Лекция 3 часть 5:

Элементы графической нотации диаграммы состояний

Аннотация: Особенности моделирования поведения объектов в виде диаграмм состояний. Понятие конечного автомата и логика изменения его состояний. Описание реакции объекта на асинхронные внешние события в форме диаграммы состояния. Внутренние действия состояния и ду-деятельность. Триггерные и нетриггерные переходы. События и их спецификация на диаграммах состояний.

Ключевые слова: диаграмма состояний, семантика, состояние, контекст, диаграммы вариантов использования, переход, цикла,событие, конечный автомат, UML, моделирование, атрибут, значение, объект, деятельность, прямоугольник, очередь, с писок,атомарная операция, истина, ложь, исключение, метка, выражение, входное действие, действие на выходе, выход, меню,диаграмма, жизненный цикл, Окружность, подсостояние, activity, сторожевое условие, место, операции, псевдокод, язык программирования, Дополнение, вычисление, конфликт, отмена транзакции, целый, синтаксис, запись, отображение, остаток,кредит, формализм

Диаграмма состояний в контексте конечного автомата

Для систем различной природы и назначения характерно взаимодействие между собой отдельных образующих их элементов. Для представления динамических особенностей взаимодействия элементов модели, в контексте реализации вариантов использования, предназначены диаграммы кооперации и последовательности. Однако для моделирования процессов функционирования большинства сложных систем, особенно систем реального времени, этих представлений недостаточно.

Диаграмма состояний (state chart diagram) - диаграмма, которая представляет конечный автомат.

Семантика понятия *состояния* довольно сложна. Дело в том, что характеристика *состояний* системы явным образом не зависит от логической структуры, зафиксированной на диаграмме классов. При рассмотрении *состояний* системы приходится отвлекаться от особенностей ее объектной структуры и мыслить категориями, описывающими динамический *контекст* поведения моделируемой системы. При построении *диаграмм состояний* необходимо использовать специальные понятия, о которых и пойдет речь в данной лекции.

Ранее отмечалось, что любая прикладная система характеризуется не только структурой составляющих ее элементов, но и некоторым поведением или функциональностью. Для общего представления функциональности моделируемой системы предназначены *диаграммы вариантов использования*, которые на концептуальном уровне описывают поведение системы в целом. Для того чтобы представить наиболее общее поведение на логическом уровне, следует ответить на вопрос: "В процессе какого поведения система реализует необходимую пользователям функциональность?".

Главное назначение *диаграммы состояний* - описать возможные последовательности *состояний* и *переходов*, которые в совокупности характеризуют поведение моделируемой системы в течение всего ее жизненного *цикла*. *Диаграмма состояний* представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие

некоторых конкретных *событий*. Системы, которые реагируют на внешние действия от других систем или от пользователей, иногда называют реактивными. Если такие действия инициируются в произвольные случайные моменты времени, то говорят об асинхронном поведении модели.

Диаграммы состояний чаще всего используются для описания поведения отдельных систем и подсистем. Они также могут быть применены для спецификации функциональности экземпляров отдельных классов, т.е. для моделирования всех возможных изменений *состояний* конкретных объектов. *Диаграмма состояний* по существу является графом специального вида, который служит для представления *конечного автомата*.

Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы для более детального представления *состояний* отдельных элементов модели. Для понимания семантики конкретной *диаграммы состояний* необходимо представлять особенности поведения моделируемой сущности, а также иметь общие сведения из теории *конечных автоматов*.

Конечный автомат (state machine) - модель для спецификации поведения объекта в форме последовательности его состояний, которые описывают реакцию объекта на внешние события, выполнение объектом действий, а также изменение его отдельных свойств.

В контексте языка *UML* понятие *конечного автомата* обладает дополнительной семантикой. Вершинами графа *конечного автомата* являются *состояния* и другие типы элементов модели, которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения *переходов* из *состояния* в *состояние*. *Конечный автомат* описывает поведение отдельного объекта в форме последовательности *состояний*, охватывающих все этапы его жизненного *цикла*, начиная от создания объекта и заканчивая его уничтожением.
Каждая *диаграмма состояний* представляет собой *конечный автомат*.

Основными понятиями, характеризующими *конечный автомат*, являются *состояние* и *переход*. Ключевое 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 общем случае *конечный автомат* представляет динамические аспекты моделируемой системы в виде ориентированного графа, вершины которого соответствуют *состояниям*, а дуги - *переходам*. При этом поведение моделируется как последовательное перемещение по графу *состояний* от вершины к вершине по связывающим их дугам с учетом их ориентации. Для графа *состояний* системы можно ввести в рассмотрение специальные свойства.

Среди таких свойств - выделение из всей совокупности *состояний* двух специальных: начального и конечного. Ни в графе *состояний*, ни на *диаграмме состояний* время нахождения системы в том или ином *состоянии* явно не учитывается, однако предполагается, что последовательность изменения *состояний* упорядочена во времени. Другими словами, каждое последующее *состояние* может наступить позже предшествующего ему *состояния*.

Состояние и его графическое изображение

Моделирование поведения объектов и системы в целом основывается на понятии *состояния*.

Состояние (state) - **условие или ситуация в ходе жизненного цикла объекта, в течение которого он удовлетворяет логическому условию, выполняет определенную деятельность или ожидает события**.

Состояние может быть задано в виде набора конкретных значений атрибутов объекта некоторого класса, при этом изменение отдельных значений этих атрибутов будет отражать изменение *состояния* моделируемого объекта или системы в целом. Однако не каждый *атрибут* класса может характеризовать *состояние* его объектов. Как правило, имеют *значение* только те свойства элементов системы, которые отражают динамический или функциональный аспект ее поведения. В этом случае *состояние* будет характеризоваться некоторым инвариантным условием, включающим в себя только принципиальные для поведения объекта или системы атрибуты классов и их значения.

Такое условие может соответствовать ситуации, когда моделируемый *объект* находится в *состоянии* ожидания возникновения внешнего *события*. В то же время нахождение объекта в некотором *состоянии* может быть связано с выполнением определенных действий. В последнем случае соответствующая *деятельность* начинается в момент *перехода* моделируемого элемента в рассматриваемое *состояние*, а после и элемент может покинуть данное *состояние* в момент завершения этой деятельности.

Состояние на диаграмме изображается прямоугольником со скругленными вершинами (рис. 3-5.1). 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. 3-5.1, а). В противном случае в первой из них записывается имя *состояния*, а во второй – *список* некоторых внутренних действий или *переходов* в данном *состоянии* (рис. 3-5.1, б). При этом под действием в языке *UML* понимают некоторую *атомарную операцию*, выполнение которой приводит к изменению *состояния* или возврату некоторого значения (например, "*истина*" или "*ложь*").

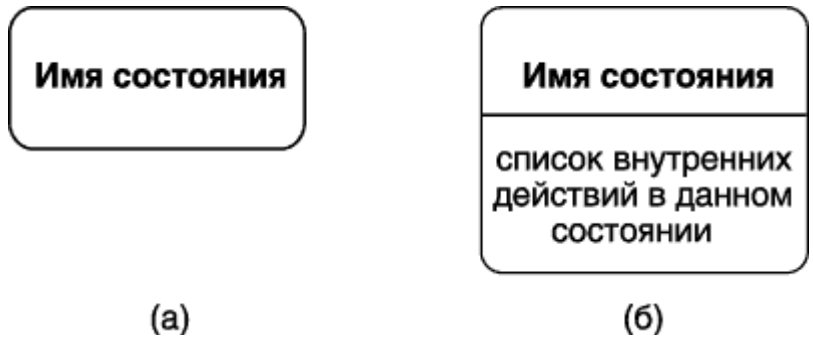

Рис. 3-5.1. Графическое изображение состояний на диаграмме состояний

Имя *состояния* представляет собой строку текста, которая раскрывает содержательный смысл или семантику данного *состояния*. Имя должно представлять собой законченное предложение и всегда записываться с заглавной буквы. Поскольку *состояние* системы является частью процесса ее функционирования, рекомендуется в качестве имени использовать глаголы в настоящем времени или соответствующие причастия. Как *исключение*, имя у *состояния* может отсутствовать, т.е. оно необязательно для некоторых *состояний*. В этом случае *состояние* является анонимным. Если на одной *диаграмме состояний* несколько анонимных *состояний*, то все они должны различаться между собой.

Действие (action) - спецификация выполнимого утверждения, которая образует абстракцию вычислительной процедуры.

Действие обычно приводит к изменению *состояния* системы, и может быть реализовано посредством передачи сообщения объекту, модификацией связи или значения атрибута. Для ряда *состояний* может потребоваться дополнительно указать действия, которые должны быть выполнены моделируемым элементом. Для этой цели служит добавочная секция в обозначении *состояния*, содержащая перечень внутренних действий или *деятельность*, которые

производятся в процессе нахождения моделируемого элемента в данном *состоянии*. Каждое действие записывается в виде отдельной строки и имеет следующий формат:

<метка действия '/ ' выражение действия>

Метка действия указывает на обстоятельства или условия, при которых будет выполняться *деятельность*, определенная выражением действия. При этом *выражение* действия может использовать любые атрибуты и связи, принадлежащие области имен или контексту моделируемого объекта. Если *список* выражений действия пустой, то *метка* действия с разделителем в виде наклонной черты '/' не указывается. Перечень меток действий в языке *UML* фиксирован, причем эти метки не могут быть использованы в качестве имен *событий*:

Входное действие (entry action) - действие, которое выполняется в момент перехода в данное состояние.

Обозначается с помощью ключевого слова - метки действия entry, которое указывает на то, что следующее за ней *выражение*действия должно быть выполнено в момент входа в данное *состояние*.

Действие выхода (exit action) - действие, производимое при выходе из данного состояния.

Обозначается с помощью ключевого слова - метки действия exit, которое указывает на то, что следующее за ней *выражение* действия должно быть выполнено в момент выхода из данного *состояния*.

Внутренняя деятельность (do activity) - выполнение объектом операций или процедур, которые требуют определенного времени.

Обозначается с помощью ключевого слова - метки деятельности do, которое специфицирует так называемую "ду-*деятельность*", выполняемую в течение всего времени, пока *объект* находится в данном *состоянии*, или до тех пор, пока не будет прервано внешним *событием*. При нормальном завершении внутренней деятельности генерируется соответствующее *событие*.

Во всех остальных случаях *метка* действия идентифицирует *событие*, которое запускает соответствующее *выражение* действия. Эти *события* называются внутренними *переходами*. Семантически они эквивалентны *переходам* в само это *состояние*, за исключением той особенности, что *выход* из этого *состояния* или повторный вход в него не происходит. Это означает, что действия входа и выхода не производятся. При этом выполнение внутренних действий в *состоянии* не может быть прервано никакими внешними *событиями*, в отличие от внутренней деятельности, выполнение которой требует определенного времени.

В качестве примера *состояния* можно рассмотреть аутентификацию клиента для доступа к ресурсам моделируемой информационной системы (рис. 3-5.2). *Список* внутренних действий в данном *состоянии* может включать следующие действия. Первое действие - *входное*, которое выполняется при входе в это *состояние* и связано с получением строки символов, соответствующих паролю клиента. Далее выполняется *деятельность* по проверке введенного клиентом пароля. При успешном завершении этой проверки выполняется *действие на выходе*, которое отображает *меню* доступных для клиента опций.

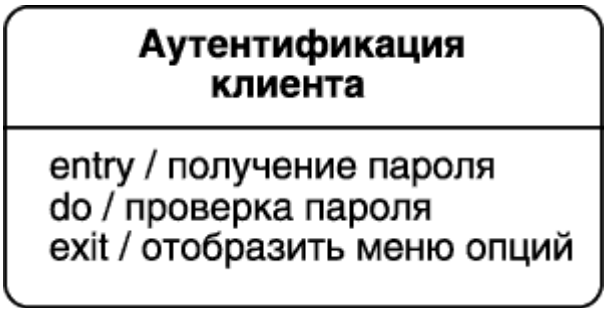

Рис. 3-5.2. Пример состояния с непустой секцией внутренних действий

Кроме обычных *состояний* на *диаграмме состояний* могут размещаться псевдосостояния.

Псевдосостояние (pseudo-state) - вершина в конечном автомате, которая имеет форму состояния, но не обладает поведением.

Примерами псевдосостояний, которые определены в языке *UML*, являются начальное и конечное *состояния*.

Начальное состояние (start state) - разновидность псевдосостояния, обозначающее начало выполнения процесса изменения состояний конечного автомата или нахождения моделируемого объекта в составном состоянии.

В этом *состоянии* находится *объект* по умолчанию в начальный момент времени. Оно служит для указания на *диаграмме состояний* графической области, от которой начинается процесс изменения *состояний*. Графически начальное *состояние* в языке*UML* обозначается в виде закрашенного кружка (рис. 3-5.3, а), из которого может только выходить стрелка-переход.

Рис. 3-5.3. Графическое изображение начального и конечного состояний на диаграмме состояний

На самом верхнем уровне представления объекта *переход* из начального *состояния* может быть помечен *событием* создания (инициализации) данного объекта. В противном случае этот *переход* никак не помечается. Если этот *переход* не помечен, то он является первым *переходом* на *диаграмме состояний* в следующее за ним *состояние*. Каждая *диаграмма* или под-*диаграмма состояний* должна иметь единственное начальное *состояние*.

Конечное состояние (final state) - разновидность псевдосостояния, обозначающее прекращение процесса изменения состояний конечного автомата или нахождения моделируемого объекта в составном состоянии.

В этом *состоянии* должен находиться моделируемый *объект* или система по умолчанию после завершения работы *конечного автомата*. Оно служит для указания на *диаграмме состояний* графической области, в которой завершается процесс изменения *состояний* или *жизненный цикл* данного объекта. Графически конечное *состояние* в языке *UML* обозначается в виде закрашенного кружка, помещенного в *окружность* (рис. 3-5.3, б), в которую может только входить стрелка-переход. Каждая *диаграмма состояний* или *подсостояний* может иметь несколько конечных *состояний*, при этом все они считаются эквивалентными на одном уровне вложенности *состояний*.

Переход и событие

Пребывание моделируемого объекта или системы в первом *состоянии* может сопровождаться выполнением некоторых внутренних действий или деятельности. При этом изменение текущего *состояния* объекта будет возможно либо после завершения этих действий (деятельности), либо при возникновении некоторых внешних *событий*. В обоих случаях говорят, что происходит *переход* объекта из одного *состояния* в другое.

Переход (transition) - отношение между двумя состояниями, которое указывает на то, что объект в первом состоянии должен выполнить определенные действия и перейти во второе состояние.

Переход осуществляется при наступлении некоторого *события*: окончания выполнения деятельности (do *activity*), получении объектом сообщения или приемом сигнала.
На *переходе* указывается имя *события*, а также действия, производимые объектом в ответ на внешние *события* при *переходе* из одного *состояния* в другое.

Переход может быть направлен в то же *состояние*, из которого он выходит. В этом случае его называют *переходом* в себя. Исходное и целевое *состояния перехода* в себя совпадают.
Этот *переход* изображается петлей со стрелкой и отличается от внутреннего *перехода*.
При *переходе* в себя *объект* покидает исходное *состояние*, а затем снова входит в него. При этом всякий раз выполняются внутренние действия, специфицированные метками entry и exit.

Срабатывание <перехода> (fire) - выполнение перехода из одного состояния в другое.

Срабатывание *перехода* может зависеть не только от наступления *события*, но и от выполнения определенного условия, называемого *сторожевым*. *Объект* перейдет из одного *состояния* в другое в том случае, если произошло указанное *событие* и *сторожевое условие* приняло *значение* "*истина*". До срабатывания *перехода объект* находится в предыдущем от него *состоянии*, называемым исходным, или в источнике (не путать с начальным *состоянием* - это разные понятия), а после его срабатывания *объект* находится в последующем от него *состоянии* (целевом *состоянии*).

На *диаграмме состояний переход* изображается сплошной линией со стрелкой, которая выходит из исходного *состояния* и направлена в целевое *состояние* (например, *выход* из строя на рис. 3-5.1). Каждый *переход* может быть помечен строкой текста, которая имеет следующий общий формат:

<имя события>'('<список параметров,
разделенных запятыми>')'
'['<сторожевое условие>']'
<выражение действия>.

Событие (event) - спецификация существенных явлений в поведении системы, которые имеют местоположение во времени и пространстве.

Формально, *событие* представляет собой спецификацию факта, имеющего *место* в пространстве и во времени. Про *события* говорят, что они "происходят", при этом отдельные *события* должны быть упорядочены во времени. После наступления *события* нельзя уже вернуться к предыдущим, если такая возможность явно не предусмотрена в модели.

Семантика понятия *события* фиксирует внимание на внешних проявлениях качественных изменений, происходящих при *переходе* моделируемого объекта из *состояния* в *состояние*. Например, при включении электрического переключателя происходит *событие*, в результате которого комната освещается. После успешного ремонта компьютера также происходит

немаловажное *событие* - восстановление его работоспособности. Если поднять трубку обычного телефона, то, в случае его исправности, мы ожидаем услышать тоновый сигнал. Это тоже является *событием*.

В языке *UML события* играют роль стимулов, которые инициируют *переходы* из одних *состояний* в другие. В качестве *событий* можно рассматривать сигналы, вызовы, окончание фиксированных промежутков времени или моменты окончания выполнения определенных действий. В зависимости от вида происходящих *событий* - стимулов в языке *UML* различают два типа *переходов*: триггерные и нетриггерные.

Переход называется триггерным, если его специфицирует событие-триггер, связанное с внешними условиями по отношению к рассматриваемому состоянию.

В этом случае рядом со стрелкой триггерного *перехода* обязательно указывается имя *события* в форме строки текста, начинающейся со строчной буквы. Наиболее часто в качестве имен триггерных *переходов* задают имена операций, вызываемых у тех или иных объектов системы. После имени такого *события* следуют круглые скобки для явного задания параметров соответствующей *операции*. Если таких параметров нет, то *список* параметров со скобками может отсутствовать. Например, переход на рис. 3-5.4, а, является триггерным, поскольку с ним связано конкретное событие-триггер, происходящее асинхронно при срабатывании некоторого датчика.

Переход называется нетриггерным, если он происходит по завершении выполнения ду-деятельности в данном состоянии.

Нетриггерные *переходы* часто называют *переходами* по завершении ду-деятельности. Для них рядом со стрелкой *перехода* не указывается никакого имени *события*, а в исходном *состоянии* должна быть описана внутренняя ду-*деятельность*, по окончании которой произойдет тот или иной нетриггерный *переход*.

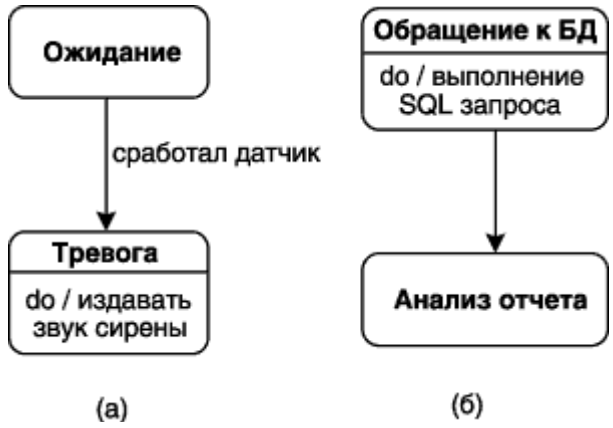

Рис. 3-5.4. Графическое изображение триггерного (а) и нетриггерного (б) переходов на диаграмме состояний

Сторожевое условие (guard condition) - **логическое условие, записанное в прямых скобках и представляющее собой булевское выражение**.

При этом булевское *выражение* должно принимать одно из двух взаимно исключающих значений: "*истина*" или "*ложь*". Из контекста *диаграммы состояний* должна явно следовать *семантика* этого выражения, а для записи выражения может использоваться обычный язык, *псевдокод* или *язык программирования*.

Дополнение триггерных и нетриггерных *переходов сторожевыми условиями* позволяет явно специфицировать семантику их срабатывания. Если *сторожевое*

условие принимает *значение* "*истина*", то соответствующий *переход* при наступлении события-триггера или завершении деятельности может сработать, в результате чего *объект* перейдет в целевое *состояние*. Если же *сторожевое условие* принимает *значение* "*ложь*", то *переход* не может сработать, даже если произошло событие-триггер или завершилась *деятельность* в исходном *состоянии*. Очевидно, в случае невыполнения *сторожевого условия* моделируемый *объект* или система останется в исходном *состоянии*.
Однако *вычисление* истинности *сторожевого условия* в модели происходит только после возникновения ассоциированного с ним события-триггера или завершения деятельности, которые инициируют соответствующий *переход*.

Поскольку общее количество выходящих *переходов* из любого *состояния* в языке *UML* не ограничено, хотя и является конечным, не исключена ситуация, когда из одного *состояния* могут выходить несколько *переходов* с идентичным событием-триггером. Каждый такой *переход* должен содержать собственное *сторожевое условие*, при этом никакие два или более *сторожевых условий* не должны одновременно принимать *значение* "*истина*". В противном случае на *диаграмме состояний* возникнет *конфликт* триггерных *переходов*, что делает несостоятельной (ill formed) модель системы в целом.

Аналогичное замечание справедливо и для нетриггерных *переходов*, когда из одного *состояния* выходят несколько *переходов* по завершении деятельности. Каждый из таких *переходов* также должен содержать собственное *сторожевое условие*, при этом никакие два или более *сторожевых условий* не должны одновременно принимать *значение* "*истина*". В противном случае на *диаграмме состояний* будет иметь *место конфликт* нетриггерных *переходов*, что также делает несостоятельной (ill formed) модель системы в целом.

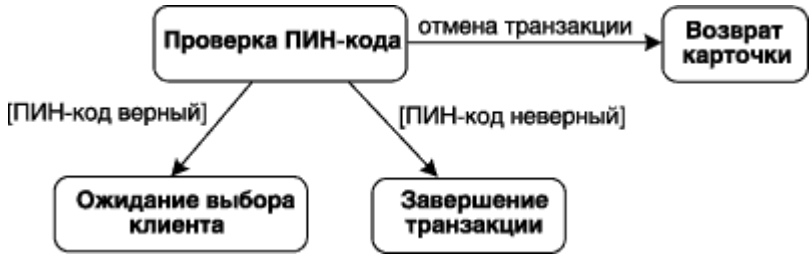

Рис. 3-5.5. Триггерные и нетриггерные переходы на диаграмме состояний

Изображенный фрагмент *диаграммы состояний* (рис. 3-5.5) моделирует изменение *состояний* банкомата при проверке ПИН-кода. Нетриггерные *переходы* на данной диаграмме помечены *сторожевыми условиями*, которые исключают *конфликт* между ними. Что касается триггерного *перехода*, помеченного *событием отмена транзакции*, то он происходит независимо от проверки ПИН-кода в том случае, когда клиент решил отказаться от ввода ПИН-кода.

Выражение действия (action expression) представляет собой вызов операции или передачу сообщения, имеет атомарный характер и выполняется сразу после срабатывания соответствующего перехода до начала действий в целевом состоянии.

Выражение действия выполняется в том и только в том случае, когда *переход* срабатывает. Атомарность действия означает, что оно не может быть прервано никаким другим действием до тех пор, пока не закончится его выполнение. Данное действие может оказывать влияние как на сам *объект*, так и на его окружение, если это с очевидностью следует из контекста модели. Данное *выражение* записывается после знака "/" в строке текста, присоединенной к соответствующему *переходу*.

В общем случае, *выражение* действия может содержать *целый список* отдельных действий, разделенных символом ";". Обязательное требование - все действия из списка должны четко

различаться между собой и следовать в порядке их записи. На *синтаксис* записи выражений действия не накладывается никаких ограничений. Главное - их *запись* должна быть понятна разработчикам модели и программистам. Поэтому чаще всего выражения записывают на одном из языков программирования, который предполагается использовать для реализации модели.

В качестве примера выражения действия *перехода* (рис. 3-5.6) может служить *отображение* сообщения на экране банкомата в том случае, когда запрашиваемая клиентом сумма превосходит *остаток* на его счету. В случае если *кредит* не превышен, то происходит *переход* в *состояние* получения наличных.

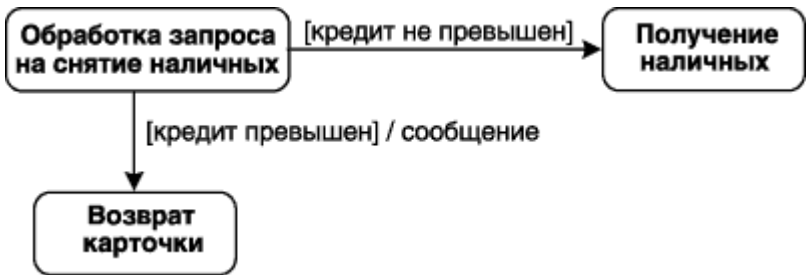

Рис. 3-5.6. Выражение действия перехода на диаграмме состояний

Формализм конечных автоматов допускает вложение одних *конечных автоматов* в другие для уточнения внутренней структуры отдельных более общих *состояний*. В этом случае вложенные *конечные автоматы* получили название конечных подавтоматов. Подавтоматы могут использоваться для внутренней спецификации процедур и функций, реализация которых обусловливает поведение моделируемой системы или объекта. *Моделирование* параллельного поведения с помощью вложенных *диаграмм состояний* рассматривается в лекции 3 часть 6.