытия предлагаемого курса и отсутствие конфликтов графика. Затем система добавляет студента в список выбранного предлагаемого курса. Курс фиксируется в графике и график сохраняется в системе.

Альтернативные потоки:

Сохранить график:

В любой момент студент может вместо принятия графика сохранить его. В этом случае шаг «Принять график» заменяется на следующий:

1. «Незафиксированные» конкретные курсы помечаются в графике как «выбранные».
2. График сохраняется в системе.

Не выполнены предварительные требования, курс заполнен или имеют место конфликты графика:

Если во время выполнения подчиненного потока «Принять график» система обнаружит, что студент не выполнил необходимые предварительные требования, или выбранный им предлагаемый курс заполнен, или имеют место конфликты графика, то выдается сообщение об ошибке. Студент может либо выбрать другой предлагаемый курс и продолжить выполнение варианта использования, либо сохранить график, либо отменить операцию, после чего основной поток начнется с начала.

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 с начала.

Система каталога курсов недоступна:

Если окажется, что невозможно установить связь с системой каталога курсов, 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.

Удаление отменено:

Если во время выполнения подчиненного потока «Удалить график» студент решит не удалять его, удаление отменяется, и основной поток начнется с начала.

Предусловия:

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

Постусловия:

Если вариант использования завершится успешно, график студента будет создан, обновлен или удален. В противном случае состояние системы не изменится.

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 проверяет состояние процесса регистрации. Если регистрация еще выполняется, выдается сообщение и вариант использования завершается.
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 фиксирует предлагаемый курс в каждом графике, который включает данный курс.
3. Для каждого студенческого графика проверяется наличие в нем максимального количества основных курсов; если их недостаточно, система пытается дополнить альтернативными курсами из списка данного графика. Выбирается первый доступный альтернативный курс. Если таких курсов нет, то никакое дополнение не происходит.
4. Система закрывает все предлагаемые курсы. Если в каком-либо предлагаемом курсе оказывается менее трех студентов (с учетом добавлений, сделанных в п.3), система отменяет его и исключает из каждого содержащего его графика.
5. Система рассчитывает плату за обучение для каждого студента в текущем семестре и направляет информацию в расчетную систему. Расчетная система посылает студентам счета для оплаты с копией их окончательных графиков.

Альтернативные потоки:

Курс никто не ведет:

Если во время выполнения основного потока обнаруживается, что некоторый курс не ведется никаким профессором, то этот курс отменяется. Система исключает данный курс из каждого содержащего его графика.

Расчетная система недоступна:

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

Предусловия:

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

Постусловия:

Если вариант использования завершится успешно, регистрация закрывается. В противном случае состояние системы не изменится.

Упражнение 11. Прикрепление файла к варианту использования

1. Щелкните правой кнопкой мыши на варианте использования.
2. В открывшемся меню выберите пункт Open Specification
3. Перейдите на вкладку файлов.
4. Щелкните правой кнопкой мыши на белом поле и из открывшегося меню выберите пункт Insert File.
5. Укажите созданный ранее файл и нажмите на кнопку Open, чтобы прикрепить файл к варианту использования.

2.9 Анализ системы

2.9.1 Архитектурный анализ

Архитектурный анализ выполняется архитектором системы и включает в себя:

- 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 начинаться с заглавной буквы.
- Имена атрибутов и операций должны начинаться со строчной буквы.
- Составные имена должны быть сплошными, без подчеркиваний, каждое отдельное слово должно начинаться с заглавной 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. 2.6).

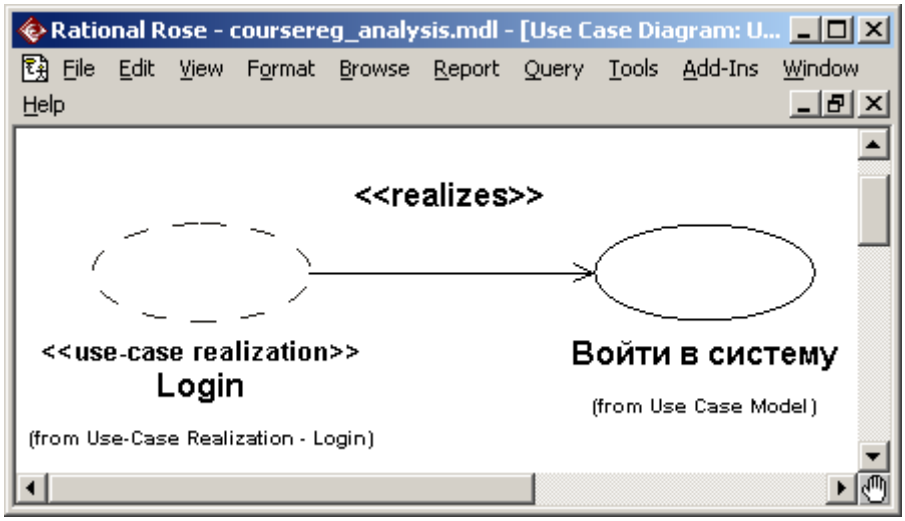

Рис. 2.6. Фрагмент диаграммы трассировки

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 идентификации основных абстракций аналогичны способам идентификации сущностей в модели ERM.

Так, для системы регистрации идентифицировано пять классов анализа:

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

Классы анализа показаны на рис. 2.7.

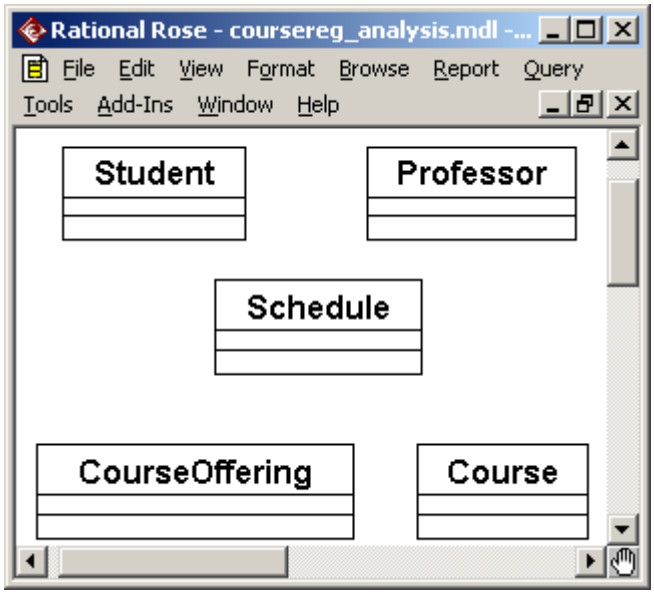

Рис. 2.7. Классы анализа системы регистрации

Упражнение 12. Создание структуры модели и классов анализа в соответствии с требованиями архитектурного анализа

Создание пакетов и диаграммы Traceabilities:

1. Щелкните правой кнопкой мыши на пакете Design Model логического представления браузера.
2. В открывшемся меню выберите пункт New > Package.
3. Создайте пакет Use-Case Realizations, затем внутри него – пакеты Use-Case Realization - Close Registration, Use-Case Realization – Login и Use-Case Realization - Register for Courses.
4. В каждом из пакетов типа Use-Case Realization создайте соответствующие кооперации Close Registration, Login и Register for Courses (каждая кооперация представляет собой вариант использования со стереотипом «use-case realization», который задается в спецификации варианта использования).
5. Создайте в пакете Use-Case Realizations новую диаграмму вариантов использования с названием Traceabilities и постройте ее в соответствии с рис. 2.8.

Создание классов анализа и соответствующей диаграммы Key Abstractions:

1. Щелкните правой кнопкой мыши на пакете Analysis Model.
2. Выберите в открывшемся меню пункт New > Class. 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 на пакете Analysis Model.
6. В открывшемся меню выберите пункт New > Class Diagram.
7. Назовите новую диаграмму классов Key Abstractions.
8. Чтобы расположить вновь созданные классы на диаграмме классов, откройте ее и перетащите классы на открытую диаграмму мышью. Диаграмма классов должна выглядеть как на рис. 2.7.

Архитектурные уровни представляются в модели (пакет Design Model) в виде пакетов со стереотипом <<layer>>. Количество и структура уровней зависят от сложности предметной области и среды реализации. В рамках архитектурного анализа определяется начальная структура

модели (набор пакетов и их зависимостей) и рассматриваются только верхние уровни (Application и Business Services).

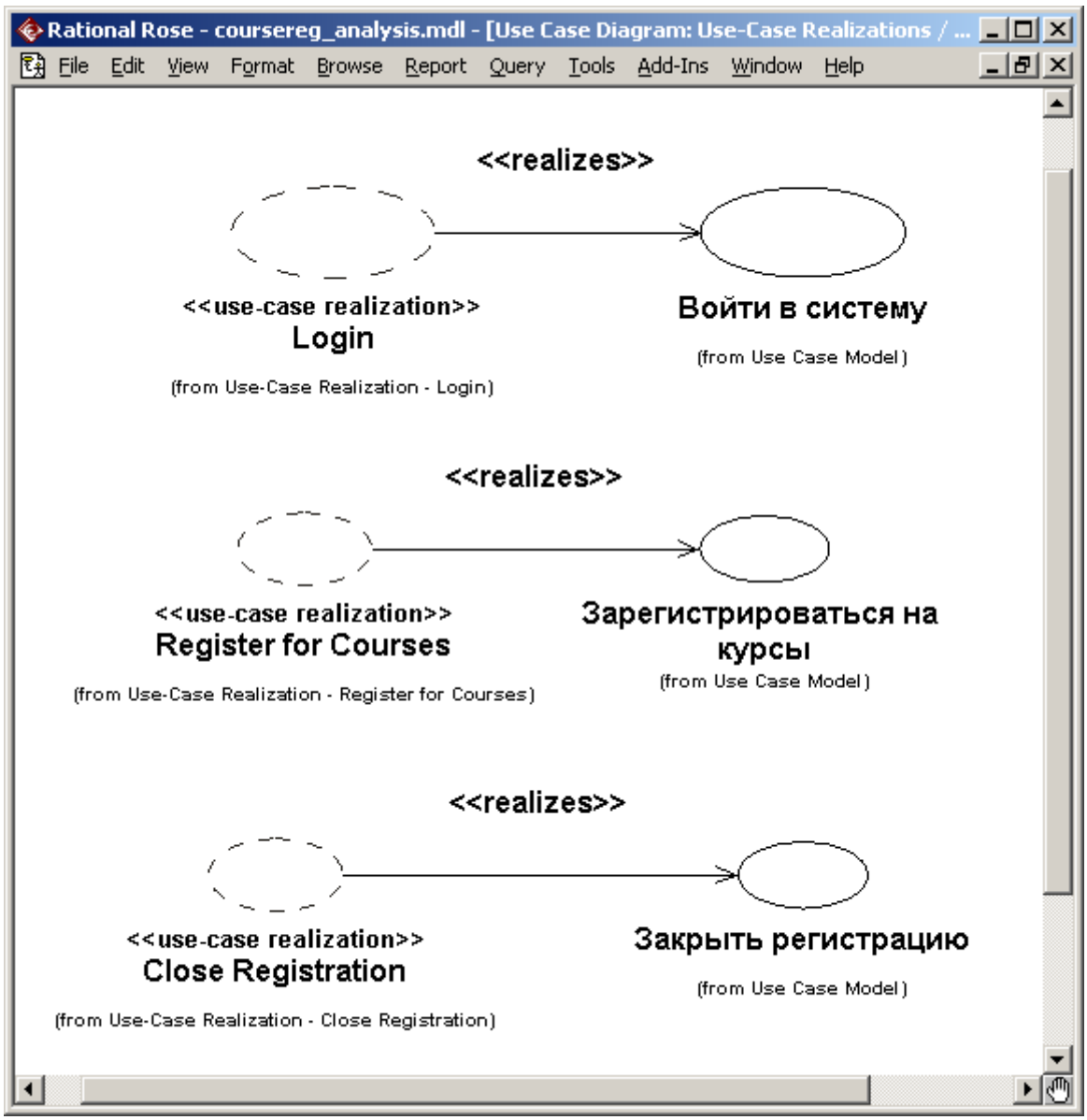

Рис. 2.8 Диаграмма Traceabilities

2.9.2 Анализ вариантов использования

Идентификация классов, участвующих в реализации потоков событий варианта использования

В потоках событий варианта использования выявляются классы трех типов:

1. **Граничные классы (Boundary)** – служат посредниками при взаимодействии внешних объектов с системой. Как правило, для каждой пары «действующее лицо - вариант использования» определяется один граничный класс. Типы граничных классов: пользовательский 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)** – представляют собой 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.

Пример набора классов, участвующих в реализации варианта использования «Зарегистрироваться на курсы», приведен на рис. 2.9.

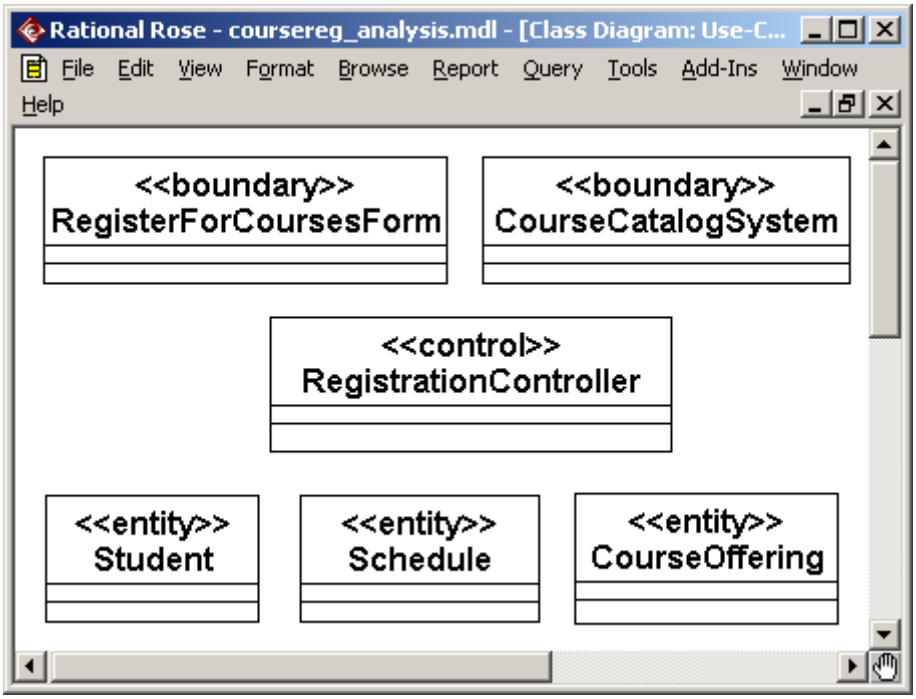

Рис. 2.9. Классы, участвующие в реализации варианта использования «Зарегистрироваться на курсы»

Упражнение 13. Создание классов, участвующих в реализации варианта использования Register for Courses, и диаграммы классов «View Of Participating Classes» (VOPC)

1. Щелкните правой кнопкой мыши на пакете Analysis Model.
2. Выберите в открывшемся меню пункт New > Class. Новый класс под названием NewClass появится в браузере.
3. Выделите его и введите имя RegisterForCoursesForm.
4. Щелкните правой кнопкой мыши на классе RegisterForCoursesForm.
5. В открывшемся меню выберите пункт Open Specification.
6. В поле стереотипа выберите Boundary и нажмите на кнопку OK.
7. Создайте аналогичным образом классы CourseCatalogSystem со стереотипом Boundary и RegistrationController со стереотипом Control.
8. Назначьте классам Schedule, CourseOffering и Student стереотип Entity.
9. Щелкните правой кнопкой мыши на кооперации Register for Courses в пакете Use-Case Realization - Register for Courses.
10. В открывшемся меню выберите пункт New > Class Diagram.
11. Назовите новую диаграмму классов VOPC (classes only).
12. Откройте ее и перетащите классы на открытую диаграмму в соответствии с рис. 2.9.

Распределение поведения, реализуемого вариантом использования, между классами

Реализуется с помощью диаграмм взаимодействия (диаграмм последовательности и кооперативных диаграмм). В первую очередь строится диаграмма (одна или более), 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 14. Создание диаграмм взаимодействия

Создадим диаграммы последовательности и кооперативные диаграммы для основного потока событий варианта использования Register for Courses. Готовые диаграммы последовательности должны иметь вид, как на рис. 2.10 – 2.14.

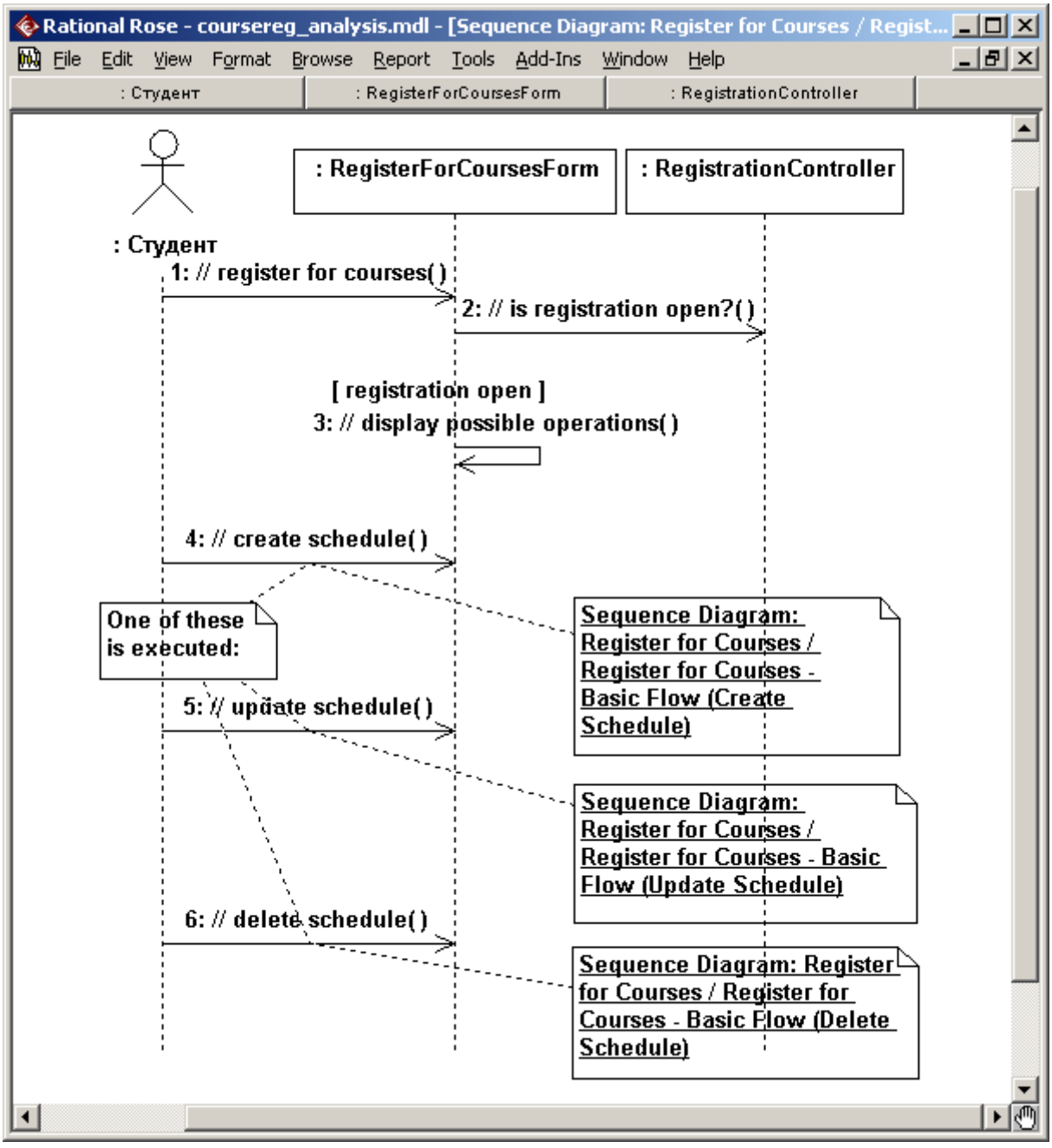

Рис. 2.10 Диаграмма последовательности Register for Courses - Basic Flow

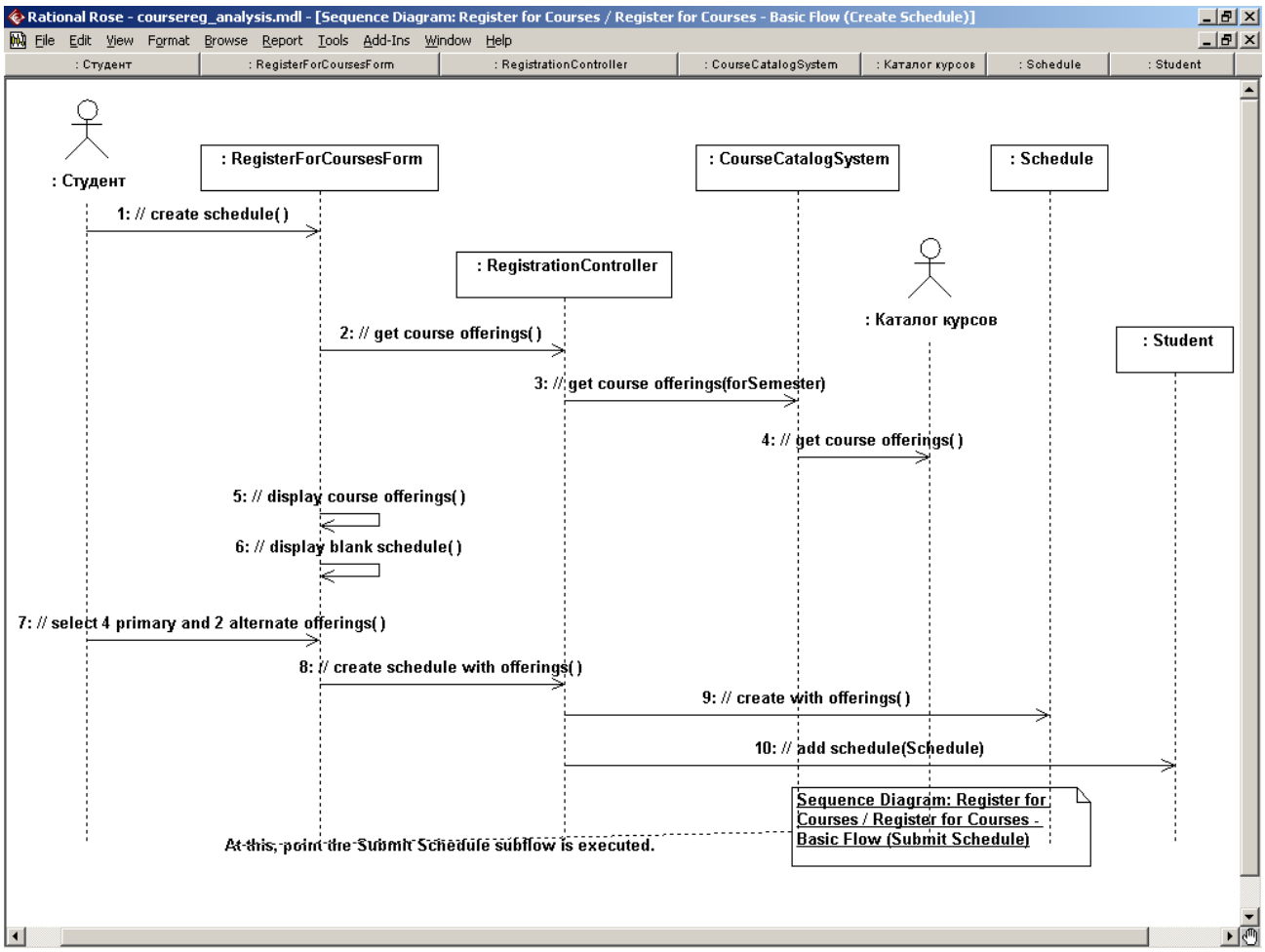

Рис. 2.11 Диаграмма последовательности Register for Courses - Basic Flow (Create Schedule)

Настройка

1. В меню модели выберите пункт Tools > Options.
2. Перейдите на вкладку диаграмм.
3. Контрольные переключатели Sequence Numbering, Collaboration Numbering должны быть помечены, а Focus of Control – нет.
4. Нажмите OK, чтобы выйти из окна параметров.

Создание диаграммы последовательности

1. Щелкните правой кнопкой мыши на кооперации Register for Courses в пакете Use-Case Realization - Register for Courses.
2. В открывшемся меню выберите пункт New > Sequence Diagram.
3. Назовите новую диаграмму Register for Courses - Basic Flow.
4. Дважды щелкните на ней, чтобы открыть ее.

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.
3. На панели инструментов нажмите кнопку Object Message (Сообщение объекта).
4. Проведите мышью от линии жизни действующего лица «Студент» к линии жизни объекта RegisterForCoursesForm.
5. Выделив сообщение, введите его имя: // register for courses.

6. Повторите действия 3 – 5, чтобы поместить на диаграмму остальные сообщения, как показано на рис. 2.10 (для рефлексивного сообщения 3 используется кнопка Message to Self).

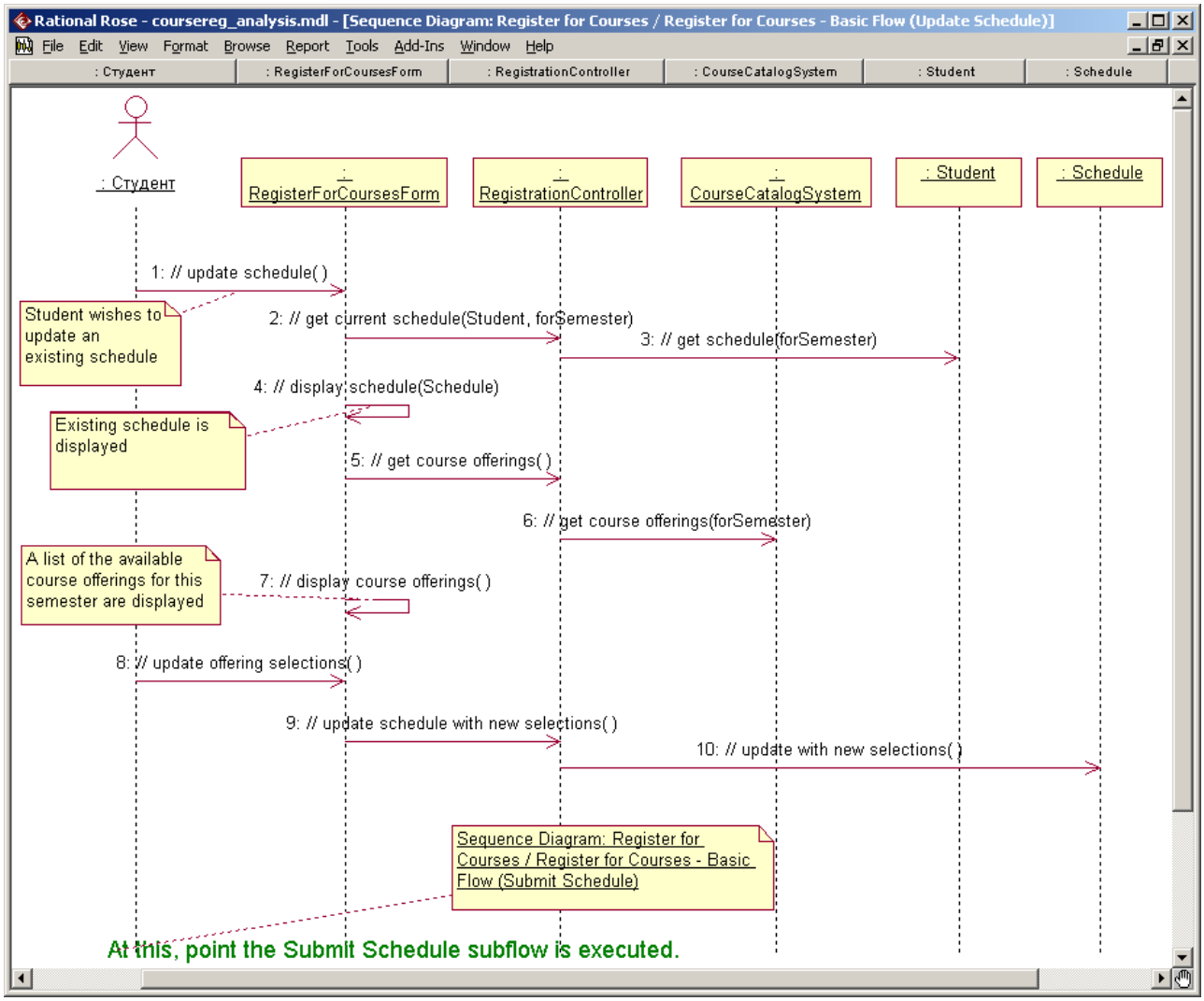

Рис. 2.12 Диаграмма последовательности Register for Courses - Basic Flow (Update Schedule)

Соотнесение сообщений с операциями

1. Щелкните правой кнопкой **на тексте** сообщении 1, // register for courses.
2. В открывшемся меню выберите пункт <new operation>. Появится окно спецификации операции.
3. В поле имени оставьте имя сообщения - // register for courses.
4. Нажмите на кнопку ОК, чтобы закрыть окно спецификации операции и вернуться на диаграмму.
5. Повторите действия 1 – 4, пока не соотнесете с операциями все остальные сообщения.

Выполните аналогичные действия для создания диаграмм последовательности, показанных на рис. 2.11 – 2.14. Обратите внимание, что на диаграмме рис. 2.14 появился объект нового класса PrimarySheduleOfferingInfo (ассоциации-класса, описывающего связь между классами Shedule и OfferingInfo), который нужно предварительно создать.

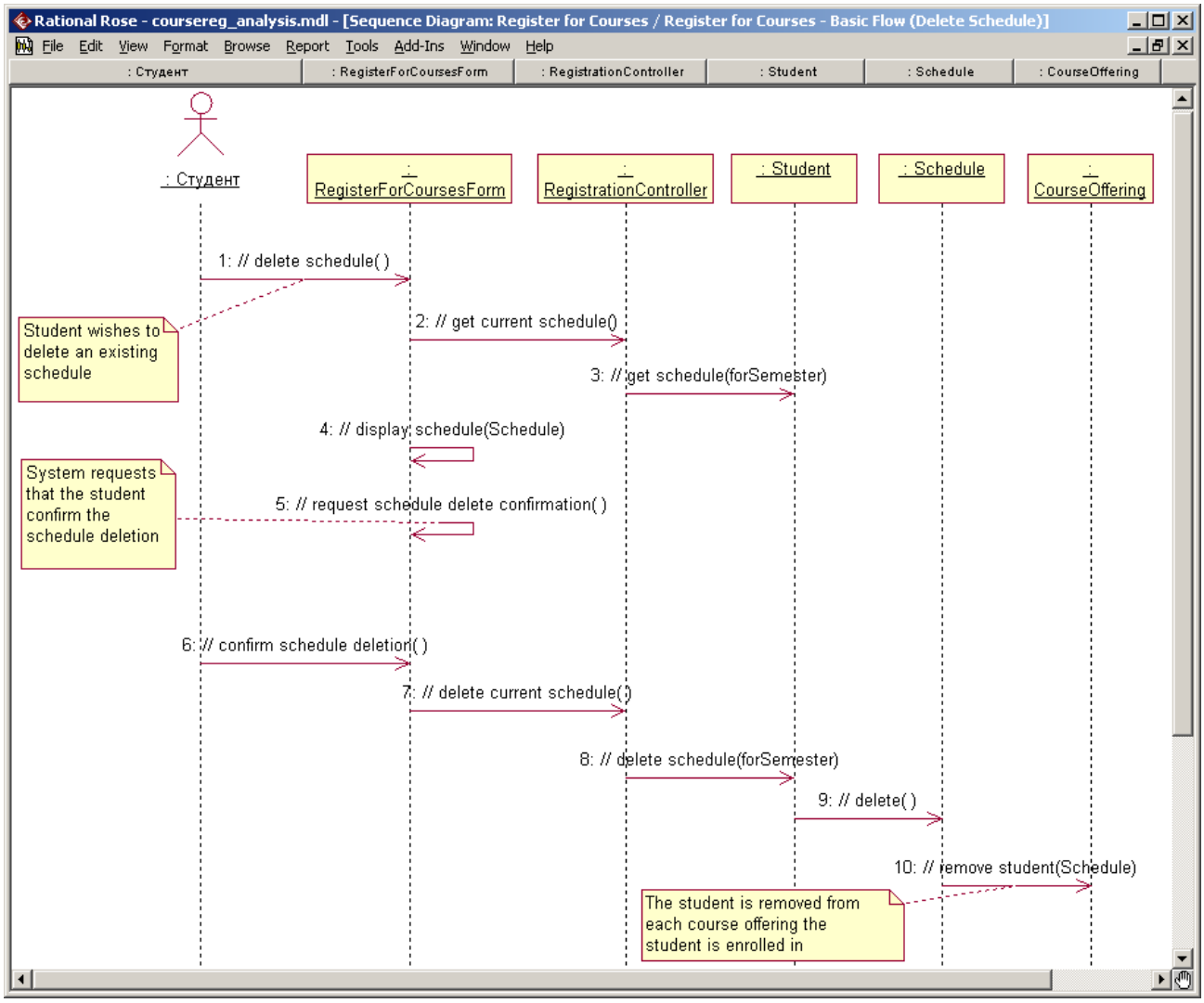

Рис. 2.13 Диаграмма последовательности Register for Courses - Basic Flow (Delete Schedule)

Создание примечаний

Чтобы поместить на диаграмму примечание:

1. Нажмите на панели инструментов кнопку Note.
2. Щелкните мышью в том месте диаграммы, куда собираетесь поместить примечание.
3. Выделив новое примечание, введите туда текст.
4. Чтобы прикрепить примечание к элементу диаграммы, на панели инструментов нажмите кнопку Anchor Notes To Item (Прикрепить примечания к элементу).
5. Нажав левую кнопку мыши, проведите указатель от примечания до элемента диаграммы, с которым оно будет связано. Между примечанием и элементом возникнет штриховая линия.
6. Чтобы создать примечание-ссылку на другую диаграмму (как это сделано на диаграмме рис. 2.7 и других), создайте пустое примечание (без текста) и перетащите на него из браузера нужную диаграмму.

Кроме примечаний, на диаграмму можно поместить также и текстовую область. С ее помощью можно, например, добавить к диаграмме заголовок.

Чтобы поместить на диаграмму текстовую область:

1. На панели управления нажмите кнопку Text Box.
2. Щелкните мышью внутри диаграммы, чтобы поместить туда текстовую область.
3. Выделив эту область, введите в нее текст.

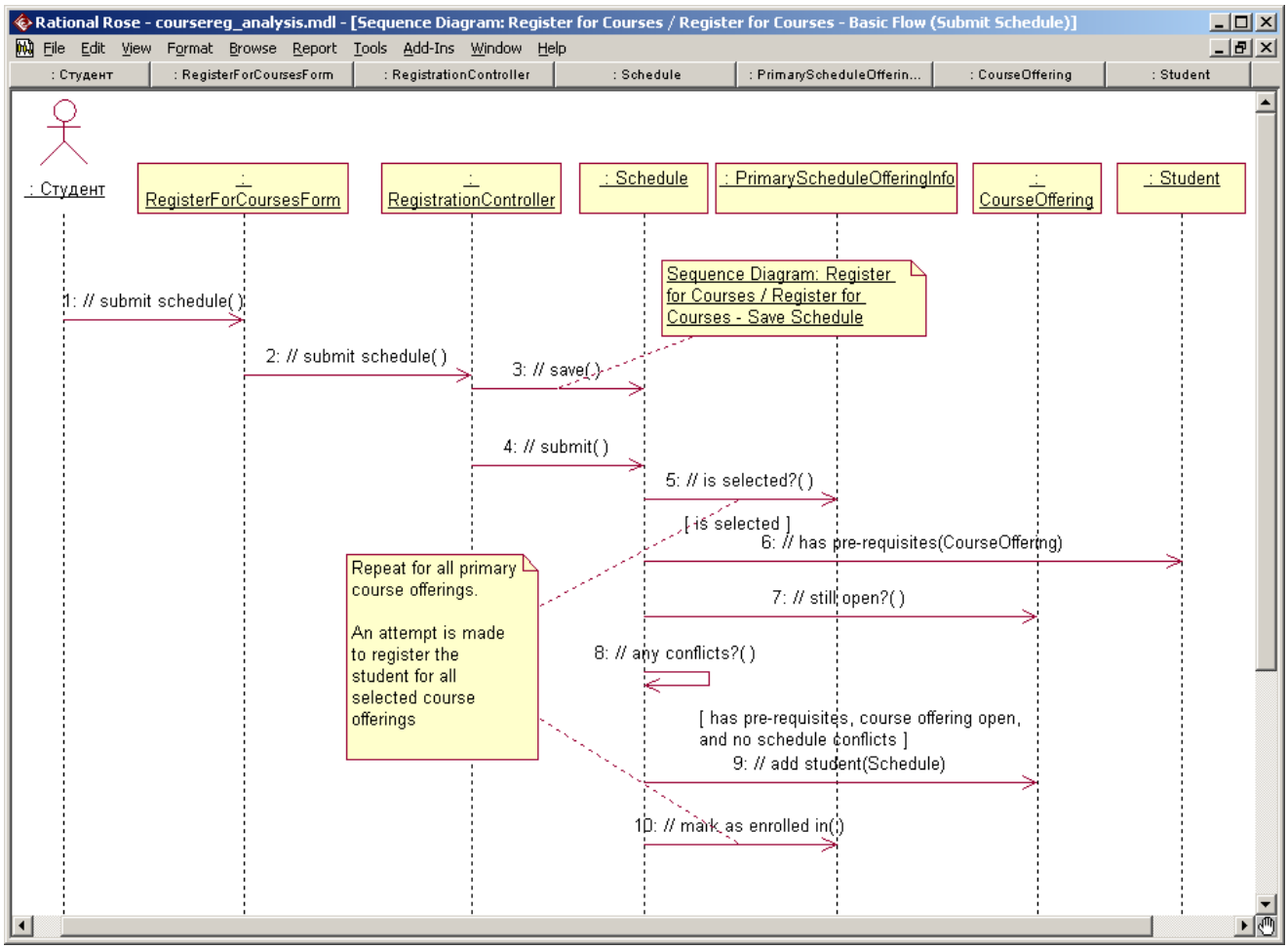

Рис. 2.14 Диаграмма последовательности Register for Courses - Basic Flow (Submit Schedule)

Создание кооперативной диаграммы

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

Определение обязанностей (responsibilities), атрибутов и ассоциаций классов

Обязанность (responsibility) – действие, которое объект обязан выполнять по запросу других объектов. Обязанность преобразуется в одну или более операций класса на шаге проектирования. Обязанности определяются исходя из сообщений на диаграммах взаимодействия и документируются в классах в виде операций «анализа», которые появляются там автоматически в процессе построения диаграмм взаимодействия (соотнесения сообщений с операциями).

Так, диаграмма классов VOPC (classes only) (рис. 2.9) после построения диаграмм взаимодействия в упражнении 8 должна принять следующий вид (рис. 2.15):

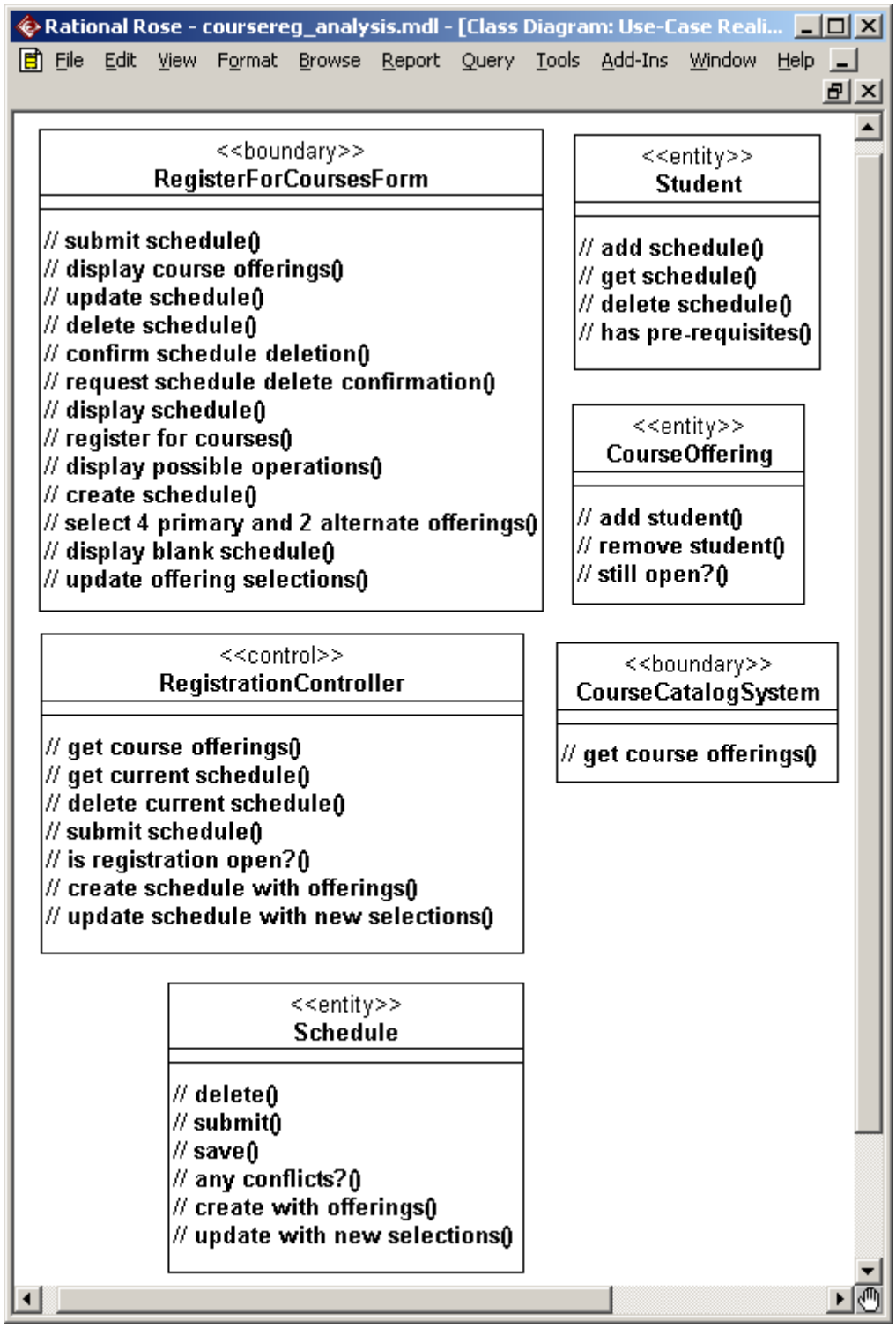

Рис. 2.15 Диаграмма классов VOPC (classes only) с операциями “анализа”

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

Упражнение 15. Добавление атрибутов к классам

Настройка

1. В меню модели выберите пункт Tools > Options.
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 на классе Student.
2. В открывшемся меню выберите пункт New Attribute.
3. Введите новый атрибут address
4. Нажмите клавишу Enter.
5. Повторите шаги 1 – 4, добавив атрибуты name и studentID.
6. Добавьте атрибуты к классам CourseOffering и Shedule, как показано на рис. 2.16.

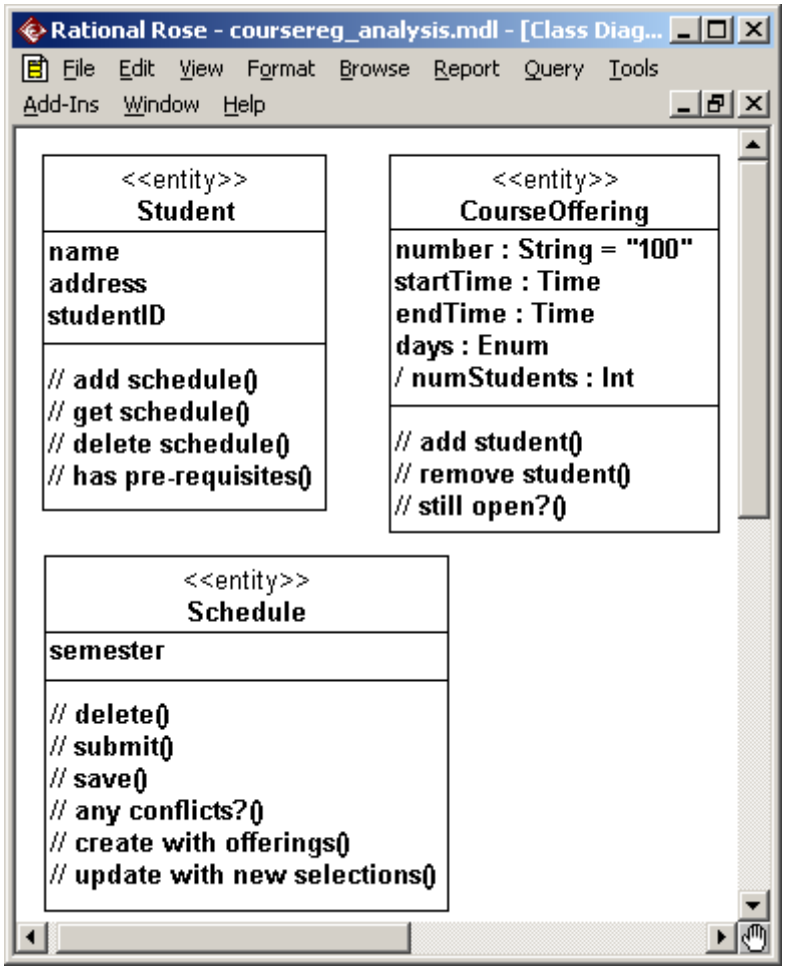

Рис. 2.16 Классы с операциями “анализа” и атрибутами

Связи между классами (ассоциации) определяются в два этапа:

1. Начальный набор связей определяется 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.
2. Анализируются и уточняются ассоциации между классами-сущностями. Задаются мощности ассоциаций, могут использоваться множественные ассоциации, агрегации, обобщения и ассоциации-классы.

Упражнение 16. Добавление связей

Добавим связи к классам, принимающим участие в варианте использования Register for Courses. Для отображения связей между классами построим три новых диаграмм классов в кооперации Register for Courses пакета Use-Case Realization - Register for Courses (рис. 2.17 – 2.19).

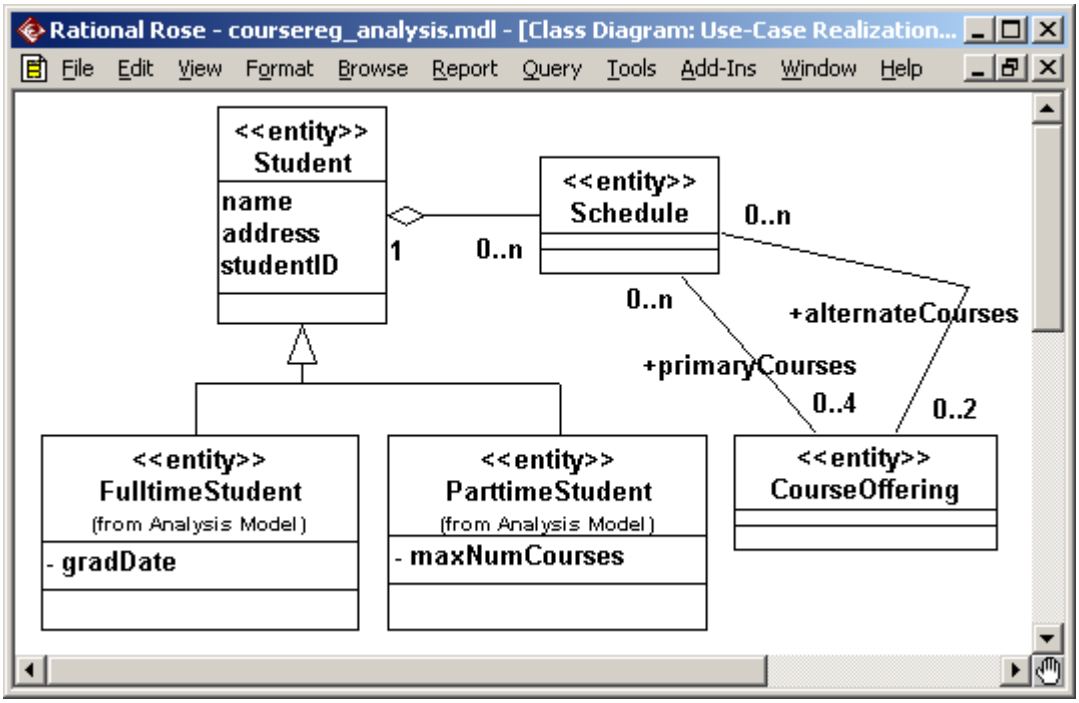

Рис. 2.17 Диаграмма Entity Classes (классы-сущности)

На рис. 2.17 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, а все графики студента за разные семестры. Между классами Schedule и CourseOffering введено две ассоциации, поскольку конкретный курс может входить в график студента в качестве основного (не более четырех курсов) и альтернативного (не более двух курсов). К классу Student добавлены два новых подкласса – FulltimeStudent (студент очного отделения) и ParttimeStudent (студент вечернего отделения).

На рис. 2.18 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», «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 и показано с помощью связи обобщения.

На рис. 2.19 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.

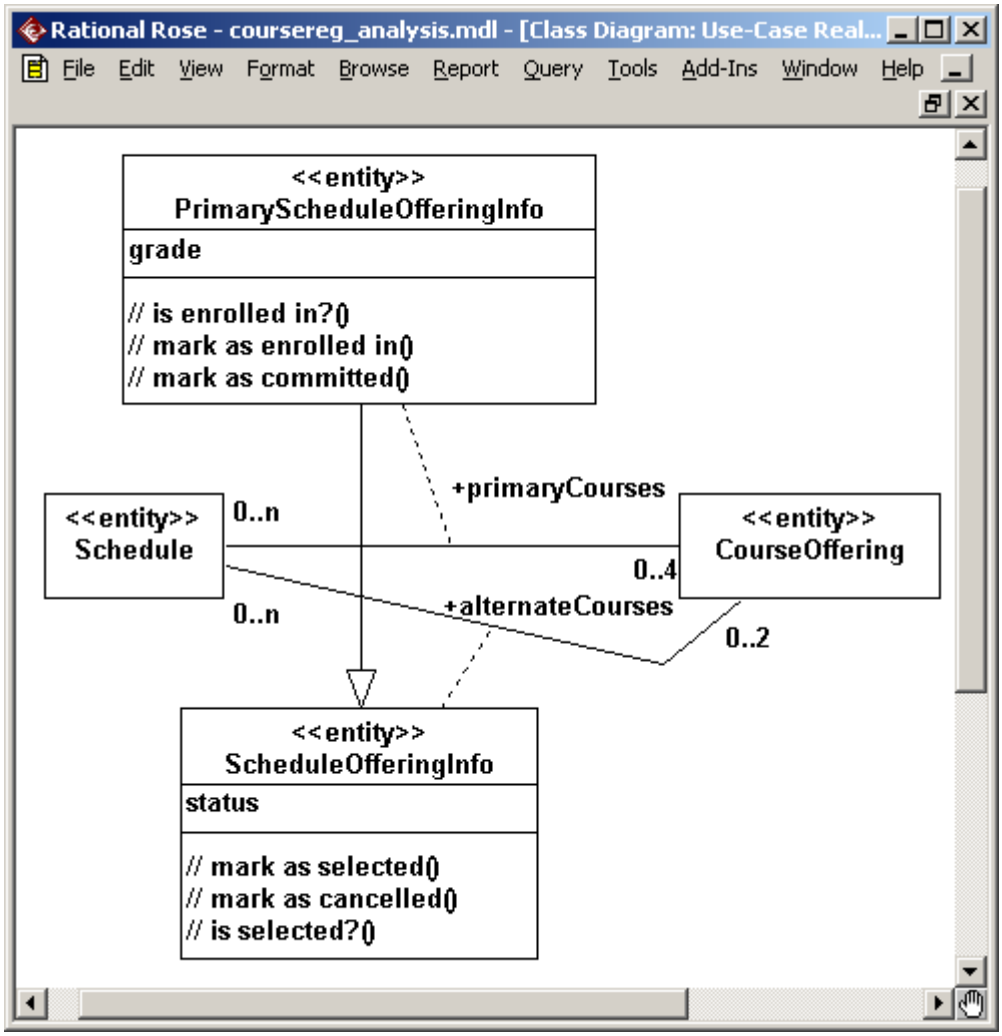

Рис. 2.18 Диаграмма CourseOfferingInfo (пример ассоциаций-классов)

Создание ассоциаций

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

1. Нажмите на панели инструментов кнопку Association.
2. Проведите мышью линию ассоциации от одного класса к другому.

Чтобы задать возможности навигации по ассоциации:

1. Щелкните правой кнопкой мыши на связи с того конца, на котором хотите показать стрелку.
2. В открывшемся меню выберите пункт Navigable.

Чтобы создать рефлексивную ассоциацию:

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

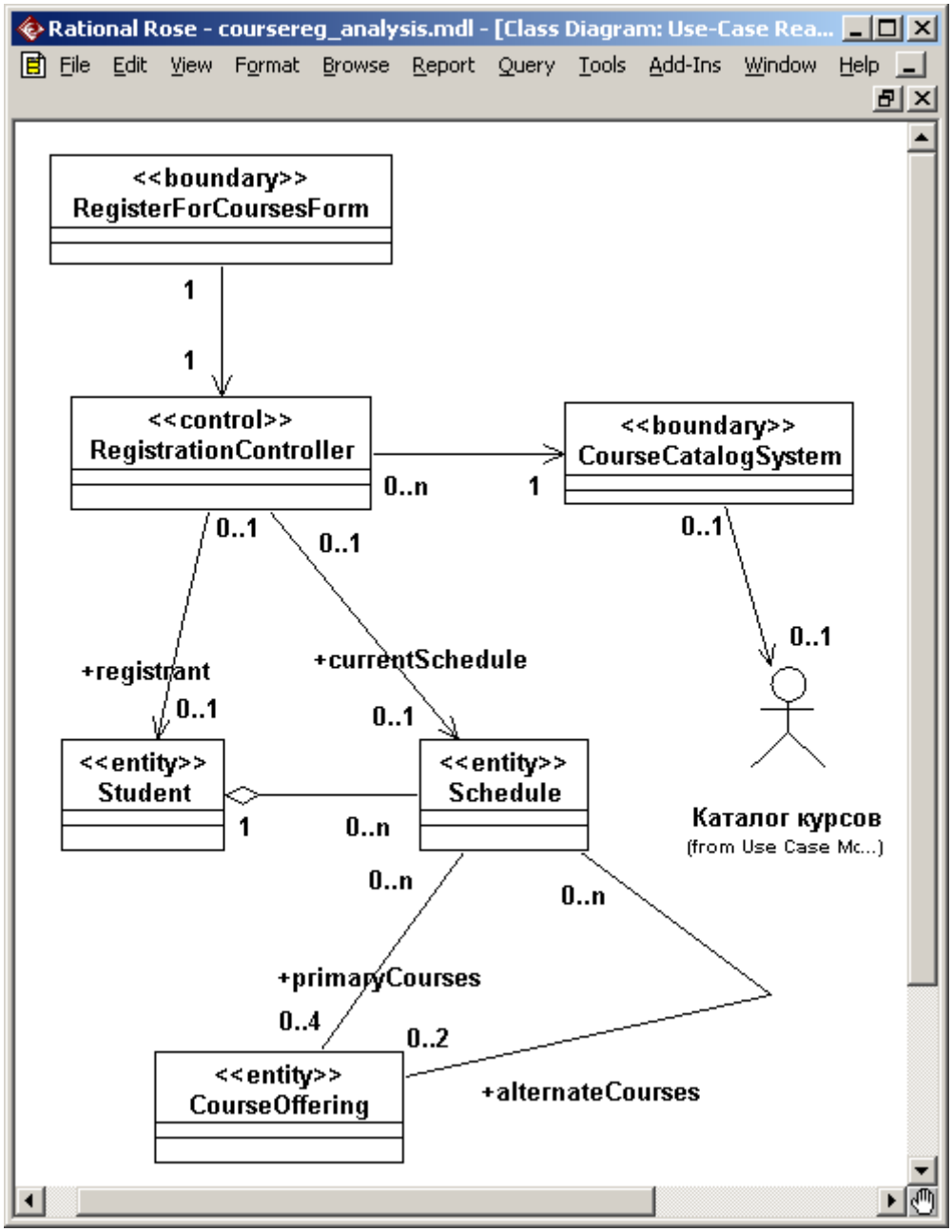

Рис. 2.19. Полная диаграмма классов VOPC (без атрибутов и операций)

Создание агрегаций

1. Нажмите кнопку Aggregation панели инструментов.
2. Проведите линию агрегации от класса-части к целому.

Чтобы поместить на диаграмму классов рефлексивную агрегацию:

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

Создание обобщений

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 для этого достаточно просто перетащить атрибуты или операции из одного класса в другой. Не забудьте удалить другую копию атрибута из второго подкласса, если он имеется.

Чтобы поместить обобщение на диаграмму классов:
1. Нажмите кнопку Generalization панели инструментов.
2. Проведите линию обобщения от подкласса к суперклассу.

Спецификации связей

Спецификации связей касаются имен ассоциаций, ролевых имена, множественности и классов ассоциаций.

Чтобы задать множественность связи:
1. Щелкните правой кнопкой мыши на одном конце связи.
2. В открывшемся меню выберите пункт Multiplicity.
3. Укажите нужную множественность.
4. Повторите то же самое для другого конца связи.

Чтобы задать имя связи:
1. Выделите нужную связь.
2. Введите ее имя.

Чтобы задать связи ролевое имя:
1. Щелкните правой кнопкой мыши на ассоциации с нужного конца.
2. В открывшемся меню выберите пункт role Name.
3. Введите ролевое имя.

Чтобы задать элемент связи (класс ассоциаций):
1. Откройте окно спецификации требуемой связи.
2. Перейдите на вкладку Detail.
3. Задайте элемент связи в поле Link Element.

Задание для самостоятельной работы

Выполнить анализ варианта использования Close Registration и построить соответствующие диаграммы взаимодействия и классов.

2.10 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.

2.10.1 Проектирование архитектуры системы

Проектирование архитектуры системы выполняется архитектором системы и включает в себя:
- идентификацию архитектурных решений и механизмов, необходимых для проектирования системы;
- анализ взаимодействий между классами анализа, выявление подсистем и интерфейсов;
- формирование архитектурных уровней;
- проектирование структуры потоков управления;
- проектирование конфигурации системы.

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 в подсистему.

Объединение классов в подсистемы осуществляется, исходя из следующих соображений:

- 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, обеспечение безопасности и защиты 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 в 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 документируется 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-посредник 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.

В качестве примера (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. 2.20 – 2.23. Классы DBCourseOfferring и CourseOfferringList отвечают за взаимодействие с БД каталога курсов в рамках JDBC. Объект CourseCatalogSystem Client на рис. 2.23 может принадлежать либо классу CloseRegistrationController, либо классу RegistrationController, в зависимости от того, в каком из вариантов использования запрашивается каталог курсов.

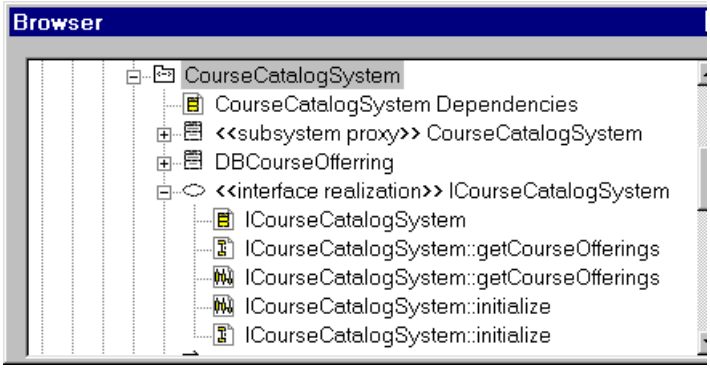

Рис. 2.20 Структура пакета CourseCatalogSystem

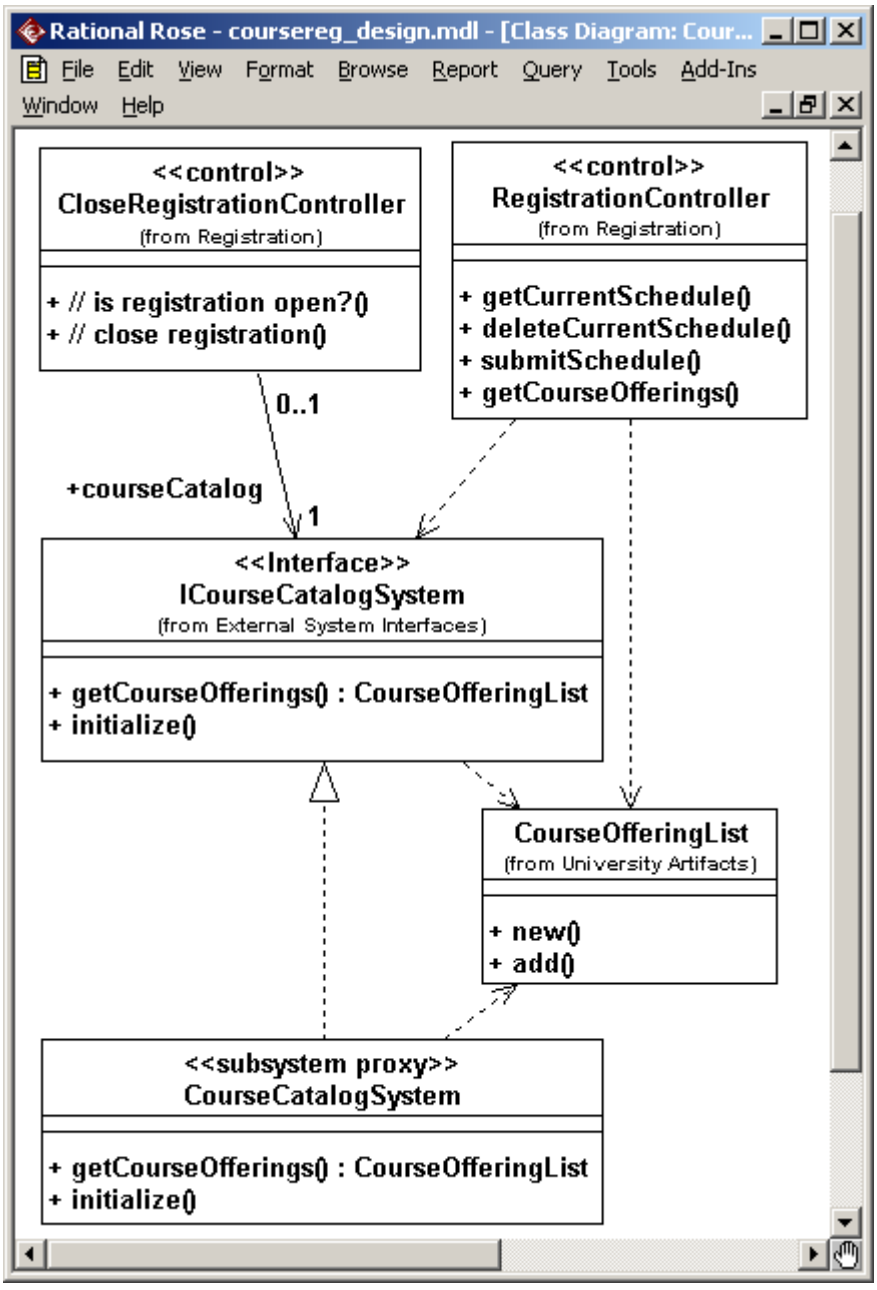

Рис. 2.21. Контекст подсистемы CourseCatalogSystem с точки зрения архитектора

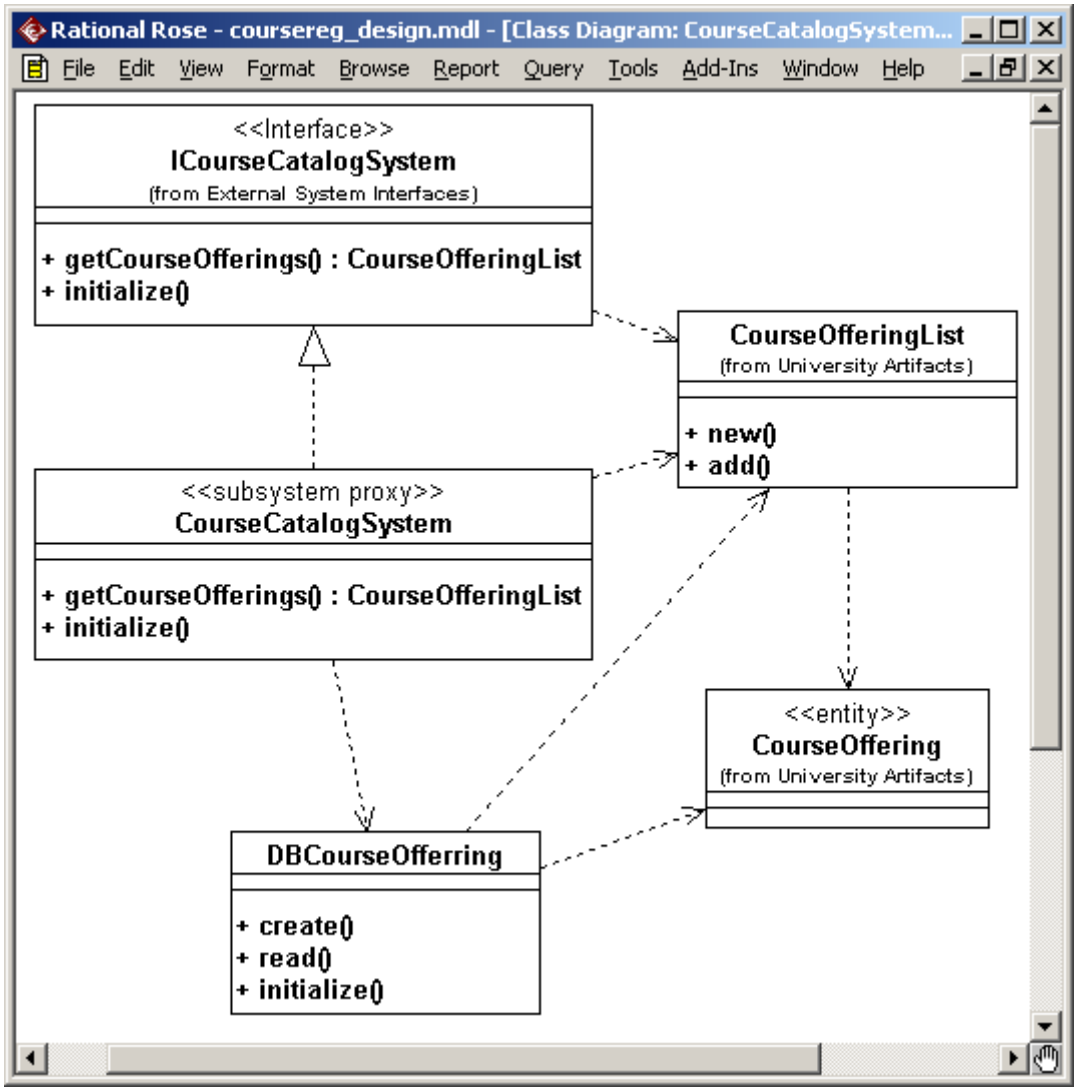

Рис. 2.22. Диаграмма классов подсистемы CourseCatalogSystem с точки зрения проектировщика

Формирование архитектурных уровней

В процессе анализа было принято предварительное решение о выделении архитектурных уровней. 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 выше, зависят от сложности предметной области и среды реализации. Подуровни могут формироваться, исходя из следующих критериев:

- 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. 2.24.

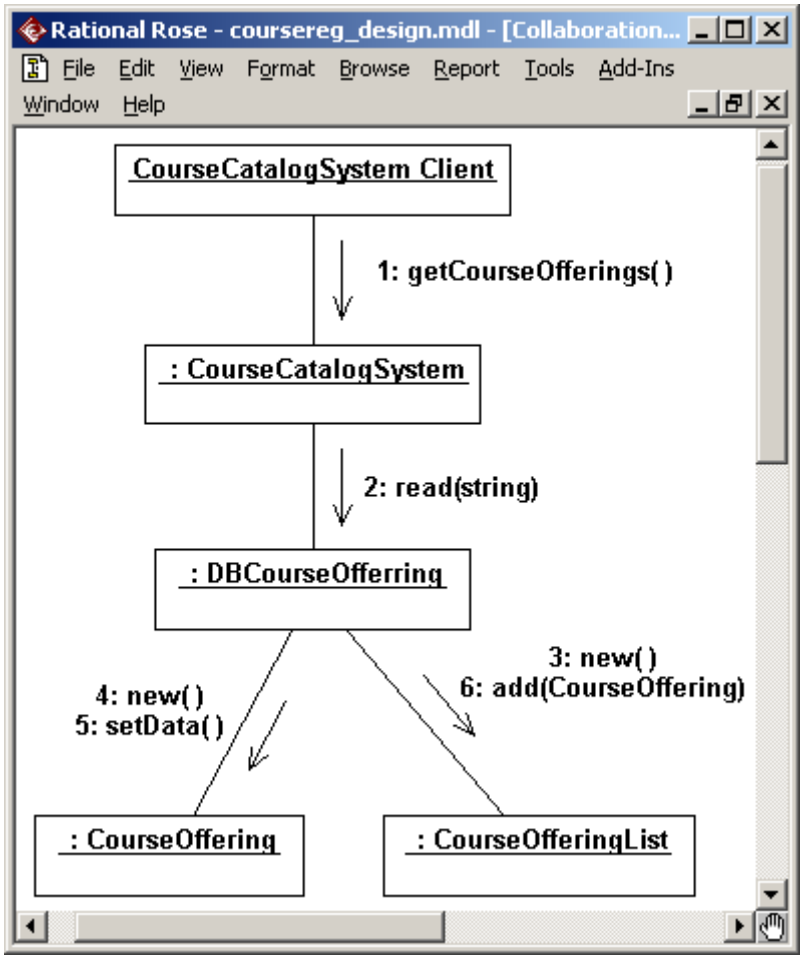

Рис. 2.23. Кооперативная диаграмма, описывающая реализацию операции интерфейса getCourseOfferings

Данное представление отражает 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 включает интерфейсы подсистем (классы со стереотипом <<Interface>>), а пакет University Artifacts – все классы-сущности.

2.10.2 Моделирование распределенной конфигурации системы

Распределенная 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: сетевая конфигурация системы регистрации (без процессов) (рис. 2.25).

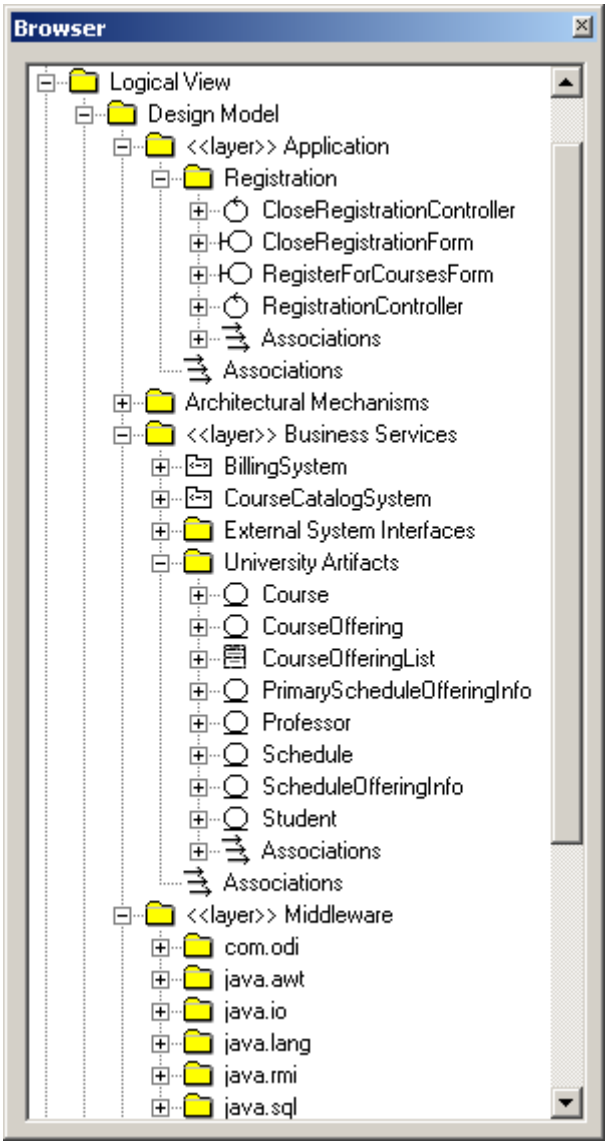

Рис. 2.24. Представление структуры модели в процессе проектирования

Распределение процессов по узлам сети производится с учетом следущих факторов:

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

Пример распределения процессов по узлам приведен на рис. 2.26.

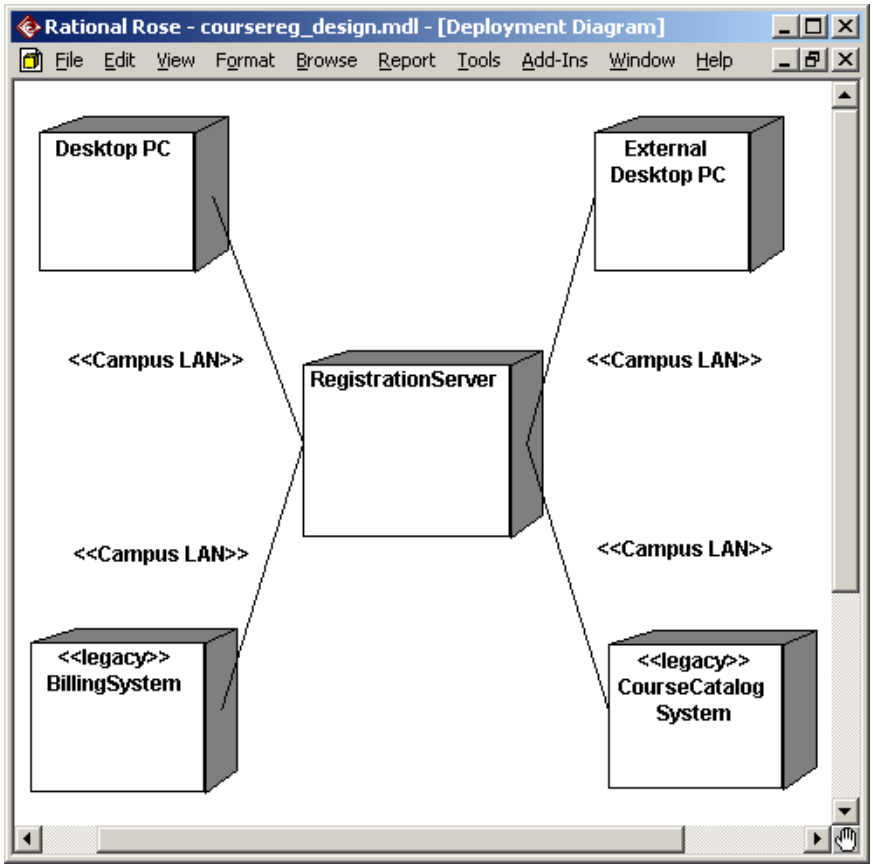

Рис. 2.25 Сетевая конфигурация системы регистрации

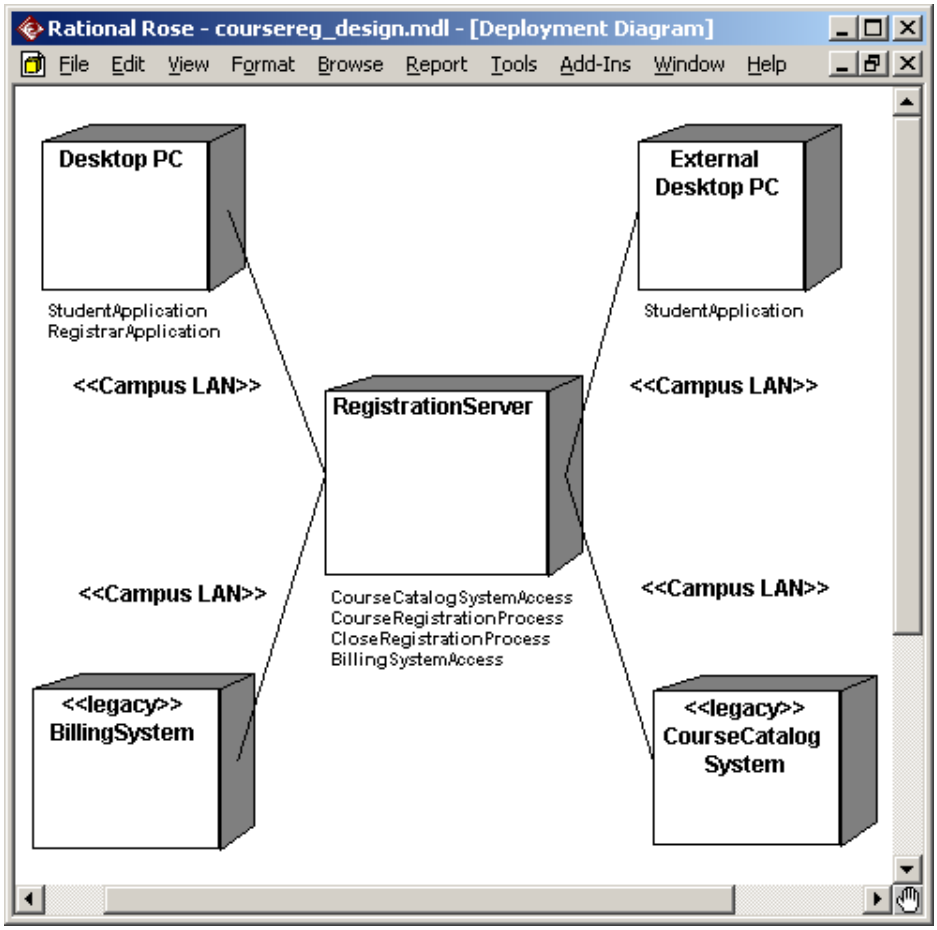

Рис. 2.26 Сетевая конфигурация системы регистрации с распределением процессов по узлам

Упражнение 17. Создание диаграммы размещения системы регистрации

Чтобы открыть диаграмму размещения, надо дважды щелкнуть мышью на представлении Deployment View (представлении размещения) в браузере.

Чтобы поместить на диаграмму процессор:

1. На панели инструментов диаграммы нажмите кнопку Processor.
2. Щелкните на 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 - это его физическое описание. Оно может, в частности, включать скорость процессора и объем памяти.

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

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

Чтобы назначить процессору стереотип:

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

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.

Чтобы показать планирование на диаграмме:

1. Щелкните правой кнопкой мыши на процессоре.
2. В открывшемся меню выберите пункт Show Scheduling.

Чтобы добавить связь на диаграмму:

1. На панели инструментов нажмите кнопку Connection.
2. Щелкните на узле диаграммы.
3. Проведите линию связи к другому узлу.

Чтобы назначить связи стереотип:

1. Откройте окно спецификации связи.
2. Перейдите на вкладку General.
3. Введите стереотип в поле Stereotype (Стереотип).

Чтобы добавить процесс:

1. Щелкните правой кнопкой мыши на процессоре в браузере.
2. В открывшемся меню выберите пункт New > Process.
3. Введите имя нового процесса.

Чтобы показать процессы на диаграмме:

1. Щелкните правой кнопкой мыши на процессоре.
2. В открывшемся меню выберите пункт Show Processes.

2.10.3 Проектирование классов

Каждый граничный класс преобразуется в некоторый 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.

Что касается управляющих классов, то 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.).

Обязанности классов, определенные в процессе анализа, преобразуются в операции. Каждой операции присваивается имя, характеризующее ее результат. Определяется полная сигнатура операции: operationName(parameter:class,…):returnType. Создается краткое описание операции, включая смысл всех ее параметров. Определяется видимость операции: public, private, protected. Определяется область действия (scope) операции: экземпляр или классификатор.

Определяются (уточняются) атрибуты классов:

- Кроме имени, задается тип и значение по умолчанию (необязательное): attributeName:Type = Default;
- Учитываются соглашения по именованию атрибутов, принятые в проекте и языке реализации;
- Задается видимость атрибутов: public, private, protected;
- При необходимости определяются производные (вычисляемые) атрибуты.

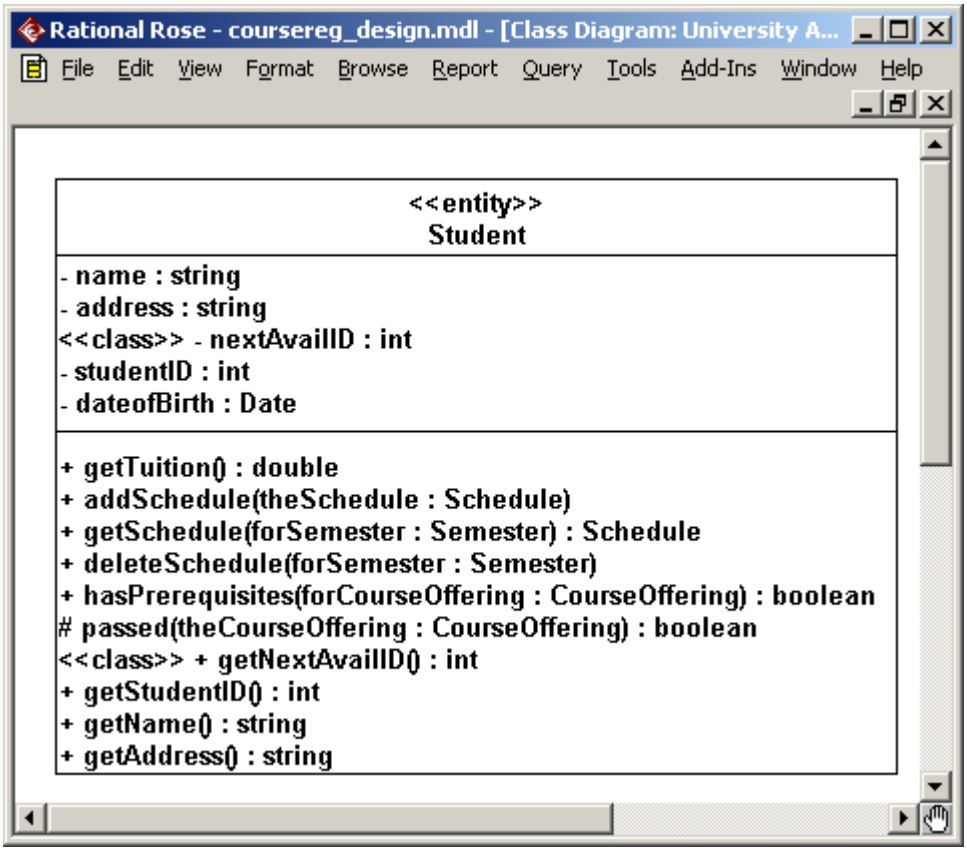

Рис. 2.27 Класс Student с полностью определенными операциями и атрибутами

Упражнение 18. Определение атрибутов и операций для класса Student

Чтобы задать тип данных, значение по умолчанию и видимость атрибута:

1. Щелкните правой кнопкой мыши на атрибуте в браузере.
2. В открывшемся меню выберите пункт Open Specification.
3. Укажите тип данных в раскрывающемся списке типов или введите собственный тип данных.
4. В поле Initial Field (Первоначальное значение) введите значение атрибута по умолчанию.
5. В поле Export Control выберите видимость атрибута: Public, Protected, Private или Implementation. По умолчанию видимость всех атрибутов соответствует Private.

Чтобы изменить нотацию для обозначения видимости:

1. В меню модели выберите пункт Tools > Options.
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 уже существующие диаграммы.

Чтобы задать тип возвращаемого значения, стереотип и видимость операции:

1. Щелкните правой кнопкой мыши на 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. В поле Export Control укажите значение видимости операции: Public, Protected, Private или Implementation. По умолчанию видимость всех операций установлена в public.

Чтобы добавить к операции аргумент:

1. Откройте окно спецификации операции.
2. Перейдите на вкладку Detail.
3. Щелкните правой кнопкой мыши в области аргументов, в открывшемся меню выберите Insert.
4. Введите имя аргумента.
5. Щелкните на колонке Data type и введите туда тип данных аргумента.
6. Если надо, щелкните на колонке default и введите значение аргумента по умолчанию.

Моделирование состояний для классов

Если некоторый объект всегда одинаково реагирует на событие, то он считается *независимым от состояния* по отношению к этому событию. В отличие от него, *зависимые от состояния* объекты по-разному реагируют на одно и то же событие в зависимости от своего состояния. Обычно в экономических ИС содержится очень мало объектов, зависимых от состояния, а системы управления технологическими процессами (системы реального времени) зачастую содержат множество таких объектов.

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:

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).

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.

3. *Идентификация переходов между состояниями.* Переходы вызываются событиями. Таким образом, состояния Unassigned и Assigned соединяются двумя переходами (рис. 2.28).

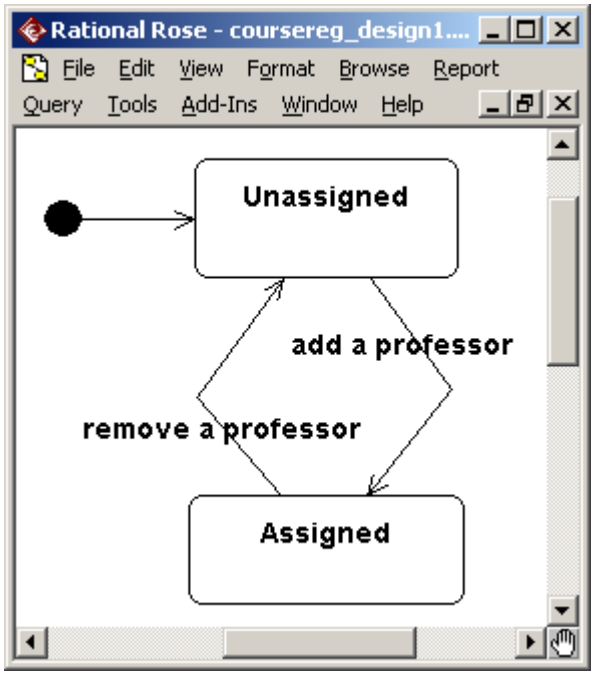

Рис. 2.28. Переходы между состояниями

Дальнейшая детализация поведения объекта CourseOffering приведет к построению диаграммы состояний, показанной на рис. 2.29. На данной диаграмме использованы такие возможности моделирования состояний, как композитные состояния (composite state) и историческое состояние (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.

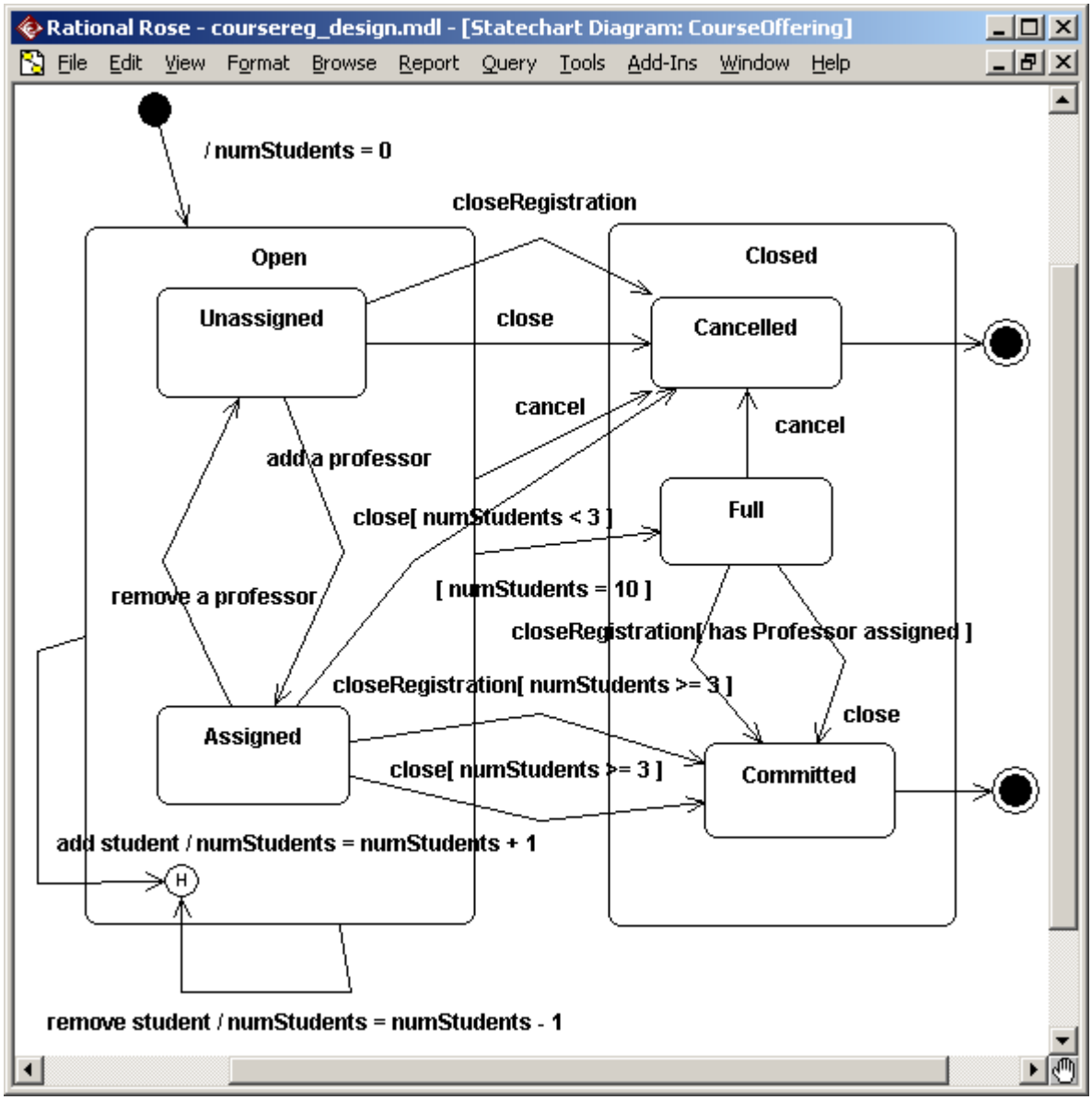

Рис. 2.29. Диаграмма состояний с композитными состояниями

Упражнение 19. Создание диаграммы состояний для класса CourseOffering

Для создания диаграммы состояний:

1. Щелкните правой кнопкой мыши в браузере на нужном классе.
2. В открывшемся меню выберите пункт New > Statechart Diagram.

Чтобы добавить состояние:

1. На панели инструментов нажмите кнопку State
2. Щелкните мышью на диаграмме состояний в том месте, куда хотите его поместить.

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

Чтобы добавить деятельность:

1. Откройте окно спецификации требуемого состояния.
2. Перейдите на вкладку Detail.
3. Щелкните правой кнопкой мыши на окне Actions.
4. В открывшемся меню выберите пункт Insert.
5. Дважды щелкните на новом действии.
6. Введите действие в поле Actions.
7. В окне When укажите Do, чтобы сделать новое действие деятельностью.

Чтобы добавить входное действие, в окне When укажите On Entry.

Чтобы добавить выходное действие, в окне When укажите On Exit.
Чтобы послать событие:
1. Откройте окно спецификации требуемого состояния.
2. Перейдите на вкладку Detail.
3. Щелкните правой кнопкой мыши на окне Actions.
4. В открывшемся меню выберите пункт Insert.
5. Дважды щелкните на новом действии.
6. В качестве типа действия укажите Send Event.
7. В соответствующие поля введите событие (event), аргументы (arguments) и целевой объект (Target).

Чтобы добавить переход:
1. Нажмите кнопку Transition панели инструментов.
2. Щелкните мышью на состоянии, откуда осуществляется переход.
3. Проведите линию перехода до того состояния, где он завершается.
Чтобы добавить рефлексивный переход:
1. Нажмите кнопку Transition to Self панели инструментов.
2. Щелкните на том состоянии, где осуществляется рефлексивный переход.

Чтобы добавить событие, его аргументы, ограждающее условие и действие:
1. Дважды щелкните на переходе, 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.
Чтобы отправить событие:
1. Дважды щелкните на переходе, 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 указания начального или конечного состояния:
1. На панели инструментов нажмите кнопку Start State или End State.
2. Щелкните мышью на 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. 2.30. 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, и количество графиков относительно мало по сравнению с количеством конкретных курсов.

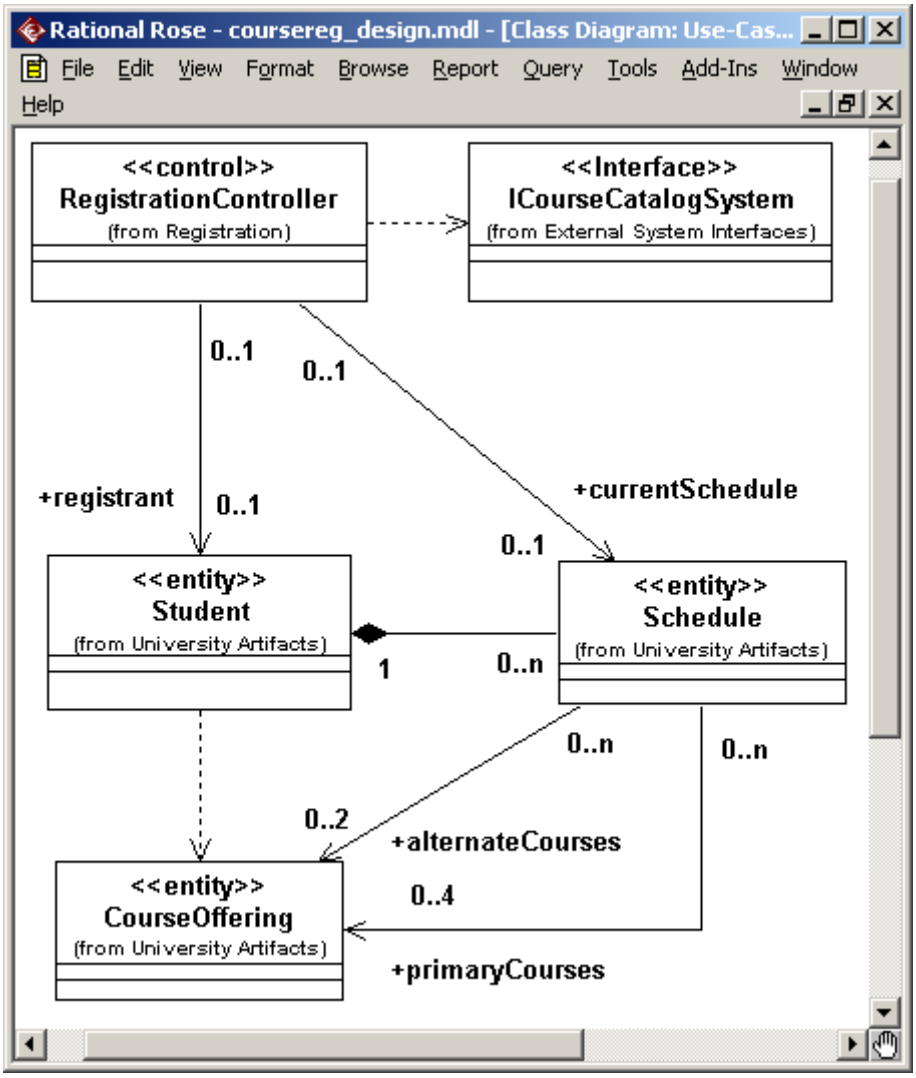

Рис. 2.30. Пример преобразования ассоциаций и агрегаций

Связи обобщения могут преобразовываться в ситуациях с так называемой метаморфозой подтипов. Например, в случае с системой регистрации студент может переходить с очной формы обучения на вечернюю, т.е., объект Student может менять свой подтип. При таком изменении придется модифицировать описание объекта в системе. Чтобы избежать этой модификации и тем самым повысить устойчивость системы, иерархия наследования реализуется с помощью классификации, как показано на рис. 2.31.

Чтобы преобразовать агрегацию в композицию:

1. Щелкните правой кнопкой мыши на том конце агрегации, который упирается в класс-часть (на рис.2.30 – Schedule).
2. В открывшемся меню выберите пункт Containment.
3. Укажите метод включения By Value.

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.

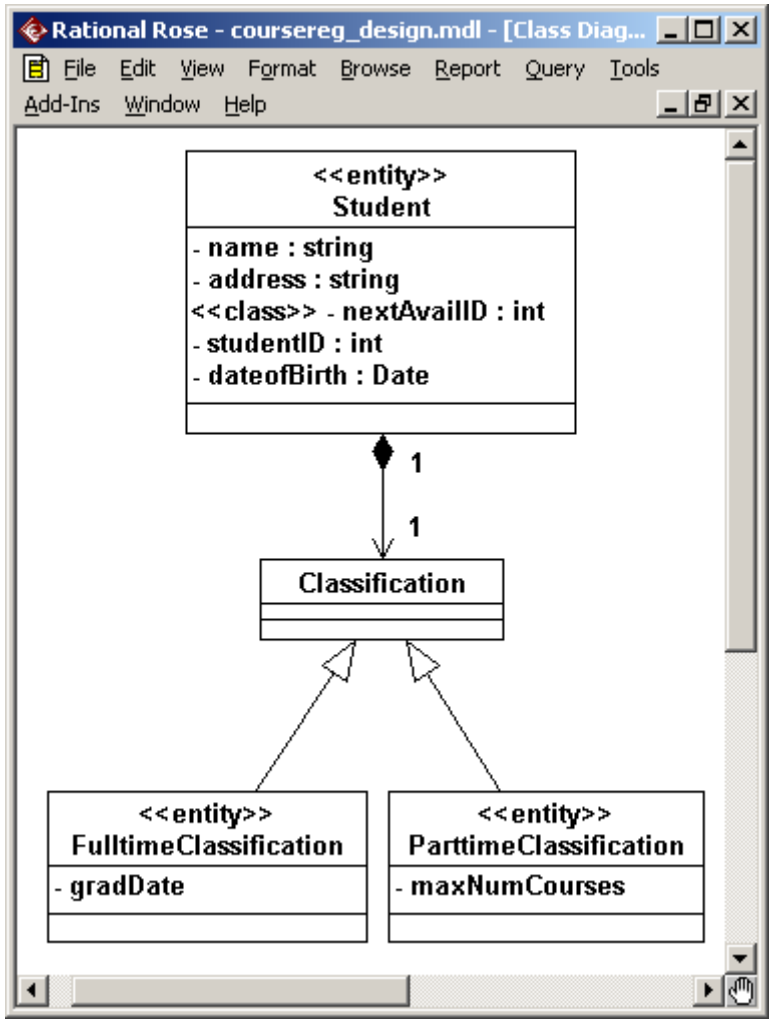

Рис. 2.31. Преобразование обобщения

2.10.4 Проектирование баз данных

Проектирование реляционных баз данных выполняется с использованием средства Data Modeler. Его работа основана на известном механизме отображения объектной модели в реляционную. Результатом является построение диаграммы «сущность-связь» и последующая генерация описания БД на SQL.

Упражнение 20. Проектирование реляционной базы данных

Проектирование БД состоит из следующих шагов:
Создание нового компонента – базы данных:

1. Щелкните правой кнопкой мыши на представлении компонентов.
2. В открывшемся меню выберите пункт Data Modeler > New > Database.
3. Откройте окно спецификации вновь созданного компонента DB_0 и в списке Target выберите Oracle 8.x.

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 на атрибуте studentID.

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

Создание схемы БД:

1. Щелкните правой кнопкой мыши на пакете University Artifacts.
2. В открывшемся меню выберите пункт Data Modeler > Transform to Data Model.
3. В появившемся окне в списке Target Database укажите DB_0 и нажмите OK. В результате в логическом представлении появится новый пакет Schemas.
4. Откройте пакет Schemas и щелкните правой кнопкой мыши на пакете <<Schema>> S_0.
5. В открывшемся меню выберите пункт Data Modeler > New > Data Model Diagram.
6. Откройте пакет, затем откройте вновь созданную диаграмму «сущность-связь» NewDiagram и перенесите на нее все классы-таблицы, находящиеся в пакете <<Schema>> S_0. Получившаяся диаграмма показана на рис. 2.32.

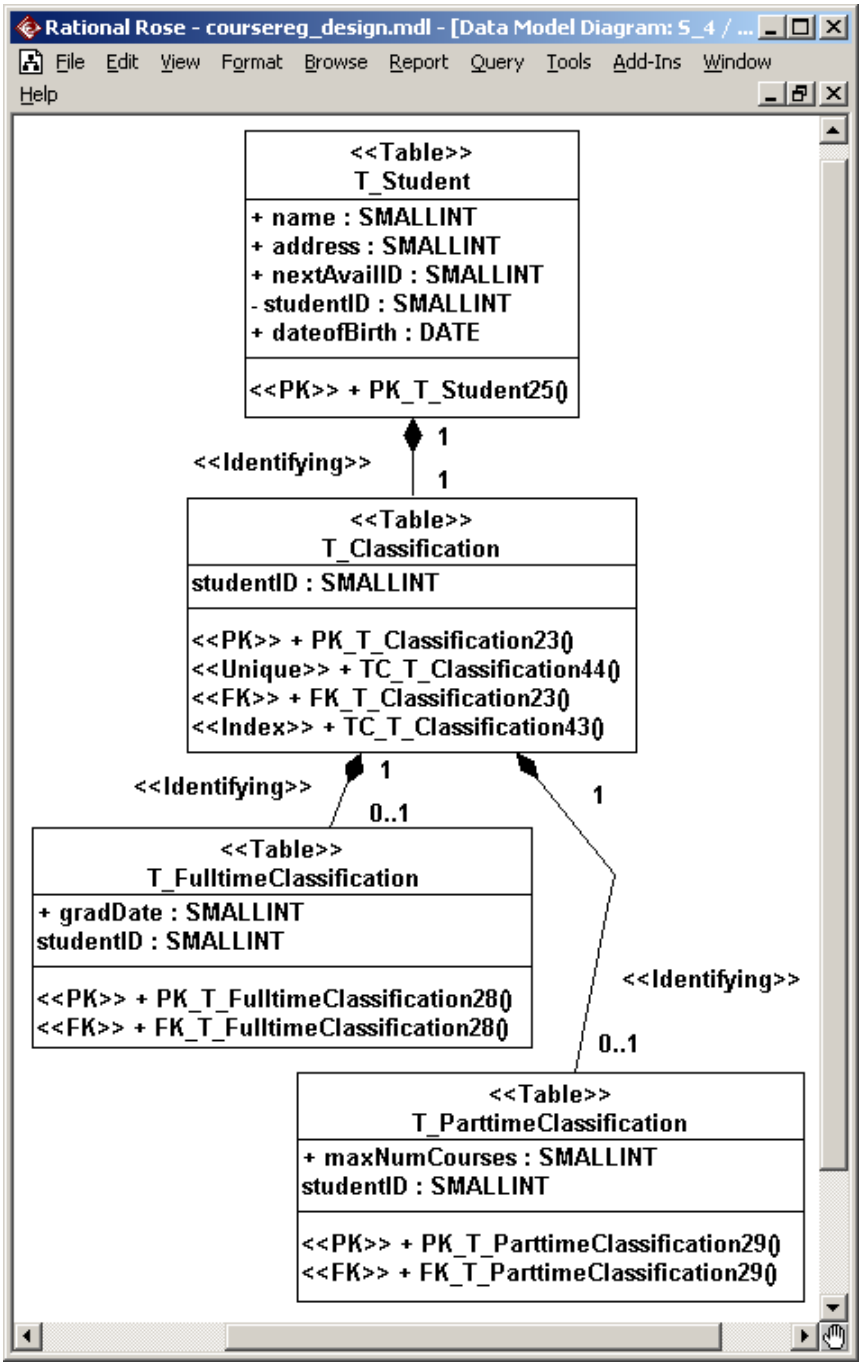

Рис. 2.32. Диаграмма «сущность-связь»

Генерация описания БД на SQL:

1. Щелкните правой кнопкой мыши на пакете <<Schema>> S_0.
2. В открывшемся меню выберите пункт Data Modeler > Forward Engineer.
3. В открывшемся окне мастера Forward Engineering Wizard нажмите Next.
4. Отметьте все флажки генерации DDL и нажмите Next.
5. Укажите имя и расположение текстового файла с результатами генерации и нажмите Next.
6. После завершения генерации откройте созданный текстовый файл и просмотрите результаты.

2.11. Реализация системы

2.7.1 Создание компонентов

В Rational Rose диаграммы компонентов создаются в представлении компонентов системы. Отдельные компоненты можно создавать непосредственно на диаграмме, или перетаскивать их туда из браузера.

Упражнение 21. Создание компонентов

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

Создание диаграммы компонентов:

1. Дважды щелкните мышью на главной диаграмме компонентов в представлении компонентов.
2. На панели инструментов нажмите кнопку Package Specification.
3. Поместите спецификацию пакета на диаграмму.
4. Введите имя спецификации пакета Student и укажите в окне спецификации язык C++.
5. На панели инструментов нажмите кнопку Package Body.
6. Поместите тело пакета на диаграмму.
7. Введите имя тела пакета Student и укажите в окне спецификации язык C++.
8. На панели инструментов нажмите кнопку Dependency.
9. Проведите линию зависимости от тела пакета к спецификации пакета.

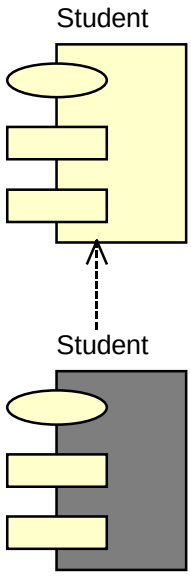

Рис. 2.33. Диаграмма компонентов

Соотнесение классов с компонентами:

1. В логическом представлении браузера найдите класс Student.
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.

2.11.2 Генерация кода

Процесс генерации кода состоит из шести основных шагов:

1. Проверка корректности модели.
2. Установка свойств генерации кода.
3. Выбор класса, компонента или пакета.
4. Генерация кода.

Для проверки модели:

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

К наиболее распространенным ошибкам относятся такие, например, как сообщения на диаграмме последовательности или кооперативной диаграмме, не соотнесенные с операцией, либо объекты этих диаграмм, не соотнесенные с классом.

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.

Чтобы обнаружить нарушение правил доступа:

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

Можно установить несколько параметров генерации кода для классов, атрибутов, компонентов и других элементов модели. Этими свойствами определяется способ генерации программ. 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.

Иногда нужно изменить свойства генерации кода для одного класса, атрибута, одной операции и т.д. Для этого отройте окно спецификации элемента модели. Выберите вкладку языка (C++, Java, …) и измените свойства здесь. Все изменения, вносимые в окне спецификации элемента модели, оказывают влияние только на этот элемент.

При генерации кода за один раз можно создать класс, компонент или целый пакет. Код генерируется с помощью диаграммы или браузера. При генерации кода из пакета можно выбрать или пакет логического представления на диаграмме классов, или пакет представления компонентов на диаграмме компонентов. При выборе пакета логического представления генерируются все классы этого пакета. При выборе пакета представления компонентов генерируются все компоненты этого пакета.

После выбора класса или компонента на диаграмме выберите в меню соответствующий вариант генерации кода. Сообщения об ошибках, возникающих в процессе генерации кода, будут появляться в окне журнала.

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.

Упражнение 22. Генерация кода C++.

1. Откройте диаграмму компонентов системы.
2. Выберите все объекты на диаграмме компонентов.
3. Выберите Tools > C++ > Code Generation в меню.
4. Выполните генерацию кода.
5. Просмотрите результаты генерации (меню Tools > C++ > Browse Header и Tools > C++ > Browse Body).