Зертханалық жұмыс №6

Күй диаграммасы.

**ЛАБОРАТОРНАЯ РАБОТА № 6**

**ДИАГРАММА СОСТОЯНИЙ**

**Цель работы**:закрепление теоретических сведений о диаграмме состояний; овладение практическими навыками моделирования процессов, описывающих вза­имодействие объектов в диаграмме состояний.

**Краткие теоретические сведения**

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** (statechart diagram) - диаграмма, которая представляет конечный ав­томат, в которой вершины обозначают состояния, а дуги показывают переходы между двумя со­стояниями.

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 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. 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. 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.

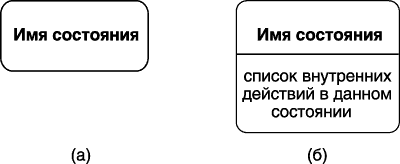
**Состояние** (state) - условие или ситуация в ходе жизненного цикла объекта, в течение которого он удовлетворяет логическому условию, выполняет определенную деятельность или ожидает события.

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. 43). Этот прямоугольник может быть разделен на две секции. Если указана лишь одна секция, то в ней записывается только имя состояния (рис. 43, а). В противном случае в первой из них записывается имя состояния, а во второй - спи­сок некото­рых внутренних действий или переходов в данном состоянии (рис. 43, 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. 43. Графическое изображение состояний на диаграмме состояний



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.

*Пример:* аутентификацию клиента для доступа к ресурсам моделируемой ин­формационной системы (рис. 44). 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.

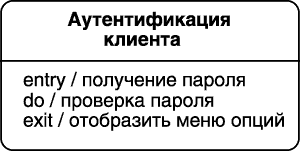


Рис. 44.  Пример состояния с непустой секцией внутренних действий

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. 45, а), из которого может только выходить стрелка-переход.

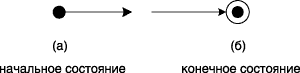


Рис. 45.  Графическое изображение начального и конечного состояний

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. 45, 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.

Пребывание объекта в первом состоянии может сопровож­даться выполнением некоторых внутренних действий или деятельности. При этом изменение те­кущего состояния объекта будет возможно либо после завершения этих действий, 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. Каждый переход может быть помечен строкой текста, которая имеет следующий общий фор­мат: <имя события> '('<список параметров>')' '['<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.

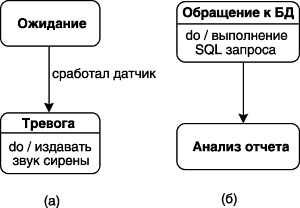
Переход называется **триггерным**, если его специфицирует событие-триггер, связанное с внешними условиями по отношению к рассматриваемому состоянию.

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. 46, 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. 46.  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 не ограничено, 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.

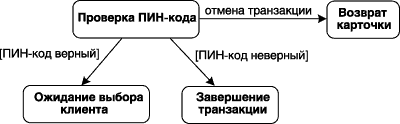


Рис. 47.  Триггерные и нетриггерные переходы на диаграмме состояний

Изображенный фрагмент диаграммы состояний (рис. 47) 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. 48) 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.

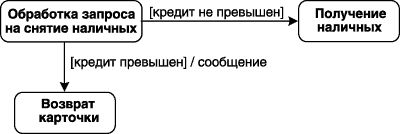


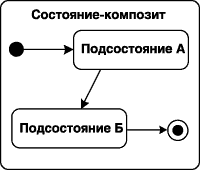
Рис. 48.  Выражение действия перехода на диаграмме состояний

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 сложных объектов и систем связано с многоуровневым пред­ставлением их состояний. В этом случае возникает необходимость детализировать отдельные со­стояния, сделав их составными.

**Составное состояние** (composite state) - сложное состояние, которое состоит из других вло­женных в него состояний (состояние-композит). Вложенные состояния вы­ступают по отношению к составному состоянию как **подсостояния** (substate). И хотя между ними имеет место отношение композиции, графически все вершины диаграммы, которые соответствуют вложенным состояниям, изображаются внутри символа составного состояния (рис. 49). В этом случае размеры графического сим­вола составного состояния увеличиваются, так чтобы вместить в себя все подсосто­яния.

Рис. 49.  Графическое представление составного состояния с двумя вложенными в него последо­вательными подсостояниями



Составное состояние может содержать или несколько последовательных подсо­стояний, или несколько параллельных конечных подавтоматов. Каждое состояние-композит может уточняться только одним из указанных способов. При этом любое из подсостояний, в свою очередь, может быть состоянием-композитом и содержать внутри себя другие вложенные подсостояния. Количе­ство уровней вложенности составных состояний в языке UML не фиксировано.

**Последовательныеподсостояния** (sequential substates) - вложенные состояния состоя­ния-композита, в рамках которого в каждый момент времени объект может находиться в одном и только одном подсостоянии.

Поведение объекта в этом случае представляет собой последовательную смену подсостоя­ний, от начального до конечного. Моделируемый объект или система про­должает находиться в составном состоянии, тем не менее, введение в рассмотрение последовательных подсостояний позволяет учесть более тонкие логические аспекты его внутреннего поведения.

В качестве примера моделируемой системы стоит рассмотреть обычный теле­фонный аппа­рат. Он может находиться в различных состояниях, в частности в со­стоянии дозвона до абонента. Очевидно, для того чтобы позвонить, необходимо снять телефонную трубку, услышать тоновый сигнал, после чего набрать нужный телефонный номер. Таким образом, состояние дозвона до або­нента является состав­ным и состоит из двух последовательных подсостояний: Телефонная трубка поднята и Набор телефонного номера. Фрагмент диаграммы состояний для этого примера содер­жит одно состояние-композит, которое состоит из двух последовательных подсостояний (рис. 50).

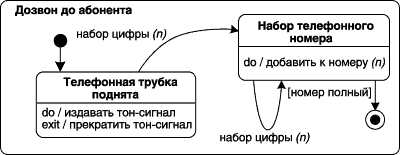


Рис. 50.  Пример составного состояния с двумя вложенными последователь­ными подсо­стояниями

Некоторых пояснений могут потребовать переходы. Два из них специфицируют событие-триггер, которое имеет имя: набор цифры(n) с параметром n. В качестве параметра, как нетрудно предположить, выступает отдельная цифра на диске теле­фонного аппарата. Переход из начального подсостояния не содержит никакой строки текста. Последний переход в конечное подсостояние также не имеет собы­тия-триггера, но имеет сторожевое условие, проверяющее полноту набран­ного но­мера абонента. Только в случае истинности этого условия телефонный аппарат мо­жет пе­рейти в конечное состояние для состояния-композита Дозвон до абонента.

Каждое составное состояние должно содержать в качестве вложенных состоя­ний начальное и конечное состояния. При этом начальное подсостояние является исходным, когда происходит переход объекта в данное составное состояние. Если составное состояние содержит внутри себя конечное состояние, то переход в это вложенное конечное состояние означает завершение нахож­дения объекта в данном составном состоянии. Важно помнить, что для последовательных подсо­стояний начальное и конечное состояния должны быть единственными в каждом составном состо­янии.

Это можно объяснить следующим образом. Каждая совокупность вложенных последова­тельных подсостояний представляет собой конечный подавтомат того конечного автомата, кото­рому принадлежит рассматриваемое составное состояние. Поскольку каждый конечный автомат может иметь по определению единственное начальное и единственное конечное состояния, то для любого его конечного подав­томата это условие также должно выполняться.

**Параллельные подсостояния** (concurrent substates) - вложенные состояния, ис­пользу­емые для спецификации двух и более конечных подавтоматов, которые могут выполняться параллельно внутри составного состояния.

Каждый из конечных подавтоматов занимает некоторую графическую область внутри со­ставного состояния, которая отделяется от остальных горизонтальной пунктирной линией. Если на диаграмме состояний имеется составное состояние с вложенными параллельными подсостояни­ями, то объект может одновременно нахо­диться в каждом из этих подсостояний.

Отдельные параллельные подсостояния могут, в свою очередь, состоять из не­скольких по­следовательных подсостояний (рис. 51). В этом случае по определению моделируемый объект может находиться только в одном из последовательных под­состояний каждого подавтомата. Та­ким образом, для фрагмента диаграммы состоя­ний (рис. 51) допустимо одновременное нахожде­ние объекта только в следующих подсостояниях: (А, В, Г), (Б, В, Г), (А, В, Д), (Б, В, Д).

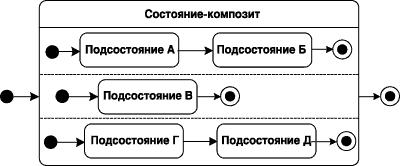


Рис. 51.  Графическое изображение состояния-композита с вложенными парал­лельными подсостояниями

**Несовместимое подсостояние** (disjoint substate) - подсостояние, в котором под­система не может находиться одновременно с другими подсостояниями одного и того же составного состояния.

В этом контексте недопустимо нахождение объекта одновременно в несовме­стимых подсо­стояниях (А, Б, В) или (В, Г, Д).

Поскольку каждый регион вложенного состояния специфицирует некоторый конечный по­давтомат, то для каждого из вложенных конечных подавтоматов могут быть определены соб­ственные начальное и конечное состояния.

При переходе в данное составное состояние каждый из конечных подавтоматов оказывается в своем начальном состоянии. Далее происходит параллельное выполнение каждого из этих конечных подавтоматов, причем выход из составного состояния будет возможен лишь в том случае, когда все конечные подавтоматы будут находиться в своих ко­нечных состояниях. Если какой-либо из конечных подавтоматов пришел в свое фи­наль­ное состояние раньше других, то он должен ожидать, пока и другие подавто­маты не придут в свои финальные состояния.

В некоторых случаях бывает желательно скрыть внутреннюю структуру состав­ного состоя­ния. Например, отдельный конечный подавтомат, специфицирующий составное состояние, может быть настолько большим по масштабу, что его визуали­зация затруднит общее представление диа­граммы состояний. В подобной ситуации допускается не раскрывать на исходной диаграмме со­стояний данное составное состояние, а указать в правом нижнем углу специальный символ-пикто­грамму (рис. 52).

В последующем диаграмма состояний для соответствующего конечного подав­томата может быть изображена отдельно от основной диаграммы с необходи­мыми комментари­ями.



Рис. 52.  Составное состояние со скрытой внутренней структурой и специаль­ной пикто­граммой

Обычный конечный автомат не позволяет учитывать предысторию в процессе моделирова­ния поведения систем и объектов. Однако функционирование ряда си­стем основано на возможно­сти выхода из отдельного состояния-композита с после­дующим возвращением в это же состояние.

Может оказаться необходимым учесть ту часть деятельности, которая была выполнена на момент выхода из этого состоя­ния-композита, чтобы не начинать ее выполнение сначала. Для этой цели в языке UML существует историческое состояние.

**Историческое состояние** (history state) - псевдосостояние, используемое для за­поминания того из последовательных подсостояний, которое было текущим в мо­мент выхода из составного состояния.

Историческое состояние применяется только в контексте составного состояния. При этом существует две разновидности исторического состояния: неглубокое или недавнее и глубокое или давнее (рис. 53).



Рис. 53.  Графическое изображение недавнего (а) и давнего (б) исторического состояния

Неглубокое историческое состояние (shallow history state) обозначается в форме небольшой окружности, в которую помещена латинская буква "H" (рис. 53, а). Это состояние обладает сле­дующей семантикой. Во-первых, оно является первым под­состоянием в составном состоянии, и переход извне в рассматриваемое составное состояние должен вести непосредственно в данное историческое состояние. Во-вто­рых, при первом попадании в неглубокое историческое состояние оно не хранит никакой истории. Другими словами, при первом переходе в недавнее историческое состояние оно заменяет собой начальное состояние соответствующего конечного подавтомата.

Далее могут последовательно изменяться вложенные подсостояния. Если в не­который мо­мент происходит выход из составного состояния (например, в случае наступления некоторого со­бытия), то рассматриваемое историческое состояние за­поминает то из подсостояний, которое было текущим на момент выхода из данного составного состояния. При последующем входе в это со­ставное состояние неглубо­кое историческое подсостояние имеет непустую историю и сразу от­правляет конеч­ный подавтомат в запомненное подсостояние, минуя все предшествующие ему под­состояния.

Историческое состояние теряет свою историю в тот момент, когда конечный по­давтомат доходит до своего конечного состояния. При этом неглубокое историче­ское состояние запоминает историю только того конечного подавтомата, к которому оно относится. Другими словами, этот тип псевдосостояния способен запомнить историю только одного с ним уровня вложенности.

Глубокое историческое состояние (deep history state или состояние глубокой ис­тории) также обозначается в форме небольшой окружности, в которую помещена латинская буква "H" с допол­нительным символом "\*" (рис. 53, б), и служит для за­поминания всех подсостояний любого уровня вложенности для исходного состав­ного состояния.

В отдельных случаях возникает необходимость явно показать ситуацию, когда переход мо­жет иметь несколько исходных состояний или целевых состояний. Такой переход получил назва­ние - **параллельный переход***.* Введение в рассмотрение па­раллельных переходов может быть обусловлено необходимостью синхронизировать и/или разделить отдельные процессы управления на параллельные нити без специ­фикации дополнительной синхронизации в параллельных конеч­ных подавтоматах.

Графически такой переход изображается вертикальной черточкой, аналогично обозначению перехода в известном формализме сетей Петри. Если параллельный переход имеет две или более исходящих из него дуг (рис. 54, а), то его называют разделением (fork). Если же он имеет две или более входящие дуги (рис. 54, б), то его называют слиянием (join). Текстовая строка специфика­ции параллельного пере­хода записывается рядом с черточкой и относится ко всем входящим или исходя­щим дугам.

Срабатывание параллельного перехода: в первом случае происходит разделение составного конечного автомата на два конечных подавтомата, образующих парал­лельные ветви вложенных подпроцессов. При этом после срабатывания перехода-разделения моделируемая система или объект одновременно будет находиться во всех целевых подсостоя­ниях этого параллельного перехода (подсостояния 1 и 2). Далее процесс изменения состояний бу­дет протекать согласно ранее рассмотренным правилам для составных состояний.

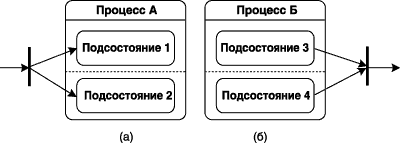


Рис. 54.  Графическое изображение перехода-разделения в параллельные под­состоя­ния (а) и перехода-слияния из параллельных подсостояний (б)

Во втором случае переход-слияние срабатывает, если имеет место событие-триггер для всех исходных состояний этого перехода, и выполнено (при его нали­чии) сторожевое условие. При срабатывании перехода-слияния одновременно поки­даются все исходные подсостояния перехода (подсостояния 3 и 4) и происходит переход в целевое состояние. При этом каждое из исходных подсостояний перехода должно принадлежать отдельному конечному подавтомату, входящему в составной конечный автомат (процессу Б).

Переход, стрелка которого соединена с границей составного состояния, обозна­чает переход в это составное состояние (переход a на рис. 55). Он эквивалентен пе­реходу в начальное состоя­ние каждого из конечных подавтоматов (единствен­ному на рис. 55), входящих в состав данного состояния-композита.

Переход f, вы­ходящий из составного состояния (рис. 55), относится к каж­дому из вложенных состояний. Это означает, что моделируемая система или объект при наступле­ния события f может покинуть данное составное состояние, находясь в любом из его вложенных состояний В и Г.

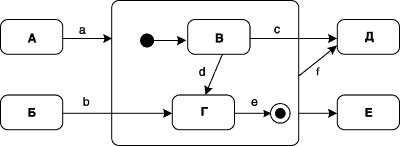


Рис. 55.  Варианты переходов в составное состояние и из составного состояния

Иногда желательно реализовать ситуацию, когда выход из отдельного вложен­ного состоя­ния соответствовал бы также выходу из составного состояния. В этом случае изображают переход, который непосредственно выходит из вложенного со­стояния и пересекает границу состояния-композита (переход c на рис. 55). Анало­гично, допускается изображение переходов, входящих извне состояния-композита в отдельное вложенное состояние (переход b на рис. 55).

Переход d является внутренним для рассматриваемого состояния-композита и никак не влияет на выход из состояния-композита. Выход из данного составного состояния также возможен при наступлении события e, которое приводит в его ко­нечное состояние, а из него - в состояние Е, находящееся вне данного состояния-композита.

**Состояние синхронизации** (synch state) - псевдосостояние в конечном авто­мате, которое ис­пользуется для синхронизации параллельных областей конечного авто­мата.

Синхронизирующее состояние обозначается небольшой окружностью, внутри которой по­мещен символ звездочки "\*". Оно используется совместно с пере­ходом-слиянием или переходом-разделением для того, чтобы явно указать события в дру­гих конечных подавтоматах, оказываю­щие непосредственное влияние на пове­дение данного подавтомата.

Так, например, при включении компьютера с некоторой сетевой операционной системой параллельно начинается выполнение нескольких процессов. В частности, происходит проверка пароля пользователя и запуск различных служб. При этом работа пользователя на компьютере станет возможной только в случае успешной его аутентификации, в противном случае компьютер может быть выключен.

Рас­смотренные особенности синхронизации этих параллельных процессов учтены на соответствующей диаграмме состояний с помощью синхронизирующего состояния (рис. 56).

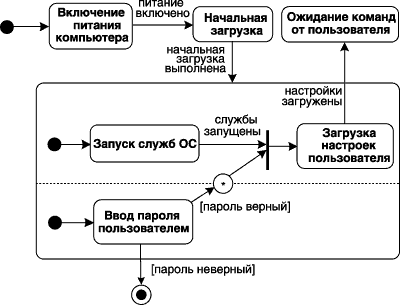


Рис. 56.  Диаграмма состояний для примера включения компьютера

Наличие у системы нескольких со­стояний, отличающихся от простых, таких как "исправен - неисправен", "активен - неактивен", "ожидание - реакция на внешние действия", служит признаком необходимости построения диаграммы состояний. В качестве начального варианта диаграммы состояний, если нет очевидных соображе­ний по поводу состояний объекта, можно воспользоваться подобными состояниями, в качестве составных, уточняя их (детализируя их внутреннюю структуру) по мере рассмотрения логики поведения моделируемой системы или объекта.

При выделении состояний и переходов следует помнить, что длительность сра­батывания отдельных переходов должна быть существенно меньшей, чем нахожде­ние моделируемых эле­ментов в соответствующих состояниях. Каждое из состояний должно характеризоваться опреде­ленной устойчивостью во времени. Другими сло­вами, из каждого состояния на диаграмме не мо­жет быть самопроизвольного пере­хода в какое бы то ни было другое состояние.

Все переходы должны быть явно спе­цифицированы, в противном случае построенная диаграмма состояний яв­ляется либо неполной (неадекватной), либо ошибочной с точки зрения нотации языка UML (ill formed).

При разработке диаграммы состояний нужно постоянно следить, чтобы объект в каждый момент мог находиться только в единственном состоянии. Если это не так, то данное обстоятель­ство может быть как следствием ошибки, так и неявным при­знаком наличия параллельности пове­дения у моделируемого объекта. В последнем случае следует явно специфицировать необходимое число конечных подавтоматов, вложив их в то составное состояние, которое характеризуется нарушением условия одновременности.

*Пример:*Программное средство представляет собой базу данных и обеспечивает корректировку данных, а именно в БД «Группы» может изменяться пере­чень пред­метов в соответствие с курсом группы и отде­лением; в БД «Пред­меты» могут из­меняться номера аудиторий и фамилии пре­по­давателей; осуществ­ляет поиск по ФИО преподавателя, номеру аудитории, названию предмета и номеру группы. Про­граммное средство составляет распи­сание работы для конкретного пре­подавателя на не­делю; для группы на неделю; для группы по конкретному пред­мету, а также отчет о загрузке аудиторий на каждый день.

****

Рис. 57. Пример диаграммы состояний

**Порядок выполнения работы**

1. Изучить теоретические сведения по теме “Диаграмма состояний”.
2. Разработать диаграмму состояний для произвольной системы индивидуаль­ного задания (Приложение А).
3. Оформить отчет, включив в него описание всех компонентов диаграммы со­стояний согласно индивидуальному варианту задания, ответы на контрольные во­просы.

**Контрольные вопросы**

1. Дайте определение понятию «диаграмма состояний».
2. Опишите назначение диаграммы состояний.
3. Дайте определение понятию «конечный автомат».
4. Дайте определение понятиям «состояния», «действие», «псевдосостояние». Графическое изображение состояния.
5. Что представляет собой переход? Какие бывают переходы, в чем их разли­чие?
6. Дайте определение понятию «событие».
7. Дайте определения следующим понятиям: «составное состояние», «последова­тельные подсостояния», «параллельные подсостояния», «несовместимое подсостояние», «историческое состояние», «параллельный переход», «состояние синхронизации».