Лабораторная работа №5

Разработка диаграмм состояний с помощью Visual Paradigm for UML

Цель работы

Целью работы является изучение основ создания диаграмм состояний на языке UML с помощью case-средстваVisual Paradigm for UML CE, применение приобретенных навыков для построения объектно-ориентированных моделей определенной предметной области.

Задачи

Основными задачами практической работы являются:

– получить навыки создания диаграмм состояний с помощью case-средства Visual Paradigm for UML CE;

– изучить основные стереотипы классов.

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.

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 детальногопредставления состояний отдельных элементов модели. Для понимания семантики конкретной диаграммы состояний необходимо представлять особенности поведения моделируемой сущности, а также иметь общиесведения 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 графаконечного 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.

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, приэтом 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 связанос выполнением определенных действий. В последнем случае соответствующая деятельность начинается в моментперехода 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. 1). Этот прямоугольник может быть разделен на две секции. Если указана лишь одна секция, то в ней записывается толькоимя состояния (рис. 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. 1, б). При этом под действием в языкеUML понимают некоторую атомарную операцию, выполнение которой приводит к изменению состояния или возврату некоторого значения (например, «истина» или «ложь»).

Рисунок 1 – 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 онидолжны различаться между собой.

Действие (action) – спецификация выполнимого утверждения, которая образует абстракцию вычислительнойпроцедуры.

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, тометка действия с разделителем в виде наклонной черты '/' не указывается. Перечень меток действий в языке UMLфиксирован, причем эти метки не могут быть использованы в качестве имен событий:

Входное действие (entry action) – действие, которое выполняется в момент перехода в дан-

ное состояние. Обозначается с помощью ключевого слова – метки действия entry, которое указывает на то, что следующее за нейвыражение 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 находитсяв данном состоянии, или до тех пор, пока не будет прервано внешним событием.

Пример: аутентификацию клиента для доступа к ресурсам моделируемой информационной системы (рис. 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. Приуспешном завершении этой проверки выполняется действие на выходе, которое отображает меню доступных для клиента опций.

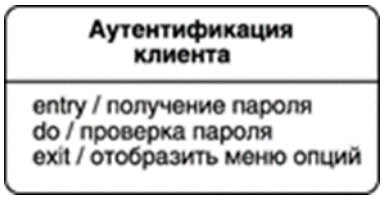

Рисунок 2 – Пример состояния с непустой секцией внутренних действий

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 надиаграмме состояний графической области, от которой начинается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка (рис. 3, а), из которого может тольковыходить стрелка-переход.

Рисунок 3 – 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. Еслиэтот 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, 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 одномуровне вложенности состояний.

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.

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 предусмотренав модели.

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

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 строкитекста, 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. 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 вданном 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 описанавнутренняя ду-деятельность, по окончании которой произойдет тот или иной нетриггерный переход.

Рисунок 4 – Графическое изображение триггерного (а) и нетриггерного (б) переходов на диаграммесостояний

Сторожевое условие (guard condition) – логическое условие, записанное в прямых скобках и представляющеесобой булевское выражение. При этом булевское выражение должно принимать одно из двух взаимноисключающих 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.

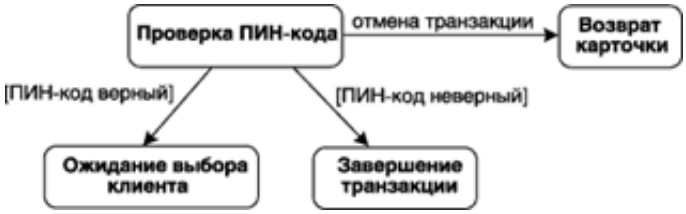

Рисунок 5 – Триггерные и нетриггерные переходы на диаграмме состояний

Изображенный фрагмент диаграммы состояний (рис. 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, имеетатомарный 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, покане 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 качестве примера выражения действия перехода (рис. 6) может служить отображение сообщения на экранебанкомата 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.

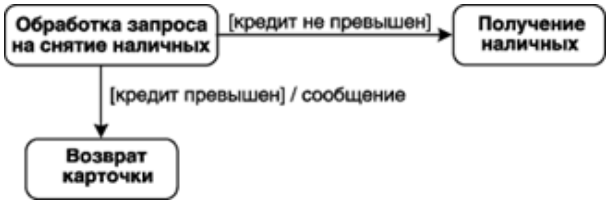

Рисунок 6 – Выражение действия перехода на диаграмме состояний

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. 7). В этом случае размеры графического символа составного состояния увеличиваются, так чтобы вместить всебя все подсостояния.

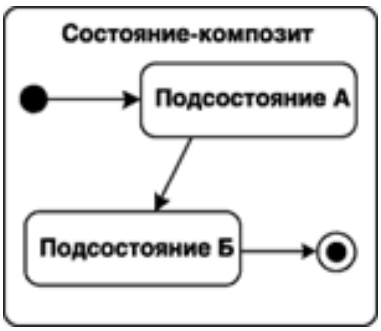

Рисунок 7 – Графическое представление составного состояния с двумя вложенными в негопоследовательными подсостояниями

Составное состояние может содержать или несколько последовательных подсостояний, или несколько параллельных конечных подавтоматов. Каждое состояние-композит может уточняться только одним из указанныхспособов. 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.

Последовательные подсостояния (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 еговнутреннего 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 диаграммысостояний для этого примера содержит одно состояние-композит, которое состоит из двух последовательных подсостояний (рис. 8).

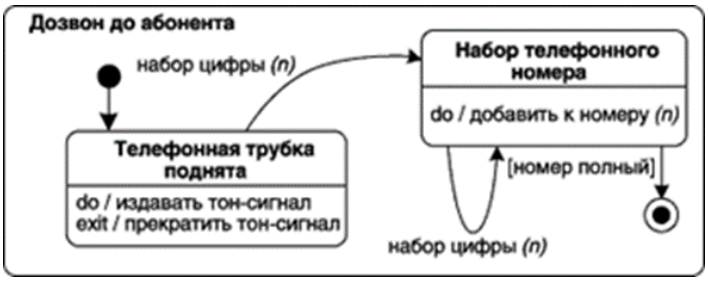

Рисунок 8 – Пример составного состояния с двумя вложенными последовательными подсостояниями

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 спецификациидвух 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 вкаждом из этих подсостояний.

Отдельные параллельные подсостояния могут, в свою очередь, состоять из нескольких последовательных подсостояний (рис. 9). В этом случае по определению моделируемый объект может находиться только в одном изпоследовательных подсостояний каждого подавтомата. Таким образом, для фрагмента диаграммы состояний (рис.9) допустимо одновременное нахождение объекта только в следующих подсостояниях: (А, В, Г), (Б, В, Г), (А, В, Д), (Б, В, Д).

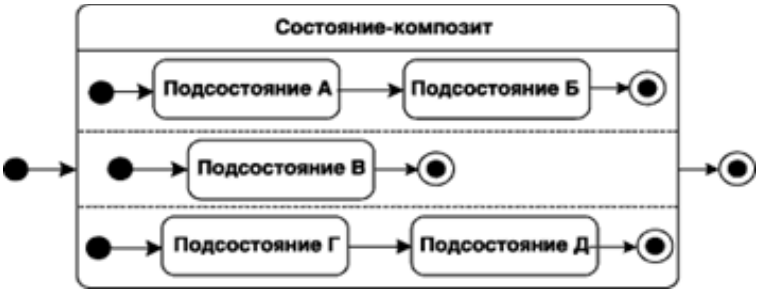

Рисунок 9 – Графическое изображение состояния-композита с вложенными параллельными подсостояниями

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).

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. 10).

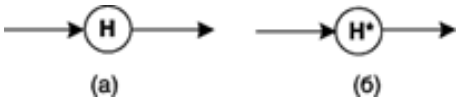

Рисунок 10 – Графическое изображение недавнего (а) и давнего (б) исторического состояния

Неглубокое историческое состояние (shallow history state) обозначается в форме небольшой окружности, в которую помещена латинская буква «H» (рис. 10, а). Это состояние обладает следующей семантикой. Во-первых, оно является первым подсостоянием в составном состоянии, и переход извне в рассматриваемое составноесостояние 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.

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. 10, 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.11, а), то его называют разделением (fork). Если же он имеет две или более входящие дуги (рис. 11, б), то его называют слиянием (join). Текстовая строка спецификации параллельного перехода записывается рядом с черточкой и относится ко всем входящим или исходящим дугам.

Рисунок 11 – Графическое изображение перехода-разделения в параллельные подсостояния (а) и перехода-слияния из параллельных подсостояний (б)

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. 12).

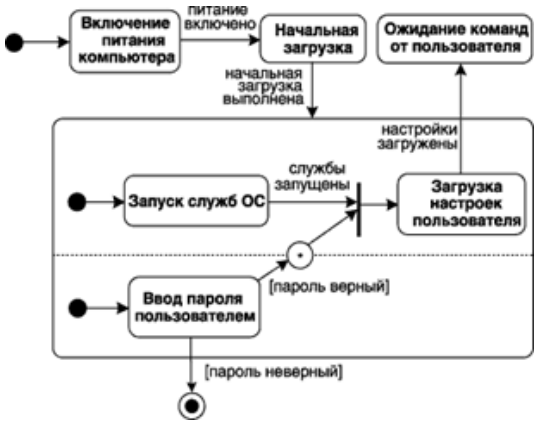

Рисунок 12 – Диаграмма состояний для примера включения компьютера

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 конкретномупредмету, а также отчет о загрузке аудиторий на каждый день.

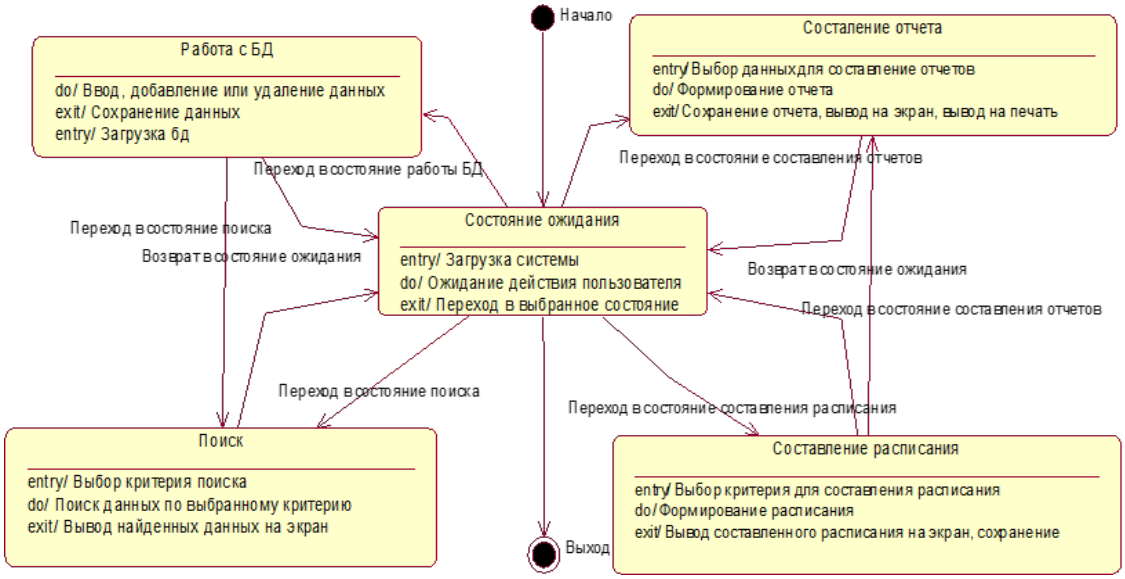

Рисунок 13 – Пример диаграммы состояний

Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

1. Запустите Visual Paradigm for UML CE.
2. Откройте проект, содержащий диаграмму вариантов использования, построенную в практическом занятии №2.
3. Для добавления новой диаграммы достаточно в меню Diagram > New Diagram выбрать тип диаграммы StateMachine Diagram и нажмите кнопку Next. Заданm имя диаграммы и нажать OK.
4. Либо, в ранее созданной диаграмме классов, щелкнуть правой кнопкой мыши по классу, для которого необходимо создать диаграмму состояний и в контекстном меню выбрать **Sub Diagrams > 1) New Diagram > StateMachine Diagram >** при необходимости изменить имя диаграммы и нажать OK.

2) Existing Diagram > выбрать в списке ранее созданную диаграмму и добавить ее в качестве поддиаграммы.

5. В результате отобразится рабочая область с элементами для построения диаграммы состояний (рис. 14).

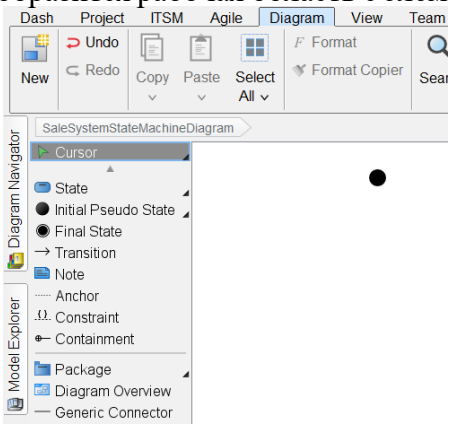

Рисунок 14 – Рабочая область для построения диаграммы состоянийДалее будут описаны принципы построения диаграммы состояний.

Для создания состояния необходимо на панели инструментов выбрать State и затем курсором мыши выбратьместо расположения на диаграмме (Рисунок 15). После создания класса необходимо задать его имя (Рисунок 16).

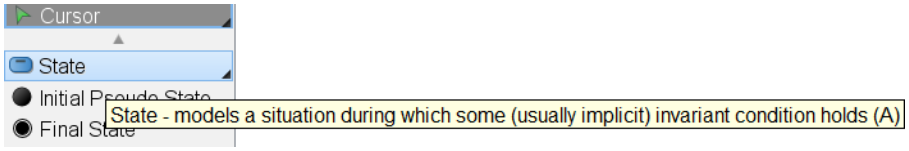

Рисунок 15 – Создание состояния

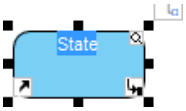

Рисунок 16 – Задание имени в созданном состоянии

Для создания перехода между на начальным состоянием и последующим состоянием или между состояниямивыберите **Transition** > **State (**Рисунок 17**).**

Перетащите курсор мыши на пустое место на диаграмме для создания нового или на уже существующее состояние для соединения. Отпустите кнопку мыши (Рисунок 17)**.**

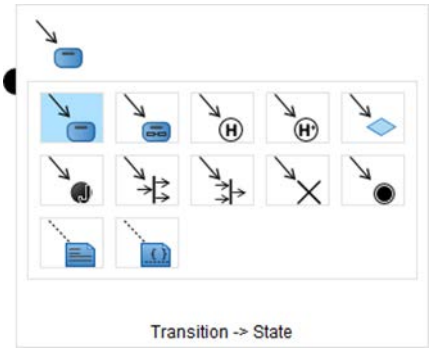

Рисунок 17 – Выбор типа перехода

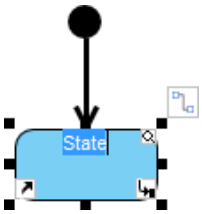

Рисунок 18 – Создание перехода

Для добавления внутренних действий, нажмите правой кнопкой мыши по состоянию и выберите пункт **OpenSpecification…** добавить действия можно посредством определения значений полей **Entry, Do, Exit Activity** (Рисунок 19). Либо выделите состояние и нажмите **Enter** (рисунок 19).

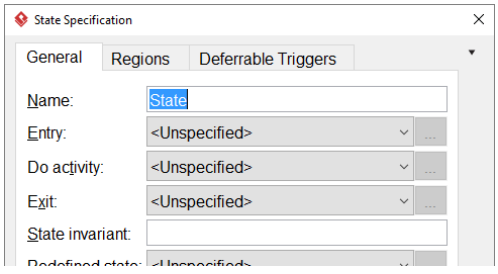

Рисунок 19 – Изменение внутренних действий

Например, для добавления внутреннего действия **Entry** щелкните раскрывающийся список и выберите **CreateActivity…** (рисунок 20).

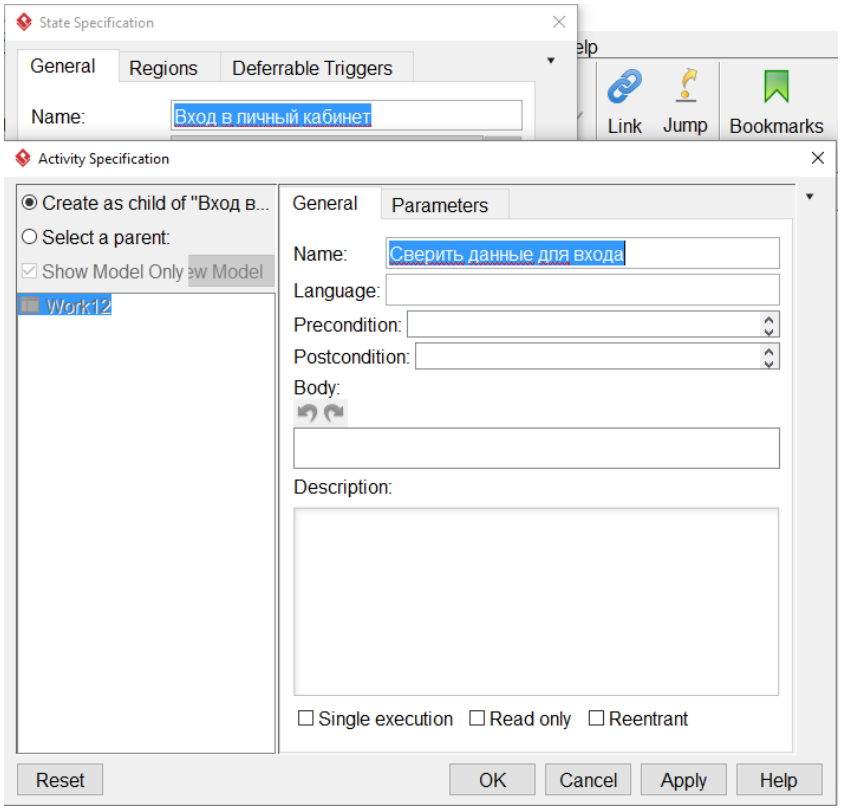

Рисунок 20 – Создание внутреннего действия

Для добавления описания перехода дважды щелкните по нему и введите описание (рисунок 21).

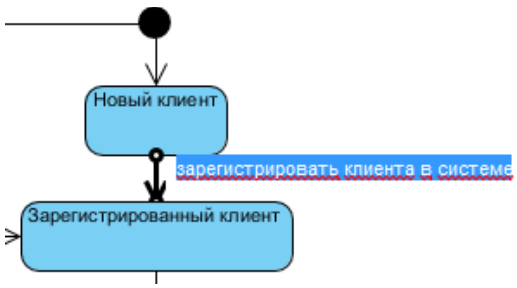

Рисунок 21 – Добавление описания перехода

Чтобы добавить переход разветвления или объединения наведите курсор в правый нижний угол элемента **Initial Pseudo State** и выберите **Fork** или **Join** (рисунок 22). После добавления элемента, для того чтобы изменитьориентацию на горизонтальную, щелкните правой кнопкой мыши по элементу и в контекстном меню выберите **Orientation > Horizontal.**

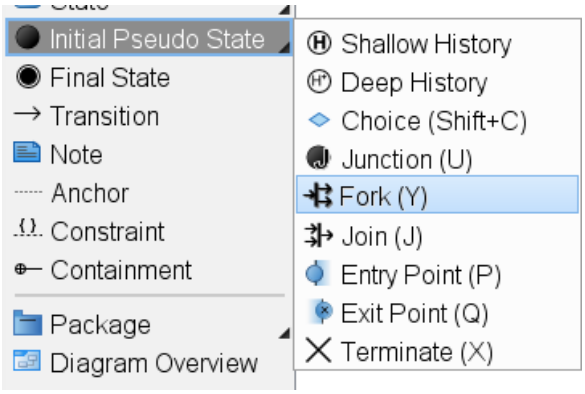

Рисунок 22 – Добавление перехода разветвления

Для того чтобы вернуться к диаграмме классов, нажмите на имя класса в иерархии диаграмм (рисунок 22).

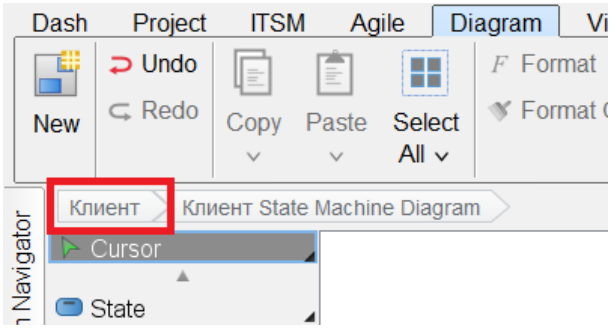

Рисунок 23 – Открытие диаграммы классов

Основываясь, на описанных выше принципах построения, постройте диаграммы состояний для системыпродажи товаров по каталогу в соответствии с рисунками 24-27.

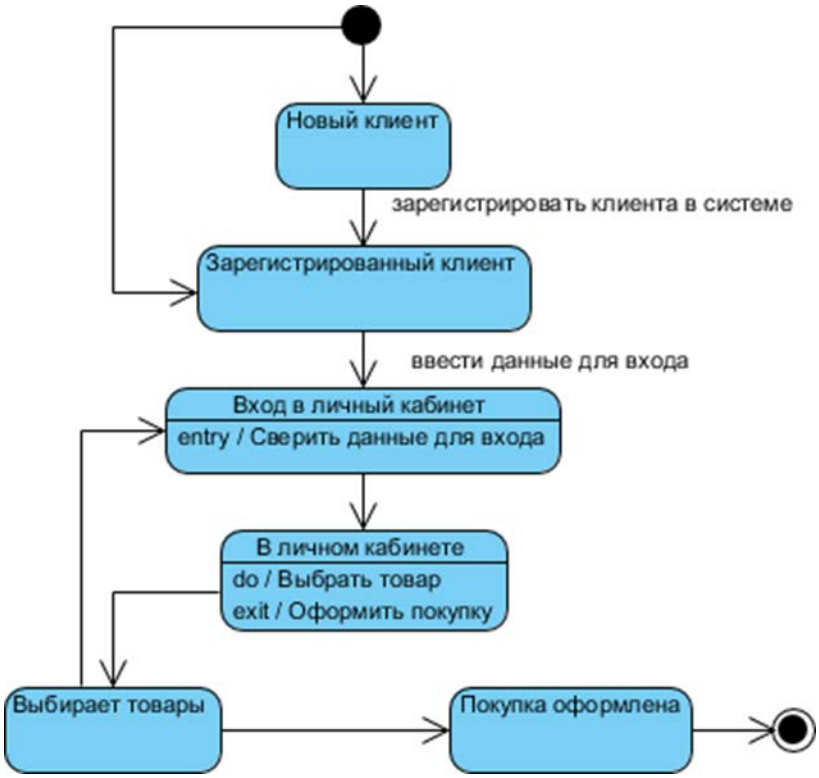

Рисунок 24 – Диаграмма состояний для класса «Клиент»

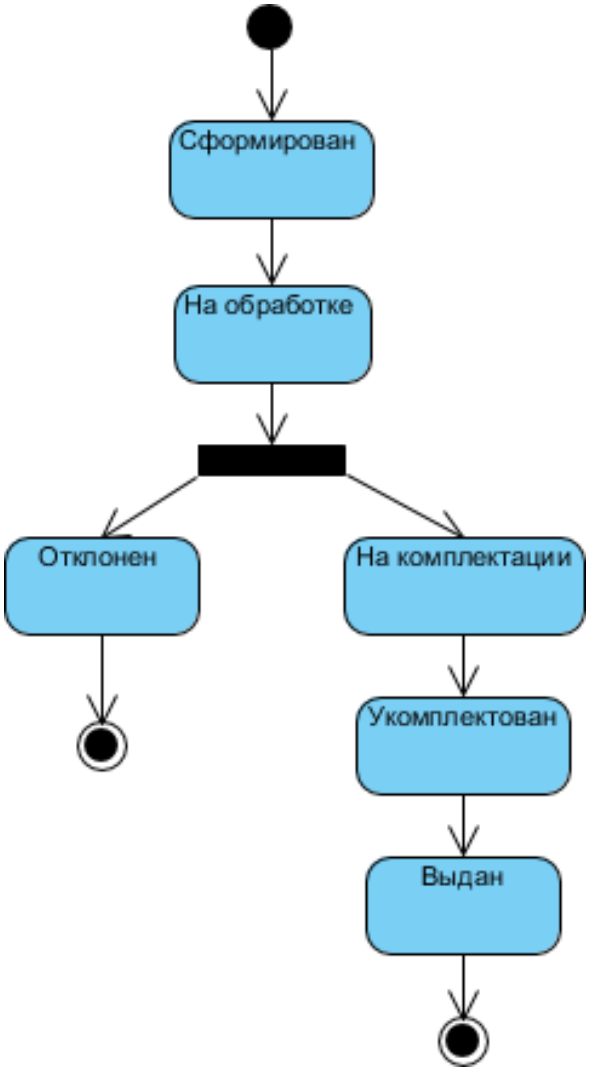

Рисунок 25 – Диаграмма состояний для класса «Заказ»

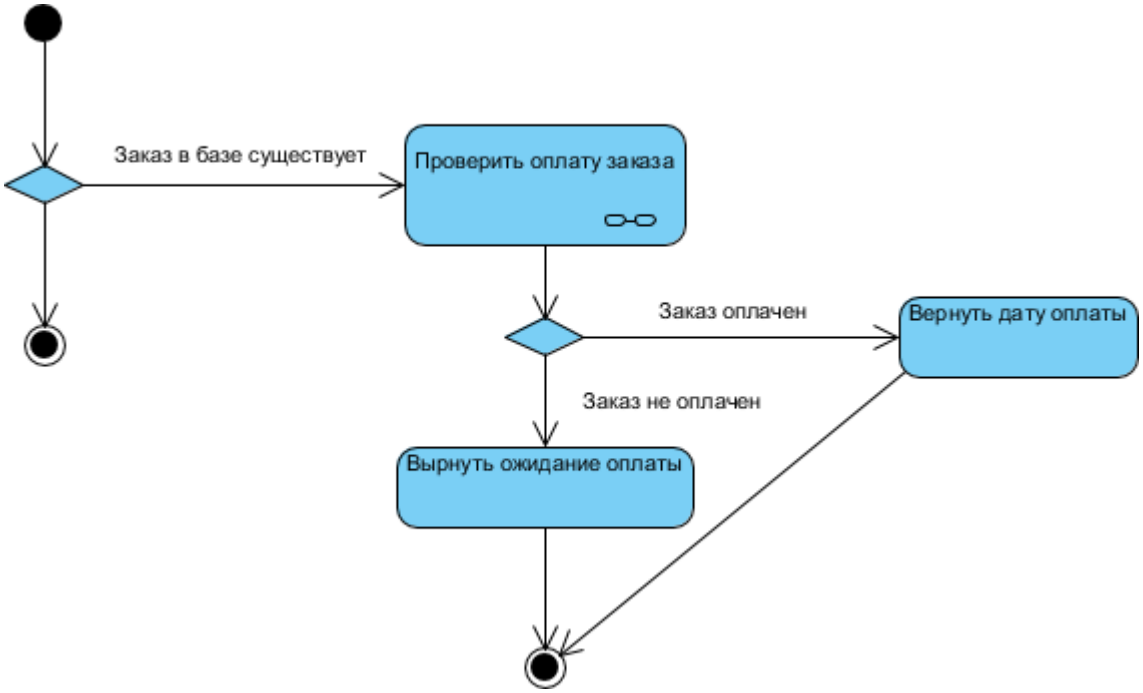

Рисунок 26 – Диаграмма состояний для класса «Заказ_Оплата»

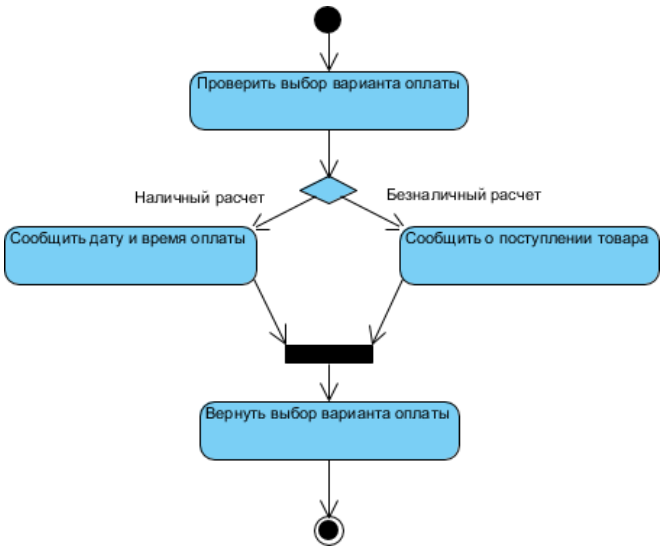

Рисунок 27 – Диаграмма состояний для класса «Вариант_Оплаты»

Задание

Построить диаграммы состояний для всех классов диаграммы классов, построенной в практическом занятии №2 в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описаниепроцесса построения диаграммы, а также скриншоты результатов согласно заданию.

Варианты

1. «Отдел кадров»;
2. «Агентство аренды»;
3. «Аптека»;
4. «Ателье»;
5. «Аэропорт»;
6. «Библиотека»;
7. «Кинотеатр»;
8. «Поликлиника»;
9. «Автосалон»;10.«Таксопарк».

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 изображениесостояния.
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», «параллельныйпереход», «состояние синхронизации».