24. Диаграмма состояний (statechart diagram). Автоматы. Состояние. Переход. Составное состояние и подсостояние. Историческое состояние. Сложные переходы.

Диаграмма состояний (statechart diagram)

Каждая диаграмма состояний в UML описывает все возможные состояния одного экземпляра определенного класса и возможные последовательности его переходов из одного состояния в другое, то есть моделирует все изменения состояний объекта как его реакцию на внешние воздействия.

Диаграммы состояний чаще всего используются для описания поведения отдельных объектов, но также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Диаграмма состояний является графом специального вида, который представляет некоторый автомат. Вершинами графа являются возможные состояния автомата, изображаемые соответствующими графическими символами, а дуги обозначают его переходы из состояния в состояние. Диаграммы состояний могут быть вложены друг в друга для более детального представления отдельных элементов модели.

Автоматы в UML

В метамодели UML ***автомат*** является пакетом, в котором определено множество понятий, необходимых для представления поведения моделируемой сущности в виде дискретного пространства с конечным числом состояний и переходов.

Длительность нахождения системы в любом из возможных состояний существенно превышает время, которое затрачивается на переход из одного состояния в другое. Предполагается, что в пределе время перехода может быть равно нулю (если дополнительно не оговорено другое), то есть смена состояний объекта может происходить мгновенно.

Поведение автомата моделируется как последовательное перемещение по графу от вершины к вершине с учетом ориентации связывающих их дуг.

Для автомата должны выполняться следующие обязательные условия:

- состояние, в которое может перейти объект, определяется только его текущим состоянием и не зависит от предыстории;
- в каждый момент времени автомат может находиться только в одном из своих состояний. При этом, автомат может находиться в отдельном состоянии как угодно долго, если не происходит никаких событий;
- время нахождения автомата в том или ином состоянии, а также время достижения того или иного состояния никак не специфицируются;
- количество состояний автомата должно быть конечным и все они должны быть специфицированы явным образом. Отдельные псевдосостояния могут не иметь спецификаций (начальное и конечное состояния). В этом случае их назначение и семантика полностью определяются из контекста модели и рассматриваемой диаграммы состояний;

- граф автомата не должен содержать изолированных состояний и переходов. Для каждого состояния, кроме начального, должно быть определено предшествующее состояние, а каждый переход должен соединять два состояния автомата;
- автомат не должен содержать конфликтующих переходов, когда объект одновременно может перейти в два и более последующих состояния (кроме случая параллельных подавтоматов). В языке UML исключение конфликтов возможно на основе введения сторожевых условий

Понятие состояния объекта

Понятие ***состояния*** (state) является фундаментальным не только в метамодели языка UML, но и в прикладном системном анализе. Вся концепция динамической системы основывается на понятии состояния. Семантика же состояния в языке UML имеет ряд специфических особенностей.

В языке UML под состоянием понимается абстрактный метакласс, используемый для моделирования отдельной ситуации, в течение которой выполняются некоторые условия. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта. Изменение отдельных значений атрибутов будет отражать изменение состояния моделируемого класса или объекта.

Состояние на диаграмме изображается ***прямоугольником со скругленными вершинами***. Прямоугольник может быть разделен на две секции горизонтальной линией. Если указана лишь одна секция, то в ней записывается только имя состояния. При наличии двух секций, в первой из них записывается имя состояния, а во второй список некоторых внутренних действий или переходов в данном состоянии. Под действием в языке UML понимают некоторую атомарную операцию, выполнение которой приводит к изменению состояния или возврату некоторого значения (например, «истина» или «ложь»).

Имя состояния представляет собой строку текста, которая раскрывает его содержательный смысл. Имя всегда записывается с заглавной буквы. Поскольку состояние системы является составной частью процесса ее функционирования, рекомендуется в качестве имени использовать глаголы в настоящем времени (звенит, печатает, ожидает) или соответствующие причастия (занят, свободен, передано, получено). Имя у состояния может отсутствовать и этом случае состояние является анонимным. Если на диаграмме анонимных состояний несколько, то они должны различаться между собой.

Список внутренних действий содержит перечень действий или деятельностей, которые выполняются во время нахождения моделируемого элемента в данном состоянии. Каждое из действий записывается в виде отдельной строки и имеет следующий формат:

<метка-дёйствия '/' 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, которые принадлежат области имен или контексту моделируемого объекта. Если список выражений действия пустой, то разделитель в виде наклонной черты '/' может не указываться.

Перечень меток действия имеет фиксированные значения, которые не могут быть использованы в качестве имен событий. Эти значения следующие:

- entry - эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент входа в данное состояние (входное действие);
- exit - эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент выхода из данного состояния (выходное действие);
- do - эта метка специфицирует выполняющуюся деятельность («do activity»), которая выполняется в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не закончится вычисление, специфицированное следующим за ней выражением действия. В этом случае при завершении события формируется соответствующий результат;
- include - эта метка используется для обращения к подавтомату, при этом следующее за ней выражение действия содержит имя этого подавтомата.

Во всех остальных случаях метка действия идентифицирует событие, которое запускает соответствующее выражение действия. Эти события называются внутренними переходами и семантически эквивалентны переходам в само это состояние, за исключением той особенности, что выход из этого состояния или повторный вход в него не происходит и действия входа и выхода не выполняются.

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояние). В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме графической области, от которой начинается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка, из которого может только выходить стрелка, соответствующая переходу.

На самом верхнем уровне представления объекта переход из начального состояния может быть помечен событием создания (инициализации) данного объекта. В противном случае переход никак не помечается. Если этот переход не помечен, то он является первым переходом в следующее за ним состояние.

Конечное состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояние). В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени. Оно служит для указания на диаграмме 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, которую может только входить стрелка, соответствующая переходу.

Переход

Простой переход (simple transition) представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния объекта другим. Если пребывание моделируемого объекта в первом состоянии сопровождается выполнением некоторых действий, то переход во второе состояние будет возможен только после завершения этих действий и, возможно, после выполнения некоторых дополнительных условий, называемых сторожевыми условиями.

На диаграмме состояний переход изображается сплошной линией со стрелкой, которая направлена в целевое состояние. Каждый переход может быть помечен строкой текста, которая имеет следующий общий формат:

<сигнатура события>['<сторожевое условие>']' <выражение действия>.

При этом ***сигнатура события*** описывает некоторое событие с необходимыми аргументами:

<имя события>'('<список параметров, разделенных запятыми>')'.

Событие (event) является самостоятельным элементом языка UML. Формально, событие представляет собой спецификацию некоторого 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 некоторого события уже нельзя вернуться к предыдущим событиям, если такая возможность не предусмотрена явно в модели.

Семантика понятия события фиксирует внимание на внешних проявлениях качественных изменений, происходящих при переходе моделируемого объекта из состояния в состояние. 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.

Имя события идентифицирует каждый отдельный переход на диаграмме состояний и может содержать строку текста, начинающуюся со строчной буквы. В этом случае принято считать переход триггерным, то есть таким, который специфицирует событие-триггер. Если рядом со стрелкой перехода не указана никакая строка текста, то соответствующий переход является нетриггерным, и в этом случае из контекста диаграммы состояний должно следовать, после окончания какой деятельности он выполняется. После имени события могут следовать круглые скобки для явного задания параметров соответствующего события-триггера. Если таких параметров нет, то список параметров со скобками может отсутствовать.

Сторожевое условие (guard condition), если оно есть, всегда записывается в прямых скобках после события-триггера и представляет собой некоторое булевское выражение. Из контекста диаграммы состояний должна явно следовать семантика этого выражения, а для записи выражения может использоваться синтаксис языка объектных ограничений.

Введение для перехода сторожевого условия позволяет явно специфицировать семантику его срабатывания. Однако вычисление истинности сторожевого условия происходит только после возникновения ассоциированного с ним события-триггера, инициирующего соответствующий переход.

В общем случае из одного состояния может быть несколько переходов с одним и тем же событием-триггером. При этом никакие два сторожевых условия не должны одновременно принимать значение «истина». Каждое из сторожевых условий необходимо вычислять всякий раз при наступлении соответствующего события-триггера.

Примером события-триггера может служить разрыв телефонного соединения с провайдером интернет-услуг после окончания загрузки электронной почты клиентской почтовой программой (при удаленном доступе к интернет). В этом случае сторожевое условие есть не что иное, как ответ на вопрос: «Пуст ли почтовый ящик клиента на

сервере провайдера?». В случае положительного ответа - «истина», следует отключить соединение с провайдером, что и делает автоматически почтовая программа-клиент. В случае отрицательного ответа - «ложь», следует оставаться в состоянии загрузки почты и не разрывать телефонное соединение.

Выражение действия (action expression) выполняется только при срабатывании перехода. Оно представляет собой атомарную операцию, выполняемую сразу после срабатывания соответствующего перехода до начала каких бы то ни было действий в целевом состоянии. Атомарность действия означает, что оно не может быть прервано никаким другим действием до тех пор, пока не закончится его выполнение. Данное действие может влиять как на сам объект, так и на его окружение, если это с очевидностью следует из контекста модели.

В общем случае выражение действия может содержать целый список отдельных действий, разделенных символом «**;**». Все действия списка должны различаться между собой и следовать в порядке записи. На синтаксис записи выражений действия не накладывается никаких ограничений. Основное условие, чтобы запись была понятна разработчикам модели и программистам. Как правило, выражение действия записывают на одном из языков программирования, который предполагается использовать для реализации модели.

Составное состояние и подсостояние

Составное состояние (composite state) это сложное состояние, состоящее из других вложенных в него состояний. Вложенные состояния выступают по отношению к сложному состоянию как подсостояия (substate). Хотя между ними имеет место отношение композиции, графически все вершины диаграммы, которые соответствуют вложенным состояниям, изображаются внутри символа составного состояния. В этом случае размеры графического символа составного состояния увеличиваются, так чтобы вместить в себя все подсостояния.

Рис. 1. Изображение составного состояния

Составное состояние может содержать два или более параллельных подавтомата или несколько последовательных подсостояний. Каждое составное состояние может уточняться только одним из указанных способов. При этом любое из подсостояний также может являться составным состоянием и содержать внутри себя другие вложенные подсостояния. Количество уровней вложенности составных состояний не ограничено в языке UML.

Последовательные подсостояния (sequential substates) используются для моделирования такого поведения объекта, во время которого в каждый момент времени объект может находиться в одном и только одном подсостояний. Поведение объекта в

этом случае представляет собой последовательную смену подсостояний, начиная от начального и заканчивая конечным. Введение в рассмотрение последовательных подсостояний позволяет учесть более тонкие логические аспекты внутреннего поведения объекта.

Составное состояние может содержать в качестве вложенных подсостояний 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 подсостояние, то переход в это вложенное конечное состояние означает завершение нахождения объекта в данном вложенном состоянии. Для последовательных подсостояний начальное и конечное состояния должны быть единственными в каждом составном состоянии.

Параллельные подсостояния (concurrent substates) позволяют специфицировать два и более подавтомата, которые могут выполняться параллельно внутри составного события. Каждый из подавтоматов занимает некоторую область или регион 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 состоять из нескольких последовательных подсостояний (подавтоматы 1 и 3 на рис. 2).

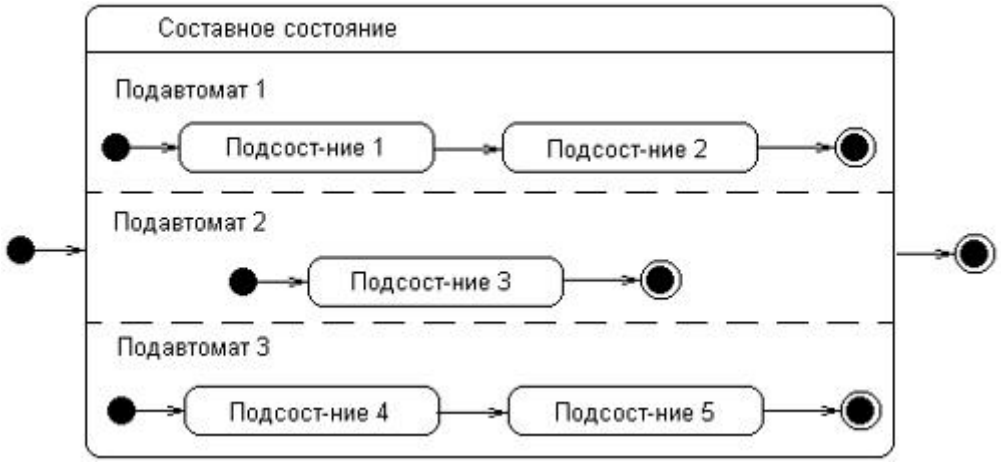

Рис. 2. Изображение составного состояния с вложенными параллельными подсостояниями

Поскольку каждый регион вложенного состояния специфицирует некоторый подавтомат, то для каждого из вложенных подавтоматов могут быть определены собственные начальное и конечные подсостояния (рис. 2). При переходе в данное составное состояние каждый из подавтоматов оказывается в своем начальном подсостоянии. Далее происходит параллельное выполнение каждого из этих подавтоматов, причем выход из составного состояния будет возможен лишь в том случае, когда все подавтоматы будут находиться в своих конечных подсостояниях. Если какой-либо из подавтоматов пришел в свое конечное состояние раньше других, то он должен ожидать, пока и другие подавтоматы не придут в свои конечные состояния.

В некоторых случаях бывает желательно скрыть внутреннюю структуру составного состояния. Например, отдельный подавтомат, специфицирующий составное состояние,

может быть настолько большим по размеру, что его визуализация затруднит общее представление диаграммы состояний. В подобной ситуации допускается не раскрывать на исходной диаграмме данное составное состояние, а указать в правом нижнем углу специальный символ-пиктограмму (рис. 3). В последующем диаграмма состояний для соответствующего подавтомата может быть изображена отдельно с необходимыми комментариями.

Рис. 3. Составное состояние со скрытой внутренней структурой

Как отмечалось выше, правила построения обычного автомата не позволяют учитывать предысторию в процессе моделирования поведения объектов. Однако функционирование целого ряда систем основано на возможности выхода из отдельных состояний с последующим возвращением в это же состояние. При этом может оказаться необходимым учесть ту часть деятельности, которая была выполнена на момент выхода из этого состоянии, чтобы не начинать ее выполнение сначала. Для этой цели в языке UML существует ***историческое состояние***.

Историческое состояние (history state) применяется в контексте составного состояния. Оно используется для запоминания того из последовательных подсостояний, которое было текущим в момент выхода из составного состояния. При этом существует две разновидности исторического состояния: *недавнее* и *давнее* (рис. 4).

Рис. 4. Изображение недавнего (а) и давнего (б) исторического состояния

Недавнее историческое состояние (shallow history state) обозначается в форме небольшой окружности, в которую помещена латинская буква «H» (рис. 4а). Это состояние обладает следующей семантикой. Во-первых, оно является первым подсостоянием в составном состоянии, и переход извне в это составное состояние должен вести непосредственно в это историческое состояние. Во-вторых, при первом попадании в недавнее историческое состояние оно не хранит никакой истории (история пуста), то есть заменяет собой начальное состояние подавтомата.

Далее следует последовательное изменение вложенных подсостояний. Если в некоторый момент происходит выход из вложенного состояния (например, в случае некоторого внешнего события), то это историческое состояние запоминает то из подсостояний, которое являлось текущим на момент выхода. При следующем входе в это же составное состояние историческое подсостояние уже имеет непустую историю и сразу отправляет подавтомат в запомненное подсостояние, минуя все предшествующие ему подсостояния.

Историческое состояние теряет свою историю в тот момент, когда подавтомат доходит до своего конечного состояния. При этом недавнее историческое состояние запоминает историю только того подавтомата, к которому он относится.

С другой стороны, запомненное состояние, в свою очередь, также может являться составным состоянием. *Давнее историческое состояние* (deep history state) обозначается в форме небольшой окружности, в которую помещена латинская буква «H» с символом «*» (рис. 4б) и служит для запоминания всех подсостояний любого уровня вложенности для текущего подавтомата.

Сложные переходы

Рассмотренное выше понятие перехода является вполне достаточным для большинства типичных расчетно-вычислительных задач. Однако современные программные системы могут реализовывать очень сложную логику поведения отдельных своих компонентов. Может оказаться, что для адекватного представления процесса изменения состояний семантика обычного перехода для них недостаточна. С этой целью в языке UML специфицированы дополнительные обозначения и свойства, которыми могут обладать отдельные переходы на диаграмме состояний.

Переходы между параллельными состояниями

В отдельных случаях переход может иметь несколько состояний-источников и несколько целевых состояний. Такой переход получил название ***параллельный переход*.**

Графически такой переход изображается вертикальной черточкой. Если параллельный переход имеет две и более входящие дуги, его называют соединением (join). Если же он имеет две или более исходящие дуги, то его называют ветвлением (fork). Текстовая строка спецификации параллельного перехода записывается рядом с черточкой и относится ко всем входящим или исходящим дугам.

Срабатывание параллельного перехода происходит следующим образом. Переход-соединение выполняется, если имеет место событие-триггер для всех исходных состояний этого перехода, и выполнено, если таковое есть, сторожевое условие. При срабатывании перехода-соединения одновременно покидаются все исходные состояния перехода и происходит переход в целевое состояние. При этом каждое из исходных состояний перехода должно принадлежать отдельному подавтомату, входящему в состав автомата.

В случае ветвления происходит разделение автомата на два подавтомата, образующих параллельные ветви вложенных подсостояний. После срабатывания перехода моделируемый объект одновременно будет находиться во всех целевых состояниях этого перехода. Далее процесс изменения состояний будет протекать согласно правилам для составных состояний.

Переходы между составными состояниями

Переход, стрелка которого соединена с границей некоторого составного состояния, обозначает переход в составное состояние. Такой переход эквивалентен переходу в начальное состояние каждого из подавтоматов, входящих в состав данного суперсостояния. Переход, выходящий из составного состояния, относится к каждому из вложенных подсостояний. Это означает, что объект может покинуть составное суперсостояние, находясь в любом из его подсостояний. Для этого вполне достаточно выполнения события и сторожевого условия.

Иногда желательно реализовать ситуацию, когда выход из отдельного вложенного состояния соответствовал бы выходу и из составного состояния тоже. В этом случае изображают переход, который непосредственно выходит из вложенного подсостояния за границу суперсостояния. Аналогично, допускается изображение переходов, входящих извне составного состояния в отдельное вложенное состояние.

Синхронизирующие состояния

Как уже было отмечено, поведение параллельных подавтоматов независимо друг от друга, что позволяет реализовать многозадачность в программных системах. Однако, в отдельных случаях может возникнуть необходимость учесть в модели синхронизацию наступления отдельных событий. Для этой цели в языке UML имеется специальное псевдосостояние, которое называется ***синхронизирующим состоянием*.**

Синхронизирующее состояние (synch state) обозначается небольшой окружностью, внутри которой помещен символ звездочки «*». Оно используется совместно с переходом-соединением или переходом-ветвлением для того, чтобы явно указать события в других подавтоматах, оказывающие непосредственное влияние на поведение данного подавтомата.