

ТЕМА 7




1. Диаграммы Sequence и Collaboration в Rational Rose.
2. Диаграмма Sequence в Rational XDE.

7.1 Построение диаграмм Sequence и Collaboration в Rational Rose

После того как в системе изучено поведение каждого объекта, необходимо точно представить взаимодействие этих объектов между собой, определить клиентов, серверы и порядок обмена сообщениями между ними. Диаграмма *Sequence* позволяет получить отражение процесса обмена сообщениями во времени.

В течение работы системы объекты, являющиеся клиентами, посылают друг другу различные сообщения, а объекты-серверы обрабатывают их. В простейшем случае можно рассматривать сообщение как вызов метода какого-либо класса, в более сложных случаях сервер имеет обработчик очереди сообщений, и сообщения им обрабатываются асинхронно, т.е. сервер накапливает несколько сообщений в очереди, если не может обработать их сразу.

Диаграмма взаимодействия создается одним из способов, которые были предложены в предыдущих разделах. Выбор какого-либо из них повлечет за собой открытие окна, в котором предложено два типа диаграмм: *Sequence* и *Collaboration*. После активизации диаграммы *Sequence* станет доступным набор ее инструментов.

Рассмотрим построение диаграммы Sequence на примере сценария работы кладовщика с карточкой товара и накладной (рис. 7.1), в нем использованы основные инструменты Sequence. Значок  – *Object* (объект) позволяет включить объект в диаграмму. Каждый из них является реализацией какого-нибудь класса, поэтому в объекте можно указать соответствующий ему класс. Для повышения наглядности в примере использованы стереотипы классов. Кладовщик представлен как *business worker*, Накладная – как *business entity*, Карточка товара – как *business entity*. Значок  – *Message* (сообщение) предназначен для передачи сообщения от одного объекта к другому. Классы должны позволять отправку или прием сообщений. Инструмент  – *Message to self* (сообщение самому себе) показывает, что отправитель сообщения является одновременно и его получателем. В этом случае объект выполняет функции и сервера, и клиента. Для реализации объектов в окне параметров доступна опция *Persistence*, отражающая время жизни объекта. Это время от его создания до уничтожения. Обычно объекты существуют в пределах их области видимости и автоматически уничтожаются системой, когда выходят за эту область. Таким образом, можно настроить следующие параметры жизни объекта:

- *Persistent* – область видимости объекта превышает время жизни.

- *Static* – элемент существует на всем протяжении работы программы.
- *Transient* – время жизни объекта и области видимости совпадают, этот вариант присваивается по умолчанию и подходит в случае нашего примера.

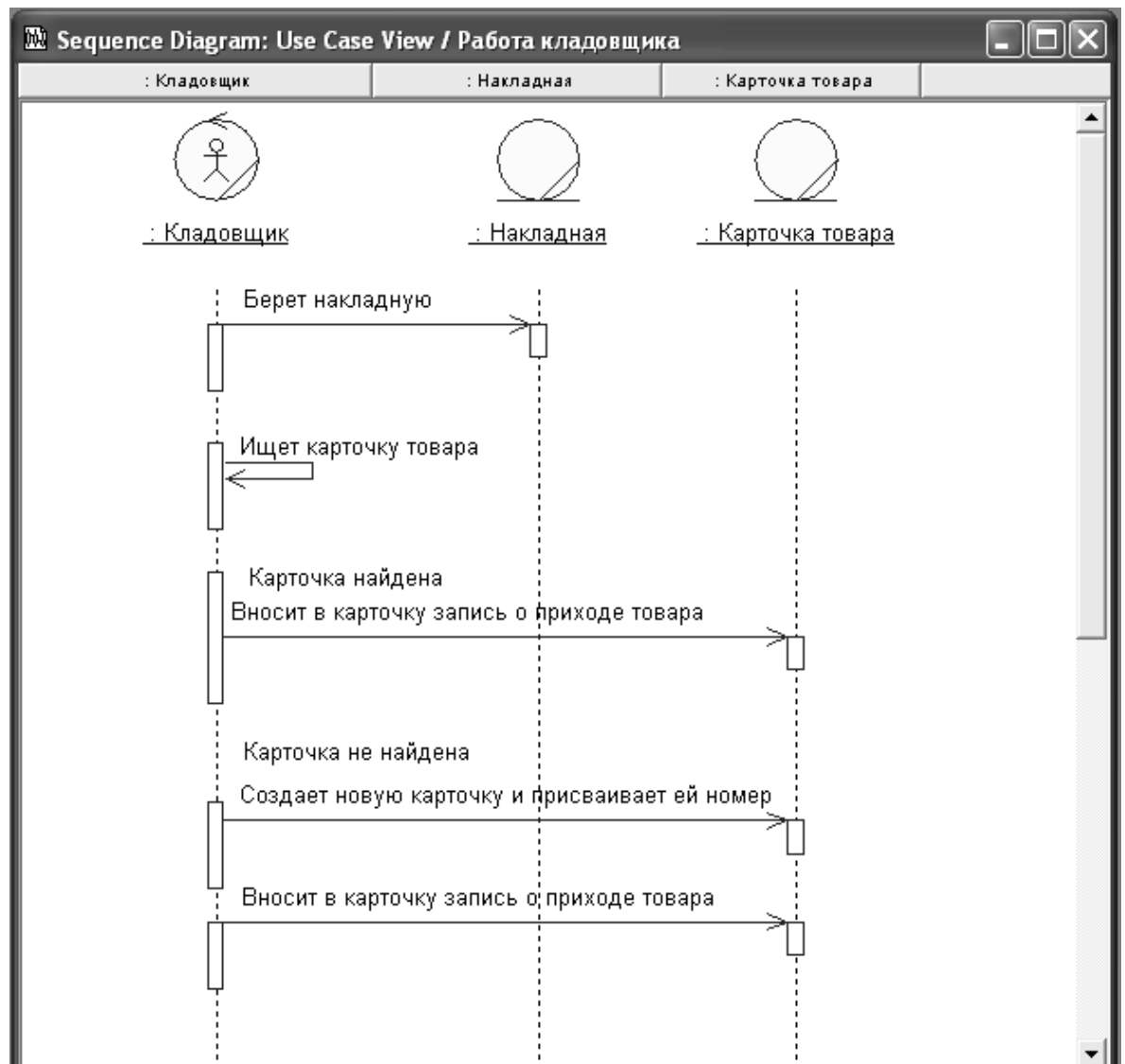


Рисунок 7.1 – Пример работы кладовщика с карточкой товара и накладной


Горизонтальные линии на рис. 7.1, проведенные от объекта к объекту, означают передачу сообщений между ними. Свойства задаются в окне спецификации с помощью двух групп кнопок.




1. *Synchronization* определяет порядок обмена сообщениями и может быть выбрана из следующих вариантов:

- *Simple* – простая посылка сообщения.
- *Synchronous* – операция происходит только в том случае, когда клиент посылает сообщение, а сервер может принять сообщение клиента.
- *Timeout* – клиент отказывается от выдачи сообщения, если сервер в течение определенного времени не может его принять.

- *Balking* – операция происходит только в том случае, когда сервер готов немедленно принять сообщение, иначе клиент не выдает сообщение.
 - *Procedure Call* – клиент вызывает процедуру сервера и полностью передает ему управление.
 - *Return* – возврат из процедуры.
 - *Asynchronous* – клиент выдает сообщение, и, не ожидая ответа сервера, продолжает выполнение своего программного кода.
2. *Frequency* определяет частоту обмена сообщениями:
- *Periodic* – сообщения поступают от клиента с заданной периодичностью.
 - *Aperiodic* – сообщения поступают от клиента нерегулярно.

Второй тип диаграмм взаимодействия – это *Collaboration*. Данная диаграмма отличается от *Sequence* тем, что здесь не акцентируется внимание на последовательности передачи сообщений, а отражается наличие взаимосвязей между клиентами и серверами вообще. Поскольку на *Collaboration* для демонстрации сообщений не применяется временная шкала, диаграмма получается более компактной и оптимально подходит для представления взаимодействий сразу всех объектов. Однако такое представление является мгновенным снимком системы в некотором состоянии, так как объекты создаются и уничтожаются на всем протяжении работы программы. В связи с этим появляются такие понятия, как время жизни и область видимости объектов.

Удобной возможностью работы в *Rational Rose* является то, что на основе *Sequence*-диаграммы можно создавать *Collaboration* и наоборот. Для построения *Collaboration* нужно, находясь в диаграмме *Sequence*, выбрать *Menu: Browse => Create Collaboration diagram*. При построении с помощью значка *Interaction diagram*  и выбора *Collaboration* в диалоговом окне создастся пустая диаграмма, и в нее не перенесутся уже существующие объекты. На рис. 7.2 представлена диаграмма *Collaboration*, построенная на основе диаграммы *Sequence* (см. рис. 7.1).

Для построения приведенной диаграммы *Collaboration* были использованы ее основные инструменты. Значок *Object* позволяет создавать объекты, которые имеют состояния и поведение. Значок  – *Object Link* (связь объекта) отражает взаимодействие объектов посредством показа их связей. При этом один из объектов может посылать сообщение другому. Значок  – *Link To Self* показывает, что объект имеет обратную связь с самим собой. Значок  – *Link Message* (передача сообщения) позволяет отразить связь, которая подразумевает обязательную передачу сообщения. У каждой связи *Link Message* есть соответствующие свойства, которые позволяют настроить область видимости для связанных объектов. Для задания области видимости объектов используется диалоговое окно,

приведенное на рис. 7.3. Сервер описывается в блоке *Supplier visibility*, клиент – в блоке *Client visibility*. В них доступны значения для выбора:

- *Unspecified* – не определено, это значение присваивается по умолчанию.
- *Field* – объект включен в другой объект.

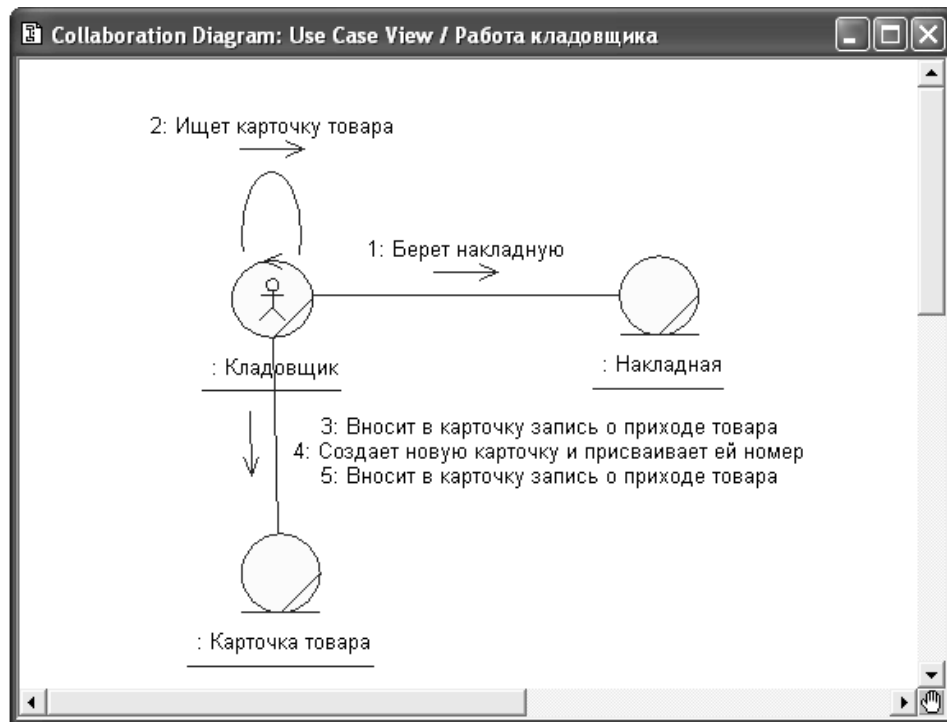


Рисунок 7.2 – Пример работы кладовщика с карточкой товара и накладной

Рисунок 7.3 – Задание области видимости для Link Message

- *Parameter* – объект передается параметром в другой объект.

- *Local* – объект локально определен в границах другого объекта.
- *Global* – объект глобален по отношению к другому объекту.

При изменении области видимости на концах соединяющей линии появляется квадратик с указанной областью видимости.

На рис. 7.2 объект *Кладовщик* выделен как центральный, и все сообщения поступают к нему или исходят от него. Взаимодействия между Кладовщиком и Накладной, а также Карточкой товара изображаются линиями с добавленными

стрелками, аналогичными тем, которые изображаются на Sequence-диаграмме. Все сообщения одного направления собираются вместе и предлагаются как подпись к одной стрелке.

Находясь в области диаграммы *Collaboration*, можно создавать новые объекты и связи между ними. Если затем переключиться в диаграмму *Sequence*, то станет очевидным, что туда автоматически перенеслись все внесенные изменения.

7.2 Построение диаграммы Sequence в Rational XDE

Rational XDE не поддерживает диаграмму *Collaboration* в том виде, в котором она присутствует в среде *Rational Rose*. *Collaboration* нельзя создать как отдельную диаграмму. Проектирование ведется с использованием только диаграммы *Sequence*.

Для создания диаграммы из контекстного меню модели нужно выбрать *Add Diagram => Sequence*. В меню находятся два пункта для создания диаграммы – это *Sequence: Instance* (реализация) и *Sequence: Role* (роль). Эти пункты отражают различные возможности в использовании диаграммы.

В случае построения диаграммы *Sequence* в окне *Model Explorer* появится новый элемент, представляющий собой иерархию группы значков. И хотя *Rational XDE* не поддерживает диаграмму *Collaboration*, отображение взаимодействия происходит через элемент *Collaboration*. Он должен представлять реализацию одного из прецедентов системы и может иметь несколько альтернативных последовательностей действий. Главную из них *Rational XDE* создает сразу, альтернативные можно добавить при необходимости.

Диаграмма *Sequence: Role* отражает роли без ссылки на конкретные классы. Роль – это не класс или объект, а определенная последовательность в обмене сообщениями. Поэтому роли обычно используются для представления образцов. Если в системе задан определенный объект, то каждая роль ссылается на базовый класс, который идентифицируется атрибутами и операциями, нуждающимися во взаимодействии с заданным объектом. В случае использования образцов в системе любой класс, взаимодействующий с ними, должен быть отображен ролью в диаграмме. Замена объектов ролями позволяет использовать один класс для реализации нескольких ролей, которые могут отражаться в нескольких прецедентах. Имя

роли отображается с косой чертой; в имени роли через двоеточие может быть представлено имя класса, реализующего эту роль.

На рис. 7.4 приведен *Toolbox* диаграммы *Sequence*.

Значок *Lifeline Actor* позволяет создать на диаграмме роль актера с указанием линии жизни. Фактически на диаграмме создается отображение актера, а линия жизни дополняет это отображение.

Однако не всегда необходимо создавать новых актеров в диаграмме. В разрабатываемой модели уже созданы актеры *Пользователь*, *Покупатель* и *Администратор*, поэтому при моделировании реализации прецедента *Регистрация* можно воспользоваться уже созданными элементами.

Инструмент *LifeLine* позволяет создать на диаграмме роль для программного объекта с линией жизни. В отличие от *LifeLine Actor* при создании этого элемента класс не добавляется, а его место в названии остается пустым. Абстрактную роль *Система* будет иллюстрировать в модели взаимодействия незарегистрированного пользователя с виртуальным магазином.

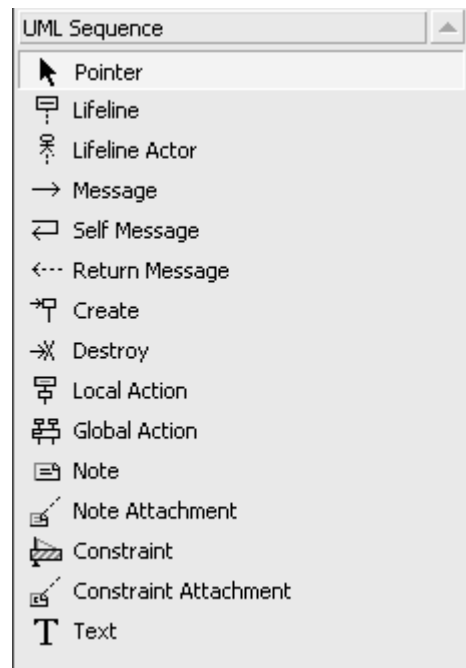


Рисунок 7.4 – Инструменты диаграммы *Sequence*

В диаграмме *Sequence: Instance* линия жизни может явно указать для объекта момент его создания и уничтожения.

Для отражения обмена сообщениями между незарегистрированным пользователем и окном регистрации создаются роли *Незарегистрированный пользователь* и *Окно регистрации*.

Инструмент *Message* позволяет создать отображение сообщения, передаваемого от одного объекта или роли к другому. Передача сообщения означает передачу управления объекту-получателю.

Значок *Return Message* показывает на диаграмме возврат управления из вызванной подпрограммы на сервере клиенту. На диаграмме такое сообщение отправляется от *Пользователя* к *Окну регистрации*.

Значок *Create* предназначен для показа сообщения, при помощи которого один объект создает другой. При этом созданный объект не получает управления, т.е. фокус активности остается у отправителя сообщения.

Значок *Destroy* отражает момент уничтожения программного объекта.

Окончательный вариант диаграммы Sequence показан на рис. 7.5.

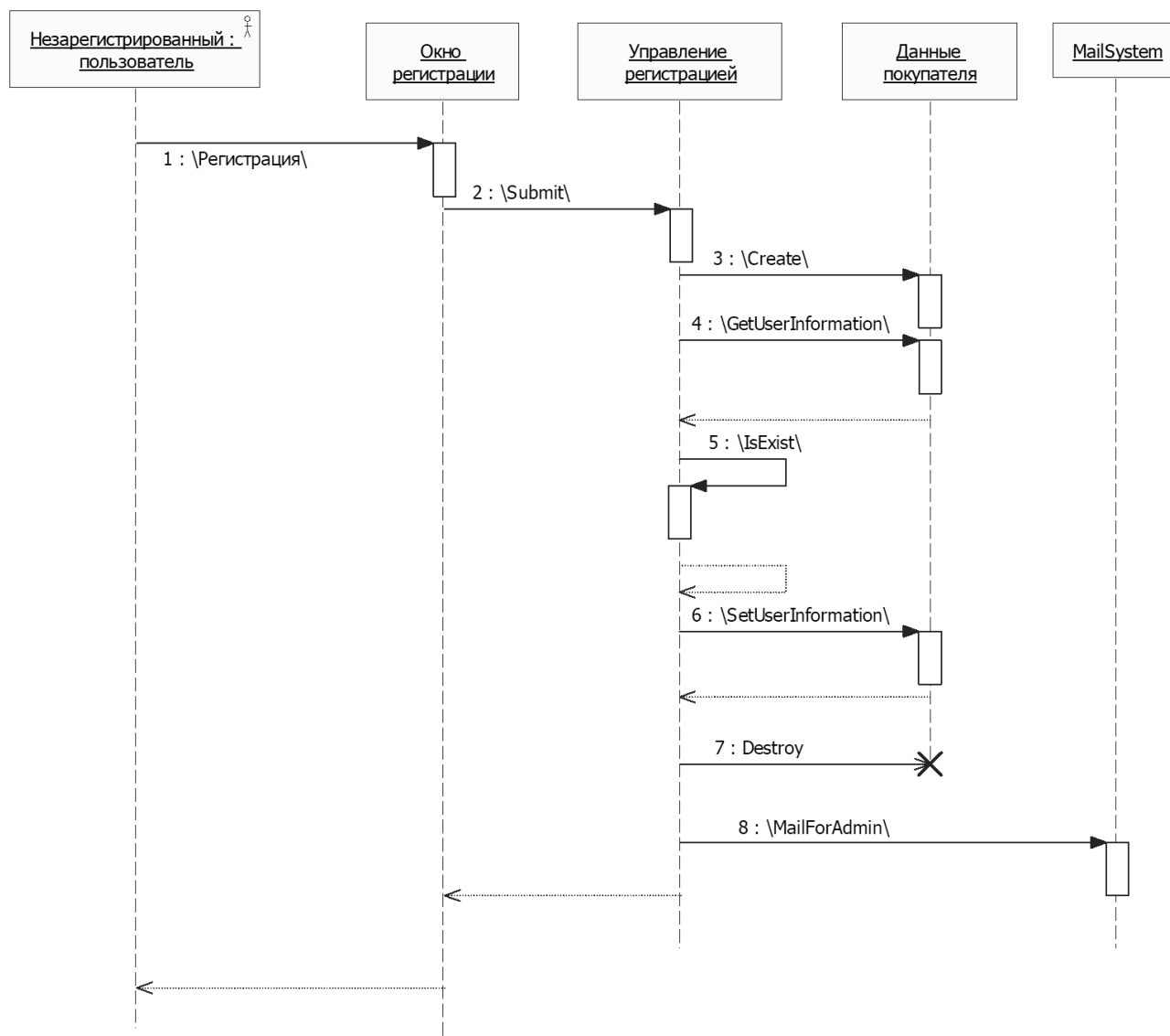


Рисунок 7.5 – Диаграмма Sequence в Rational XDE

Здесь происходят следующие события. *Незарегистрированный пользователь* хочет зарегистрироваться и нажимает кнопку в *Окне регистрации*. Окно регистрации выдает сообщение Submit для объекта управляющего регистрацией, который, в свою очередь, создает объект доступа к данным, выдавая сообщение Create, а затем запрашивает у

созданного объекта данные пользователя для проверки, не был ли зарегистрирован такой пользователь ранее. Это происходит внутри обработчика сообщения *IsExist*, которое объект выдает самому себе. После получения данных и удостоверения того, что записи об этом пользователе еще нет в системе, отправляется сообщение о сохранении данных о пользователе *SetUserInformation*, после чего объект уничтожается.

Сообщение *MailFormAdmin* дает команду на отправку оповещения администратору о том, что в системе зарегистрировался новый пользователь, причем обработка этого сообщения происходит асинхронно, т.е. без ожидания завершения почтовой системы, поскольку отправка почты может занимать довольно длительное время.