Лекция 5. Поведенческие диаграммы.

Рассмотрим поведенческие диаграммы.

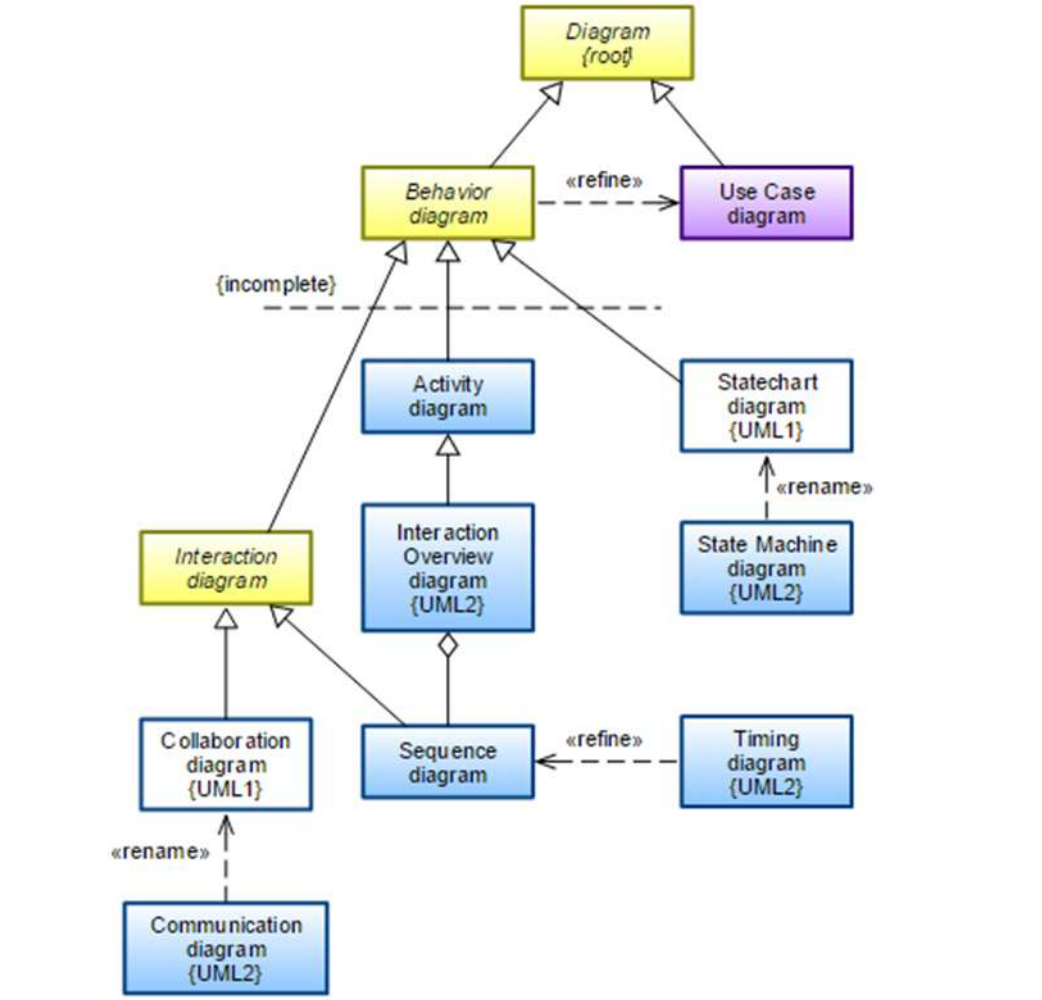

Рис. Иерархия типов диаграмм для UML 2 (часть 2)

Диаграммы вариантов использования (use case diagram)

На диаграммах вариантов использования отображается взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему. Из диаграмм вариантов использования можно получить довольно много информации о системе. Этот тип диаграмм описывает общую функциональность системы. Пользователи, менеджеры проектов, аналитики, разработчики, специалисты по контролю качества и все, кого интересует система в целом, могут, изучая диаграммы вариантов использования, понять, что система должна делать.

2.1 Базовые элементы диаграммы вариантов использования

К базовым элементам рассматриваемой диаграммы относятся *вариант*

использования, *актер* и *интерфейс*.

Вариант использования (рис. 20) применяется для спецификации общих особенностей поведения системы или другой сущности без рассмотрения ее внутренней структуры (например, оформление заказа на покупку товара, получение информации о кредитоспособности клиента, отображение графической формы на экране монитора).

Рис. 20 Графическое обозначение варианта использования

Актер – это внешняя по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для решения определенных задач (рис. 21). При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой.

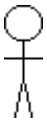

Рис. 21 Графическое обозначение актера

Имя актера должно быть достаточно информативным с точки зрения семантики, например клиент банка, продавец магазина, пассажир авиарейса, водитель автомобиля, сотовый телефон.

Так как в общем случае актер всегда находится вне системы, его внутренняя структура никак не определяется. Для актера имеет значение только его внешнее представление, т.е. то, как он воспринимается со стороны системы. Актеры взаимодействуют с системой посредством передачи и приема сообщений от вариантов использования. Сообщение представляет собой запрос актером сервиса от системы и получение этого сервиса. Это взаимодействие может быть выражено посредством ассоциаций между отдельными актерами и вариантами использования или классами. Кроме этого, с актерами могут быть связаны интерфейсы, которые определяют, каким образом другие элементы модели взаимодействуют с этими актерами.

Интерфейс служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры (рис. 22). В диаграммах вариантов использования интерфейсы определяют совокупность операций, обеспечивающих необходимый набор сервисов или функциональности для актеров. Интерфейсы не могут содержать ни атрибутов, ни состояний, ни направленных ассоциаций. Они содержат только операции без указания особенностей их реализации. Формально интерфейс эквивалентен абстрактному классу без атрибутов и методов с наличием только абстрактных операций.

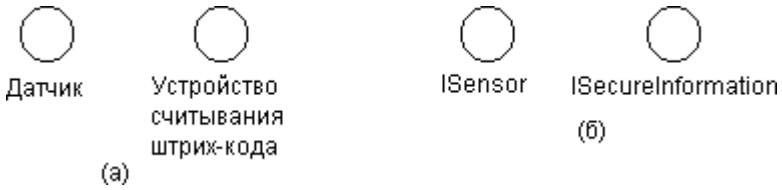

Рис. 22 Графическое изображение интерфейсов на диаграммах вариантов использования

Интерфейс соединяется с вариантом использования сплошной линией, если он реализует все операции, необходимые для данного интерфейса (рис. 23, а). Если же вариант использования определяет только тот сервис, который необходим для реализации данного интерфейса, используется пунктирная стрелка (рис. 23, б).

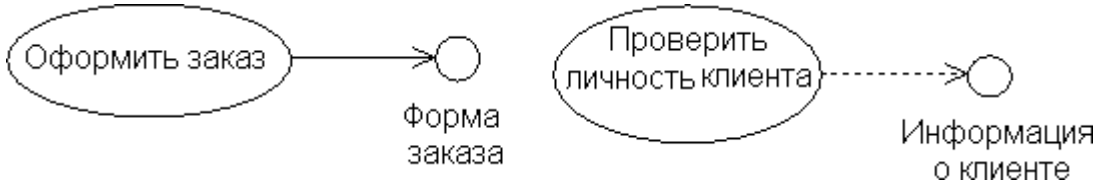

Рис. 23 Графическое изображение взаимосвязей интерфейсов с вариантами использования
а) реализация всех операций, б) реализация только необходимого сервиса

Важность интерфейсов заключается в том, что они определяют стыковочные узлы в проектируемой системе, что совершенно необходимо для организации коллективной работы над проектом. Более того, спецификация интерфейсов способствует “безболезненной” модификации уже существующей системы при переходе на новые технологические решения. В этом случае изменению подвергается только реализация операций, но никак не функциональность самой системы. А это обеспечивает совместимость последующих версий программ с первоначальными при спиральной технологии разработки программных систем.

Примечания в языке UML предназначены для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта (рис. 24). В качестве такой информации могут быть комментарии разработчика (например, дата и версия разработки диаграммы или ее отдельных компонентов), ограничения (например, на значения отдельных связей или экземпляры сущностей) и помеченные значения.

Рис. 24 Примеры примечаний в языке UML

Применительно к диаграммам вариантов использования примечание может носить самую общую информацию, относящуюся к общему контексту системы.

2.2 Отношения на диаграмме вариантов использования

Для выражения отношений между актерами и вариантами использования применяются стандартные виды отношений, описаные в разделе 1.2.

Отношение ассоциации применительно к диаграммам вариантов использования служит для обозначения специфической роли актера в отдельном варианте использования (рис. 25). Другими словами, ассоциация определяет семантические особенности взаимодействия актеров и вариантов использования в графической модели системы. Таким образом, это отношение устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. Графическое обозначение отношения ассоциации может включать дополнительные условные обозначения (имя и кратность).

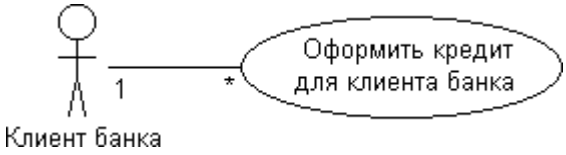

Рис. 25 Отношение ассоциации между актером и вариантом использования

Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования. Данная линия со стрелкой помечается ключевым словом “extend” (“расширяет”), как показано на рис. 26.

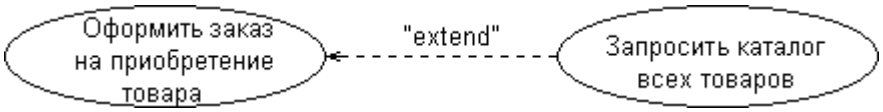

Рис. 26 Отношение расширения между вариантами использования

Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования.

Отношение обобщения графически обозначается сплошной линией со стрелкой, которая указывает на родительский вариант использования (рис. 27).

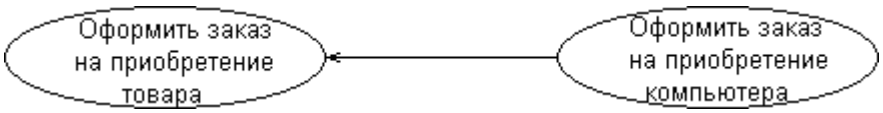

Рис. 27 Отношение обобщения между вариантами использования

Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. При этом дочерние варианты использования участвуют во всех отношениях родительских вариантов. В свою очередь, дочерние варианты могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения.

Отношение включения между двумя вариантами использования указывает, что поведение одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Графически данное отношение обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от базового варианта использования к включаемому. При этом данная линия со стрелкой помечается ключевым словом “include” (“включает”), как показано на рис. 28.

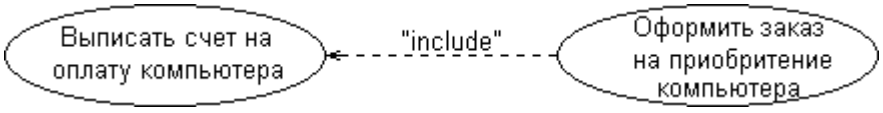

Рис. 28 Отношение включения между вариантами использования

При моделировании возникает необходимость в указании количества объектов, связанных посредством одного экземпляра ассоциации. Это число называется *кратностью* (Multiplicity) роли ассоциации и записывается либо как выражение, значением которого является диапазон значений, либо в явном виде (рис. 29). Кратность указывает на то, столько объектов должно соответствовать каждому объекту на противоположном конце. Кратность можно задать равной единице (1), указать диапазон: “ноль или единица” (0..1), “много” (0..*), “единица или больше” (1..*). Разрешается также указывать определенное число (например, 3).

Рис. 29 Кратность

2.3 Пример диаграммы вариантов использования

Рассмотрим диаграмму вариантов использования отражающую систему работы банковского автомата (Automated Teller Machine, ATM) (рис. 30).

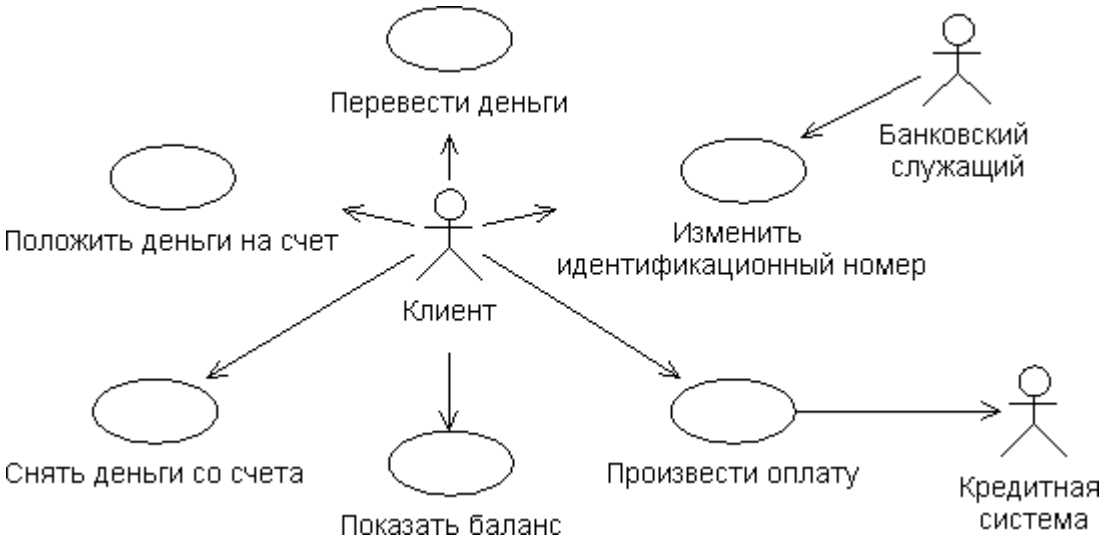

Рис. 30 Диаграмма вариантов использования для ATM

Клиент банка инициирует различные варианты использования: снять деньги со счета, перевести деньги, положить деньги на счет, Показать баланс, изменить идентификационный номер, произвести оплату. Банковский служащий может инициировать вариант использования "Изменить идентификационный номер". Действующими лицами могут быть и внешние системы, в данном случае кредитная система показана именно как действующее лицо – она является внешней для системы ATM. Стрелка, направленная от варианта использования к действующему лицу, показывает, что вариант использования предоставляет некоторую информацию действующему лицу. В данном случае вариант использования "Произвести оплату" предоставляет кредитной системе информацию об оплате по кредитной карточке.

Диаграммы деятельности (activity diagram)

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций.

Для моделирования процесса выполнения операций в языке UML используются так называемые *диаграммы деятельности*. Применяемая в них

графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления не деятельностей, а действий, и в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой, операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

В контексте языка UML деятельность (activity) представляет собой некоторую совокупность отдельных вычислений, выполняемых автоматом. При этом отдельные элементарные вычисления могут приводить к некоторому результату или действию (action). На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Сам же результат может привести к изменению состояния системы или возвращению некоторого значения.

7.1. Основные элементы диаграммы деятельности

Состояние действия (action state) является специальным случаем состояния с некоторым входным действием и по крайней мере одним выходящим из состояния переходом. Графически состояние действия изображается фигурой, напоминающей прямоугольник, боковые стороны которого заменены выпуклыми дугами (рис. 58). Внутри этой фигуры записывается выражение действия (action-expression), которое должно быть уникальным в пределах одной диаграммы деятельности.

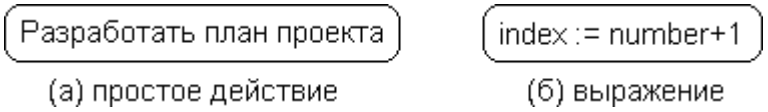

Рис. 58 Графическое изображение состояния действия

Действие может быть записано на естественном языке, некотором псевдокоде или языке программирования. Никаких дополнительных или неявных ограничений при записи действий не накладывается.

При построении диаграммы деятельности используются только те переходы, которые переводят деятельность в последующее состояние сразу, как только закончится действие в предыдущем состоянии (нетриггерные). На диаграмме такой переход изображается сплошной линией со стрелкой.

Ветвление на диаграмме деятельности обозначается небольшим ромбом, внутри которого нет никакого текста (рис. 59).

В качестве примера рассмотрим фрагмент алгоритма нахождения корней квадратного уравнения. В общем случае после приведения уравнения второй степени к каноническому виду: $a*x*x + b*x + c = 0$ необходимо вычислить

его дискриминант. Причем, в случае отрицательного дискриминанта уравнение не имеет решения на множестве действительных чисел, и дальнейшие вычисления должны быть прекращены. При неотрицательном дискриминанте уравнение имеет решение, корни которого могут быть получены на основе конкретной расчетной формулы.

Процедуру вычисления корней квадратного уравнения можно представить в виде диаграммы деятельности с тремя состояниями действия и ветвлением (рис. 59). Каждый из переходов, выходящих из состояния "Вычислить дискриминант", имеет сторожевое условие, определяющее единственную ветвь, по которой может быть продолжен процесс вычисления корней в зависимости от знака дискриминанта.

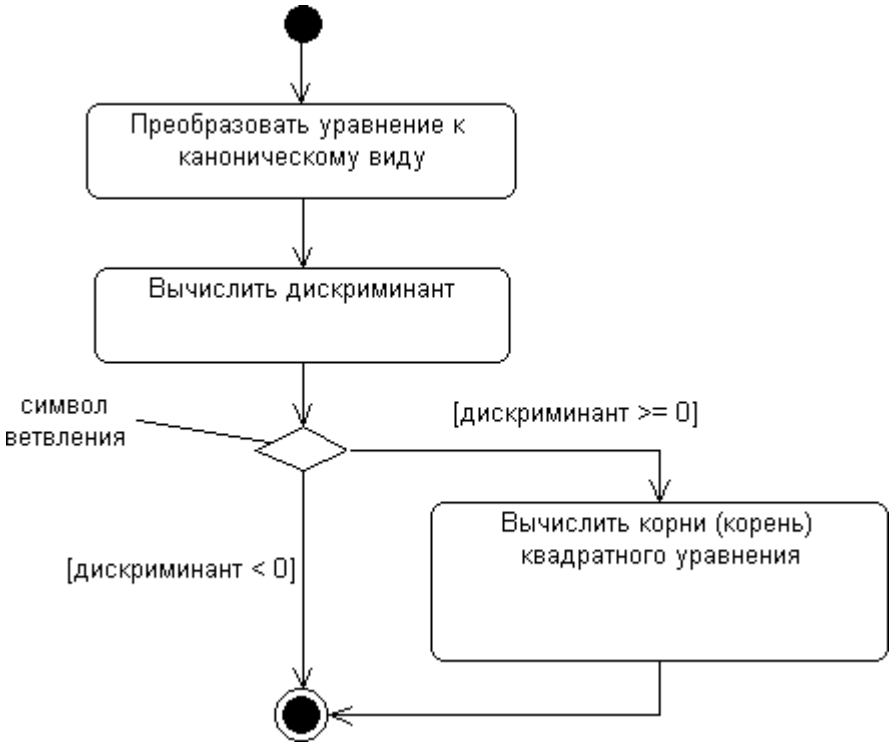

Рис. 59 Фрагмент диаграммы деятельности для алгоритма нахождения корней квадратного уравнения

В языке UML для распараллеливания вычислений используется специальный символ для разделения (рис. 60, а) и слияния (рис. 60, б) параллельных вычислений или потоков управления.

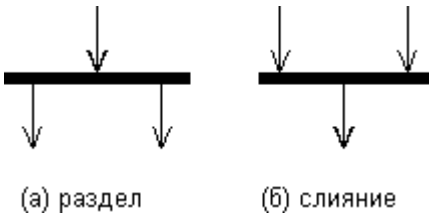

Рис. 60 Разделения и слияния параллельных потоков управления

Диаграммы деятельности могут быть использованы не только для спецификации алгоритмов вычислений или потоков управления в программных системах. Не менее важная область их применения связана с моделированием бизнес-процессов. Для моделирования этих особенностей в

языке UML используется специальная конструкция, получившее название *дорожки* (swimlanes). Имеется в виду визуальная аналогия с плавательными дорожками в бассейне, если смотреть на соответствующую диаграмму. При этом все состояния действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями. Две соседние линии и образуют дорожку, а группа состояний между этими линиями выполняется отдельным подразделением (отделом, группой, отделением, филиалом) компании (рис. 61).

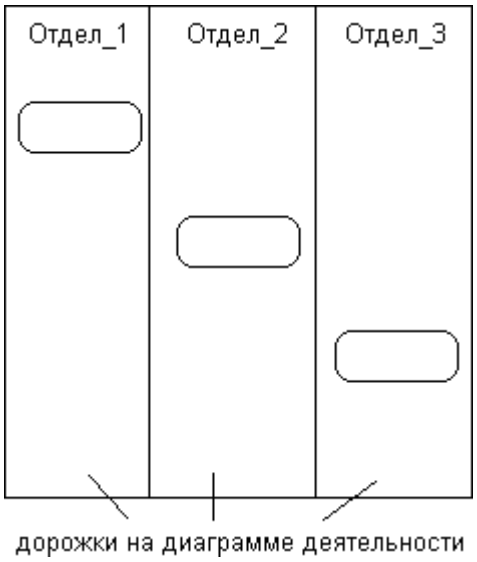

Рис. 61 Вариант диаграммы деятельности с дорожками

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Поскольку в таком ракурсе объекты играют определенную роль в понимании процесса деятельности, иногда возникает необходимость явно указать их на диаграмме деятельности.

Для графического представления объектов используются прямоугольник класса, с тем отличием, что имя объекта подчеркивается. Далее после имени может указываться характеристика состояния объекта в прямых скобках. Такие прямоугольники объектов присоединяются к состояниям действия отношением зависимости пунктирной линией со стрелкой. Соответствующая зависимость определяет состояние конкретного объекта после выполнения предшествующего действия.

7.2 Пример диаграммы деятельности

В качестве примера рассмотрим фрагмент диаграммы деятельности торговой компании, обслуживающей клиентов по телефону. Подразделениями компании являются отдел приема и оформления заказов, отдел продаж и склад.

Этим подразделениям будут соответствовать три дорожки на диаграмме деятельности, каждая из которых специфицирует зону ответственности

подразделения (рис. 62).

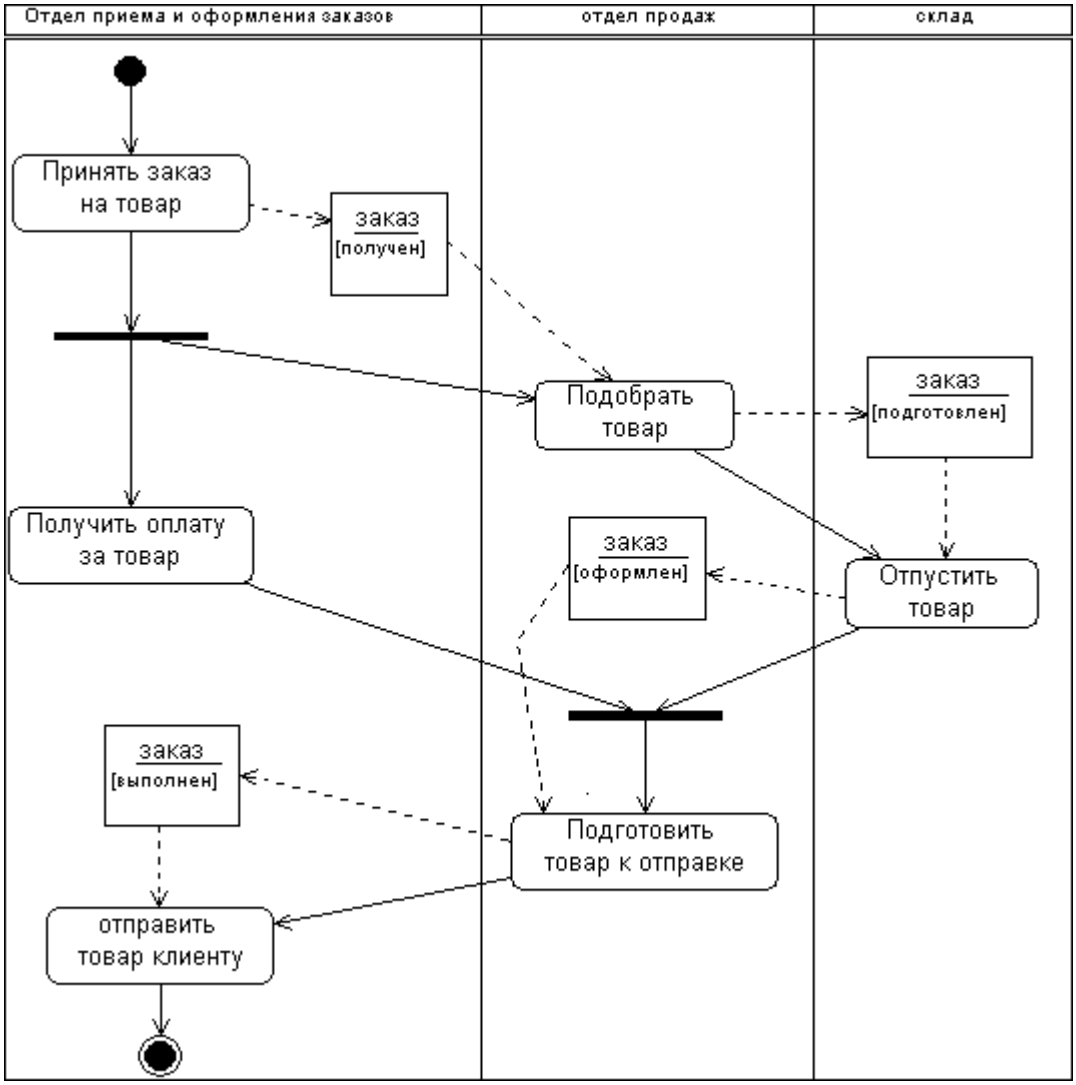

Рис. 62 Диаграмма деятельности торговой компании с объектом-заказом

В данном случае диаграмма деятельности заключает в себе не только информацию о последовательности выполнения рабочих действий, но и о том, какое из подразделений торговой компании должно выполнять то или иное действие. Кроме того центральным объектом процесса продажи является заказ или вернее состояние его выполнения. Вначале до звонка от клиента заказ как объект отсутствует и возникает лишь после такого звонка. Однако этот заказ еще не заполнен до конца, поскольку требуется еще подобрать конкретный товар в отделе продаж. После его подготовки он передается на склад, где вместе с отпуском товара заказ окончательно дооформляется. Наконец, после получения подтверждения об оплате товара эта информация заносится в заказ, и он считается выполненным и закрытым.

Диаграммы состояний (statechart diagram)

Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Чаще всего диаграммы состояний используются для описания поведения

отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Для понимания семантики конкретной диаграммы состояний необходимо представлять не только особенности поведения моделируемой сущности, но и знать общие сведения по теории автоматов.

6.1. Автоматы

Автомат (state machine) в языке UML представляет собой некоторый формализм для моделирования поведения элементов модели и системы в целом. Автомат описывает поведение отдельного объекта в форме последовательности состояний, которые охватывают все этапы его жизненного цикла, начиная от создания объекта и заканчивая его уничтожением. Каждая диаграмма состояний представляет некоторый автомат.

Простейшим примером визуального представления состояний и переходов на основе формализма автоматов может служить ситуация с исправностью технического устройства, такого как компьютер. В этом случае вводятся в рассмотрение два самых общих состояния: "исправен" и "неисправен" и два перехода: "выход из строя" и "ремонт". Графически эта информация может быть представлена в виде изображенной ниже диаграммы состояний компьютера (рис. 53).

Рис. 53 Простейший пример диаграммы состояний для технического устройства типа компьютер

Основными понятиями, входящими в формализм автомата, являются состояние и переход. Главное различие между ними заключается в том, что длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. Предполагается, что в пределе время перехода из одного состояния в другое равно нулю (если дополнительно ничего не сказано). Другими словами, переход объекта из состояния в состояние происходит мгновенно.

В языке UML под *состоянием* понимается абстрактный метакласс, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Состояние может быть задано

в виде набора конкретных значений атрибутов класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта.

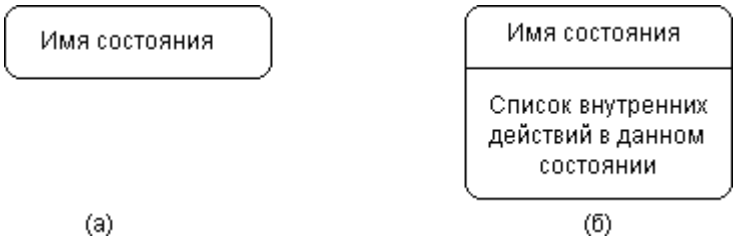

Рис. 54 Графическое изображение состояний на диаграмме состояний

Состояние на диаграмме изображается прямоугольником со скругленными вершинами (рис. 54). Этот прямоугольник, в свою очередь, может быть разделен на две секции горизонтальной линией. Если указана лишь одна секция, то в ней записывается только имя состояния (рис. 54, а). В противном случае в первой из них записывается имя состояния, а во второй – список некоторых внутренних действий или переходов в данном состоянии (рис. 54, б).

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий. В этом состоянии находится объект по умолчанию в начальный момент времени. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка (рис. 55, а), из которого может только выходить стрелка, соответствующая переходу.

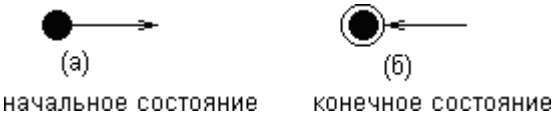

Рис. 55 Графическое изображение начального и конечного состояний на диаграмме состояний

Конечное (финальное) состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий . В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени. Графически конечное состояние в языке UML обозначается в виде закрашенного кружка, помещенного в окружность (рис. 55, б), в которую может только входить стрелка, соответствующая переходу.

Простой переход (simple transition) представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим. Пребывание моделируемого объекта в первом состоянии может сопровождаться выполнением некоторых действий, а переход во второе состояние будет возможен после завершения этих действий, а также после удовлетворения некоторых дополнительных условий. На диаграмме состояний переход изображается сплошной линией со стрелкой, которая направлена в целевое состояние (рис. 56).

Рис. 56 Диаграмма состояний для моделирования почтовой программы-клиента

6.2 Пример диаграммы состояний

Рассмотрим пример диаграммы состояний для моделирования поведения банкомата (рис. 57).

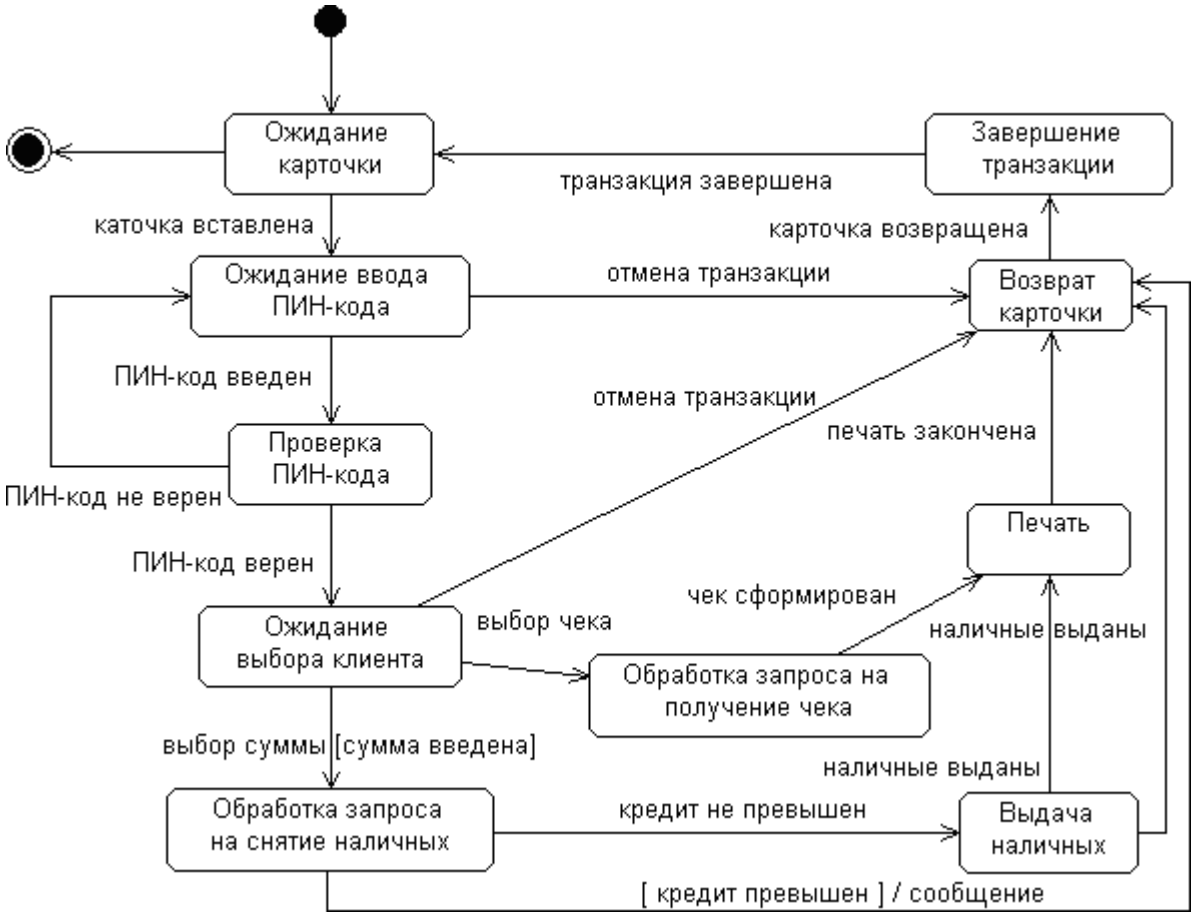

Рис. 57 Диаграмма состояний для моделирования поведения банкомата

Следует заметить, что в разрабатываемой модели диаграмма состояний является единственной и описывает поведение системы управления банкоматом в целом. Главное достоинство данной диаграммы состояний – возможность моделировать условный характер реализации всех вариантов использования в форме изменения отдельных состояний разрабатываемой системы. Иногда разработку диаграммы состояний, особенно в условиях дефицита времени, отпущенного на выполнение проекта, опускают, т.к. часто происходит дублирование информации, представленной на диаграммах

кооперации и последовательности.

Диаграммы последовательности (sequence diagram)

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. При этом учитываются два аспекта: во-первых, взаимодействия объектов можно рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется *диаграмма последовательности*. Во-вторых, можно рассматривать структурные особенности взаимодействия объектов. Для представления структурных особенностей передачи и приема сообщений между объектами используется диаграмма кооперации.

3.1 Объекты диаграммы последовательности

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования. На этих диаграммах изображаются только те объекты, которые непосредственно участвуют во взаимодействии т.к. ключевым моментом является именно динамика взаимодействия объектов во времени и не используются возможные статические ассоциации с другими объектами. При этом диаграмма последовательности имеет два измерения (рис. 31). Одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии. Второе измерение – вертикальная временная ось, направленная сверху вниз. При этом взаимодействия объектов реализуются посредством сообщений, которые посылаются одними объектами другим. Сообщения изображаются в виде горизонтальных стрелок с именем сообщения и также образуют порядок по времени своего возникновения. Другими словами, сообщения, расположенные на диаграмме последовательности выше, инициируются раньше тех, которые расположены ниже.

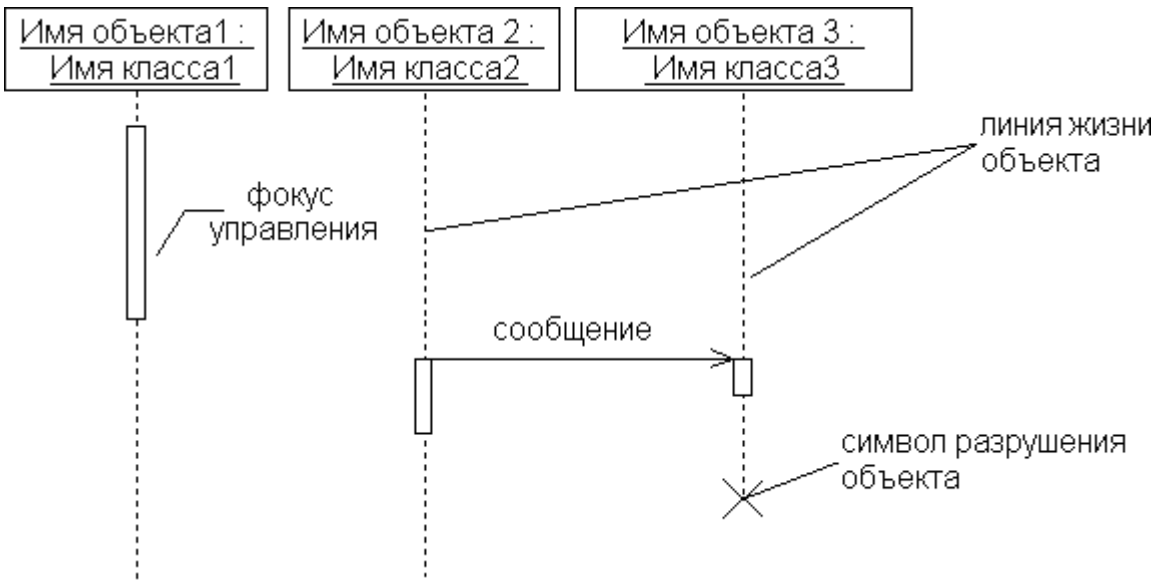

Рис. 31 Графические примитивы диаграммы последовательности

Линия жизни объекта (object lifeline) изображается пунктирной вертикальной линией, ассоциированной с единственным объектом на диаграмме последовательности. Линия жизни служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях. Если объект существует в системе постоянно, то его линия жизни должна начинаться в верхней части диаграммы и заканчиваться в нижней части (объекты 1 и 2 на рис. 31). Отдельные объекты, выполнив свою роль в системе, могут быть уничтожены, чтобы освободить занимаемые ими ресурсы. Для обозначения момента уничтожения объекта в языке UML используется специальный символ в форме латинской буквы “X” (объект 3 на рис. 31). Ниже этого символа пунктирная линия не изображается, поскольку соответствующего объекта в системе уже нет, и этот объект должен быть исключен из всех последующих взаимодействий.

Отдельные объекты в системе могут создаваться по мере необходимости, существенно экономя ресурсы системы и повышая ее производительность (объект 6 на рис. 32).

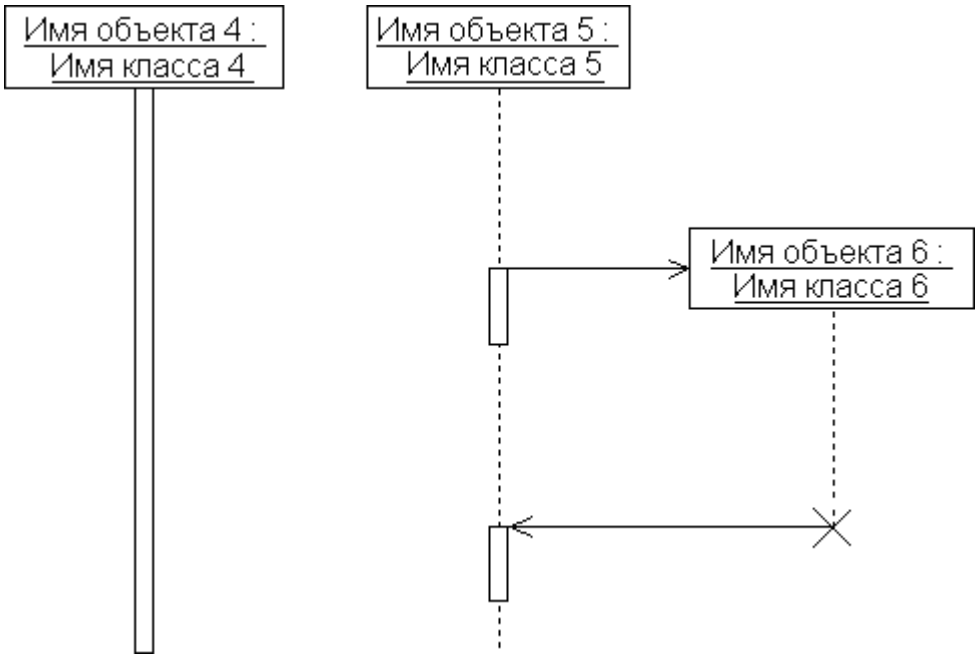

Рис. 32 Варианты линий жизни и фокусов управления объектов

Комментарии или примечания уже рассматривались ранее при изучении других видов диаграмм. Они могут включаться и в диаграммы последовательности, ассоциируясь с отдельными объектами или сообщениями.

Как уже отмечалось выше, взаимодействия объектов реализуются с помощью сообщений. У каждого сообщения должно быть имя, соответствующее его цели. Существует несколько видов сообщений: простое, синхронное, с отказом становиться в очередь и др. (рис. 33).

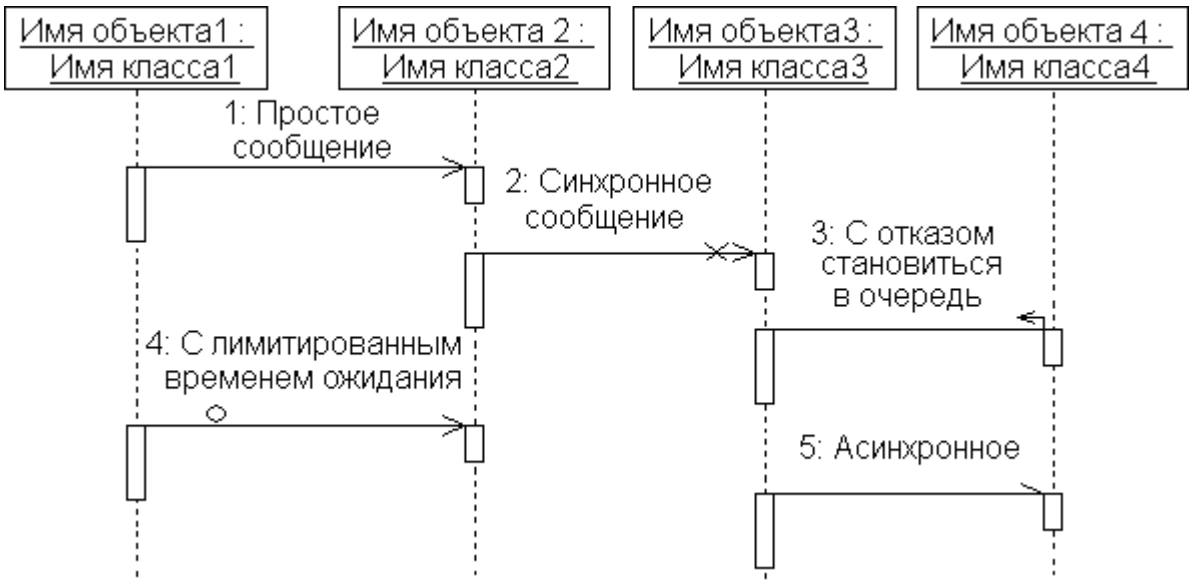

Рис. 33 Примеры сообщений

Простое сообщение используется по умолчанию. Означает, что все сообщения выполняются в одном потоке управления (рис. 33, 1).

Синхронное (synchronous) применяется, когда клиент посылает сообщение и ждет ответа пользователя (рис. 33, 2).

Сообщение с отказом становиться в очередь (balking): клиент посылает сообщение серверу и, если сервер не может немедленно принять сообщение, оно отменяется (рис. 33, 3).

Сообщение с лимитированным временем ожидания (timeout): клиент посылает сообщение серверу, а затем ждет указанное время; если в течение этого времени сервер не принимает сообщение, оно отменяется (рис. 33, 4).

Асинхронное сообщение (asynchronous): клиент посылает сообщение серверу и продолжает свою работу, не ожидая подтверждения о получении (рис. 33, 5).

3.2 Пример диаграммы последовательности

Пример сценария снятия 20$ со счета (при отсутствии таких проблем, как неправильный идентификационный номер или недостаток денег на счету) показан на рис. 34.

Эта диаграмма последовательности отображает поток событий в рамках варианта использования “Снять деньги”. В верхней части диаграммы показаны все действующие лица и объекты, требуемые системе для выполнения варианта использования “Снять деньги”. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. Следует отметить также, что на диаграмме Последовательности показаны именно объекты, а не классы. Классы представляют собой типы объектов. Объекты конкретны; вместо класса Клиент на диаграмме Последовательности представлен конкретный клиент Джо.

Вариант использования начинается, когда клиент вставляет свою

карточку в устройство для чтения – этот объект показан в прямоугольнике в верхней части диаграммы. Он считывает номер карточки, открывает объект “счет” (account) и инициализирует экран АТМ. Экран запрашивает у клиента его регистрационный номер. Клиент вводит число 1234. Экран проверяет номер у объекта “счет” и обнаруживает, что он правильный. Затем экран предоставляет клиенту меню для выбора, и тот выбирает пункт *“Снять деньги”*. Экран запрашивает, сколько он хочет снять, и клиент указывает 20$. Экран снимает деньги со счета. При этом он инициирует серию процессов, выполняемых объектом “счет”. Во-первых, осуществляется проверка, что на этом счету лежат, по крайней мере, 20$. Во-вторых, из счета вычитается требуемая сумма. Затем кассовый аппарат получает инструкцию выдать чек и $20 наличными. Наконец все тот же объект “счет ” дает устройству для чтения карточек инструкцию вернуть карточку.

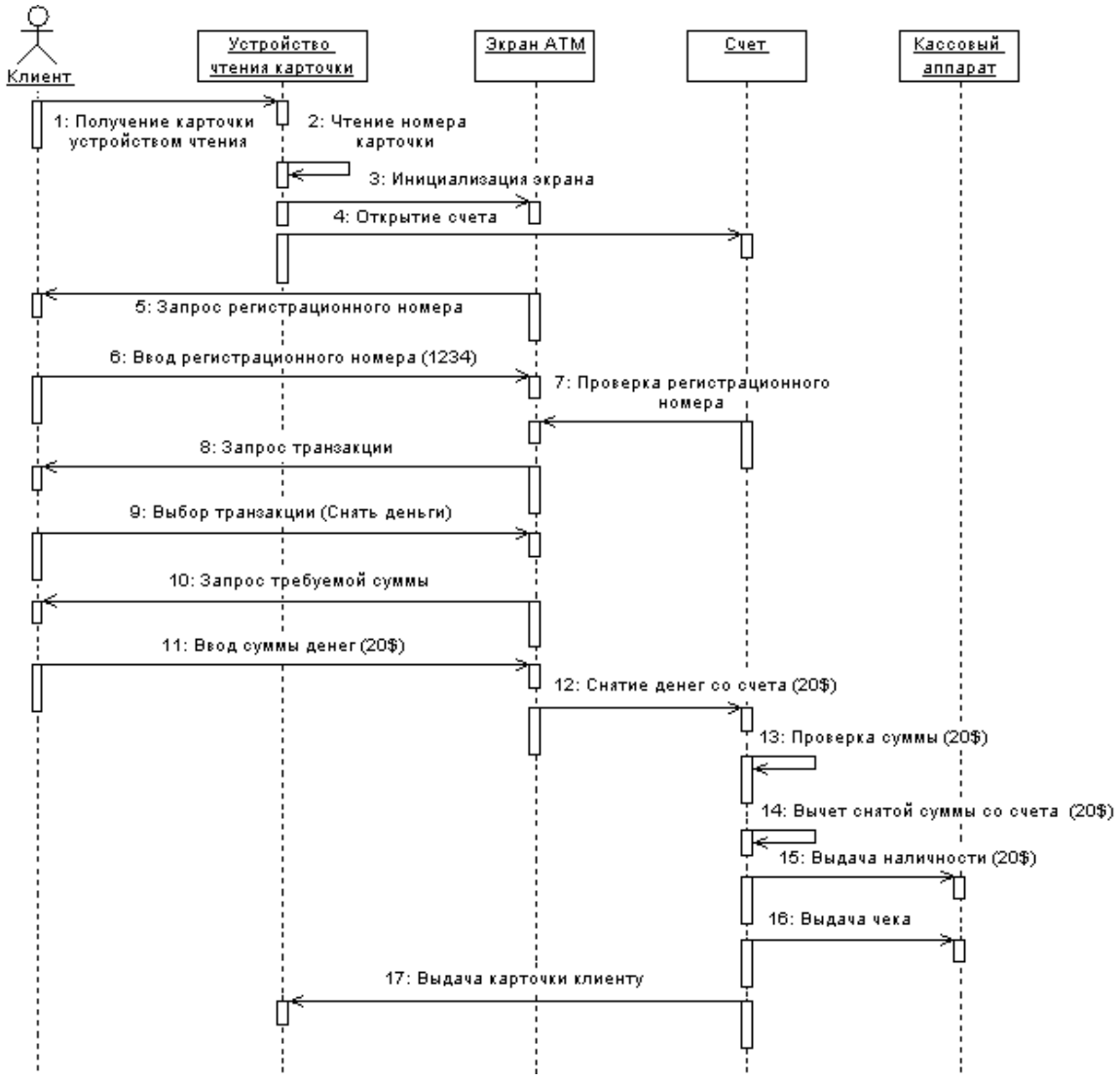

Рис. 34 Диаграмма последовательности для снятия клиентом 20$

Таким образом, диаграмма последовательности иллюстрирует последовательность действий, реализующих вариант использования “Снять деньги со счета” на примере снятия клиентом 20$. Глядя на эту диаграмму, пользователи знакомятся со спецификой своей работы. Аналитики видят

последовательность (поток) действий, разработчики – объекты, которые надо создать, и их операции. Специалисты по контролю качества поймут детали процесса и смогут разработать тесты для их проверки. Таким образом, диаграммы последовательности полезны всем участникам проекта.

Диаграммы кооперации (collaboration diagram)

Подобно диаграммам последовательности, *диаграммы кооперации* отображают поток событий в конкретном сценарии варианта использования. Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Прежде всего, на диаграмме кооперации в виде прямоугольников изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее, как и на диаграмме классов, указываются ассоциации между объектами в виде различных соединительных линий. При этом можно явно указать имена ассоциации и ролей, которые играют объекты в данной ассоциации. Дополнительно могут быть изображены динамические связи – потоки сообщений. Они представляются также в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии, а последовательность взаимодействий и параллельных потоков определяется с помощью порядковых номеров.

4.1 Объекты диаграммы кооперации

Отдельные аспекты спецификации объектов как элементов диаграмм уже рассматривались ранее при описании диаграмм последовательности. Эти же объекты являются основными элементами из которых строится диаграмма кооперации. Для графического изображения объектов используется такой же символ прямоугольника, что и для классов.

Объект является отдельным экземпляром класса, который создается на этапе выполнения программы. Он может иметь свое собственное имя и конкретные значения атрибутов. Для обозначения роли классификатора небходимо указать либо имя класса (вместе с двоеточием), либо имя роли (вместе с наклонной чертой). В противном случае прямоугольник будет соответствовать обычному классу. Если роль, которую должен играть объект, наследуется от нескольких классов, то все они должны быть указаны явно и разделяться запятой и двоеточием.

Отдельные примеры изображения объектов и классов на диаграмме

кооперации приводятся на рис. 35. В первом случае (рис. 35, а) обозначен объект с именем “клиент”, играющий роль “инициатор запроса”. Далее (рис. 35, б) следует обозначение анонимного объекта, который играет роль инициатора запроса. В обоих случаях не указан класс, на основе которого будут созданы эти объекты. Обозначение класса присутствует в следующем варианте записи (рис. 35, в), причем объект также анонимный.

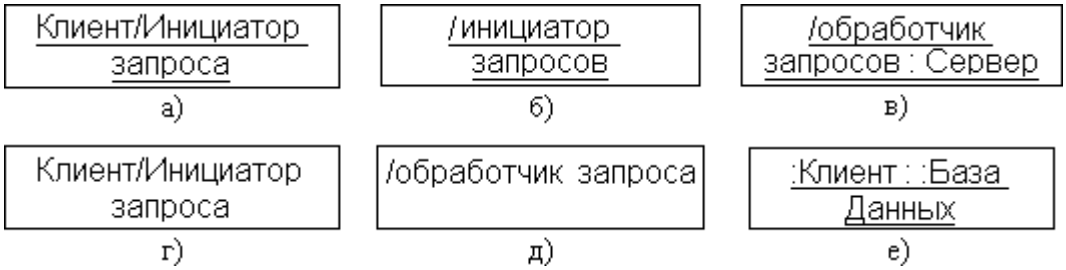

Рис. 35 Варианты записи имен объектов, ролей и классов на диаграммах кооперации

Применительно к уровню спецификации на диаграммах кооперации могут присутствовать именованные классы с указанием роли класса в кооперации (рис. 35, г) или анонимные классы, когда указывается только его роль (рис. 35, д). Последний случай характерен для ситуации, когда в модели могут присутствовать несколько классов с именем “Клиент”, поэтому требуется явно указать имя соответствующего пакета База данных (рис. 35, е).

Мультиобъект (multiobject) представляет собой целое множество объектов на одном из концов ассоциации (рис. 36, а). На диаграмме кооперации мультиобъект используется чтобы показать операции и сигналы, адресованные всему множеству объектов, а не только одному. При этом стрелка сообщения относится ко всему множеству объектов, которые обозначают данный мультиобъект. На диаграмме кооперации может быть явно указано отношение композиции между мультиобъектом и отдельным объектом из его множества (рис. 36, б).

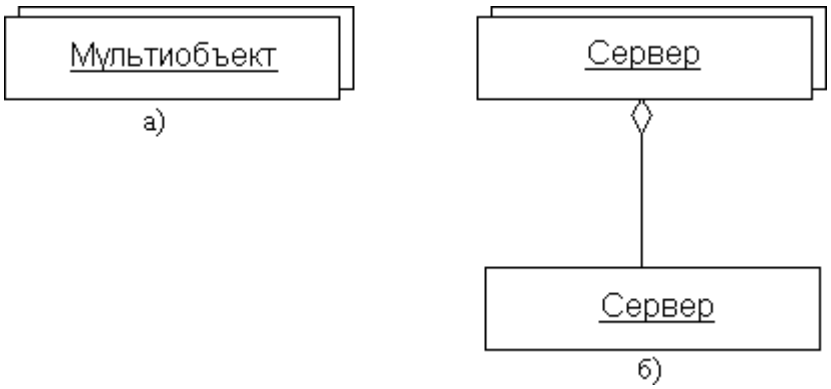

Рис. 36 Графическое изображение мультиобъектов на диаграмме кооперации

В контексте языка UML все объекты делятся на две категории: *пассивные* и *активные*. Пассивный объект оперирует только данными и не может инициировать деятельность по управлению другими объектами. В

тоже время пассивные объекты могут посылать сигналы в процессе выполнения запросов, которые они получают.

Активный объект имеет свою собственную нить управления и может инициировать деятельность по управлению другими объектами. При этом под нитью понимается поток управления, который может выполняться параллельно с другими вычислительными нитями или нитями управления в пределах одного вычислительного процесса.

В приведенном фрагменте диаграммы кооперации (рис. 37) активный объект “a: Вызывающий абонент” является инициатором процесса установления соединения для обмена информацией с другим абонентом.

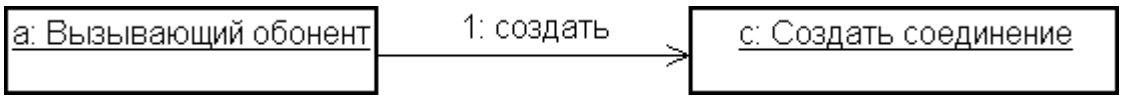

Рис. 37 Активный объект (слева) на диаграмме кооперации

Составной объект (composite object) или объект-контейнер предназначен для представления объекта, имеющего собственную структуру и внутренние потоки (нити) управления. Составной объект является экземпляром составного класса (класса-контейнера), который связан отношением агрегации или композиции со своими частями. Аналогичные отношения связывают между собой и соответствующие объекты.

На диаграммах кооперации составной объект состоит из двух секций: верхней и нижней. В верхней секции записывается имя составного объекта, а в нижней – его элементы (рис. 38), которые могут быть составными объектами.

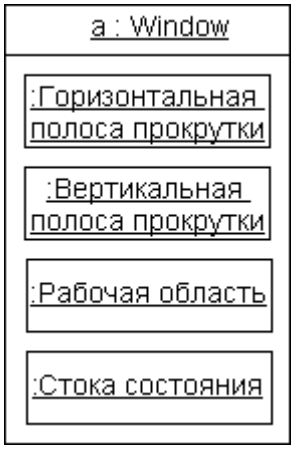

Рис. 38 Составной объект на диаграмме кооперации

Связь (link) является экземпляром или примером произвольной ассоциации. Связь как элемент языка UML может иметь место между двумя и более объектами. Связь на диаграмме кооперации изображается отрезком прямой линии, соединяющей два прямоугольника объектов. На каждом из концов этой линии могут быть явно указаны имена ролей данной ассоциации. Рядом с линией в ее средней части может записываться имя соответствующей ассоциации. Связи не имеют собственных имен, поскольку полностью идентичны как экземпляры ассоциации. Для связей не

указывается также и кратность.

Применительно к диаграммам кооперации сообщения имеют некоторые дополнительные семантические особенности. Они определяют коммуникацию между двумя объектами, один из которых передает другому некоторую информацию. При этом первый объект ожидает, что после получения сообщения вторым объектом последует выполнение некоторого действия. Таким образом, именно сообщение является причиной или стимулом для начала выполнения операций, отправки сигналов, создания и уничтожения отдельных объектов. Связь обеспечивает канал для направленной передачи сообщений между объектами от объекта-источника к объекту-получателю.

4.2 Пример диаграммы кооперации

На рис. 39 приведена кооперативная диаграмма, описывающая, как клиент снимает со счёта 20$.

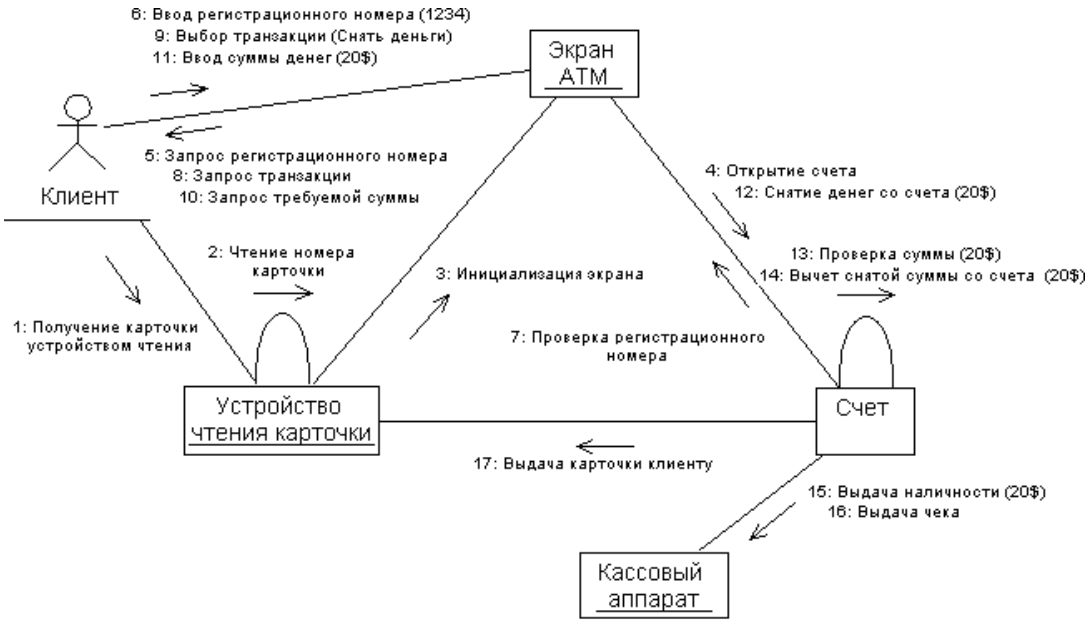

Рис. 39 Диаграмма кооперации для снятия клиентом 20$

Из Кооперативной диаграммы легче понять поток событий и отношения между объектами, однако труднее уяснить последовательность событий, поэтому для сценария создают диаграммы обоих типов.