Диаграмма состояний (State diagram)

Назначение диаграммы состояний

Каждая прикладная система характеризуется не только структурой, но и ***некоторым поведением или функциональностью***.

Для представления поведения на логическом уровне необходимо ответить на вопрос:

«В процессе какого поведения система обеспечивает необходимую функциональность?»

Главное предназначение диаграммы состояний – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

Назначение диаграммы состояний

Диаграмма состояний – граф специального вида, который представляет некоторый *автомат*.

Вершинами графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами.

Дуги графа служат для обозначения переходов из состояния в состояние.

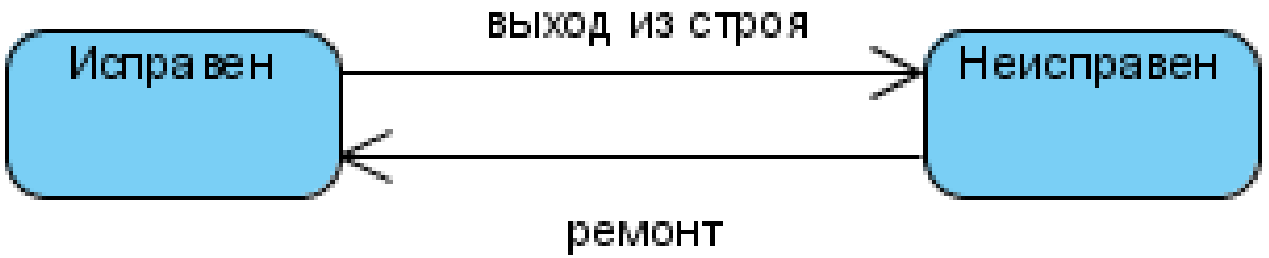
Диаграммы состояний могут быть вложены друг в друга.

Простейшим примером визуального представления состояний и переходов на основе автоматов может служить ситуация с исправностью технического устройства.

В этом случае имеется два самых общих состояния: «исправен» и «неисправен» и два перехода: «выход из строя» и «ремонт».

Автоматы

Графически такая информация может быть представлена в виде диаграммы состояний технического устройства:



Основными понятиями в автомате являются *состояние* и *переход*.

Главное различие между ними: длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое.

Автоматы

Для графа состояний системы вводят в рассмотрение специальные свойства.

Одно из таких свойств – выделение из всей совокупности состояний двух специальных: *начального* и *конечного*.

Хотя на диаграмме состояний время нахождения системы в том или ином состоянии явно не учитывается, предполагается, что последовательность изменения состояний упорядочена во времени.

Каждое последующее состояние всегда наступает позже предшествующего ему состояния.

Автоматы

Следующее свойство графа состояний – *достижимость* состояний.

Это свойство характеризует потенциальную возможность перехода системы из рассматриваемого состояния в некоторое другое состояние.

Описание автоматов допускает вложение одних автоматов в другие для уточнения внутренней структуры отдельных более общих состояний (макросостояний).

В этом случае вложенные автоматы получили название *подавтоматов*.

Автоматы

Описание обычного автомата основано на выполнении следующих условий:

1. Автомат не запоминает историю перемещения из состояния в состояние. Автомат «забывает» все состояния, которые предшествовали текущему в данный момент времени.
2. В каждый момент времени автомат может находиться в одном и только в одном из своих состояний (предназначен для моделирования последовательного поведения). ***Автомат может находиться в отдельном состоянии как угодно долго.***
3. Хотя процесс изменения состояний автомата происходит во времени, явно концепция времени не входит в формализм автомата. *Длительность нахождения автомата в том или ином состоянии, а также время достижения того или иного состояния не специфицируются.*

Автоматы

5. Количество состояний автомата должно быть обязательно конечным. При этом отдельные псевдосостояния могут не иметь спецификаций (начальное и конечное состояния).
6. Граф автомата не должен содержать изолированных состояний и переходов. *Для каждого состояния, кроме начального, должно быть определено предшествующее состояние. Каждый переход должен обязательно соединять два состояния автомата. Допускается переход из состояния в себя – «петля».*
7. Автомат не должен содержать конфликтующих переходов, т.е. таких переходов из одного и того же состояния, когда объект одновременно может перейти в два и более последующих состояния (кроме случая параллельных подавтоматов).

Состояние

Состояние (state) на диаграмме изображается прямоугольником со скругленными вершинами:

Этот прямоугольник может быть разделен на две секции горизонтальной линией. Во второй секции указывается список внутренних действий или переходов в данном состоянии.

Действие – атомарная операция, выполнение которой приводит к изменению состояния или возврату некоторого значения

Имя состояния

Имя состояния – строка текста, которая раскрывает содержательный смысл данного состояния.

Имя всегда записывается с заглавной буквы.

В качестве имени рекомендуется использовать глаголы в настоящем времени (звенит, печатает, ожидает) или соответствующие причастия (занят, свободен, передано, получено).

Имя у состояния может отсутствовать – в этом случае состояние является анонимным.

Список внутренних действий

Эта секция содержит перечень внутренних действий или деятельностей, которые выполняются в процессе нахождения моделируемого элемента в данном состоянии.

Каждое из действий записывается в виде отдельной строки следующего формата:

<*метка-действия ‘/’ выражение-действия*>

Метка действия указывает на условия, при которых будет выполняться деятельность. При этом выражение действия может использовать любые атрибуты и связи, которые принадлежат области имен или контексту моделируемого объекта.

Список внутренних действий

Перечень меток действия имеет фиксированные значения в UML:

• *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);

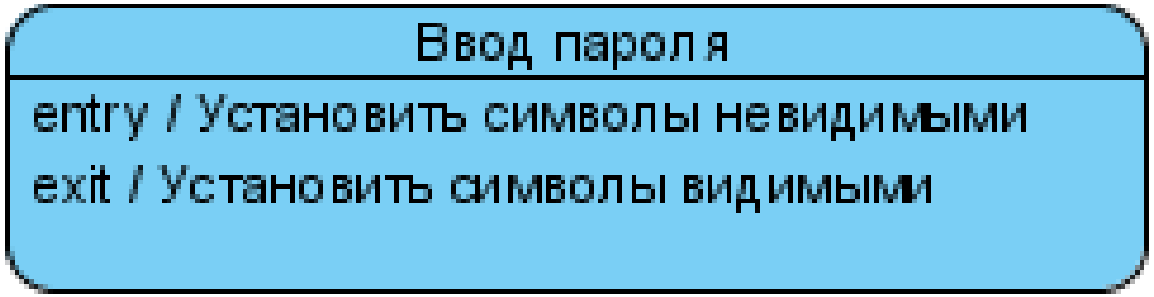
• *do* – эта метка специфицирует выполняющуюся деятельность (do activity), которая выполняется в течении всего времени, пока объект находится в данном состоянии.

Во всех остальных случаях метка действия идентифицирует событие, которое запускает соответствующее выражение действия.

Такие события называются *внутренними переходами*.

Список внутренних действий

Пример: ситуация ввода пароля пользователя при аутентификации входа в некоторую программную систему:

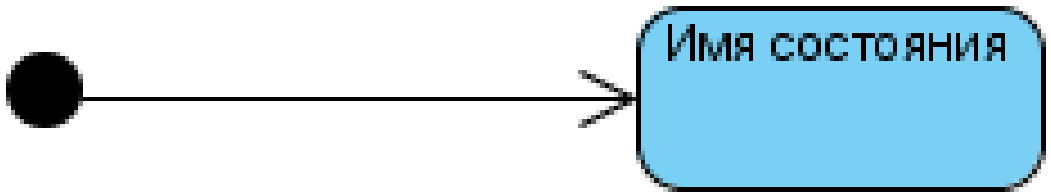


Начальное состояние

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояния).

В этом состоянии находится объект по умолчанию в начальный момент времени.

Графически начальное состояние в UML обозначается в виде закрашенного кружка, из которого может только выходить стрелка, соответствующая переходу:

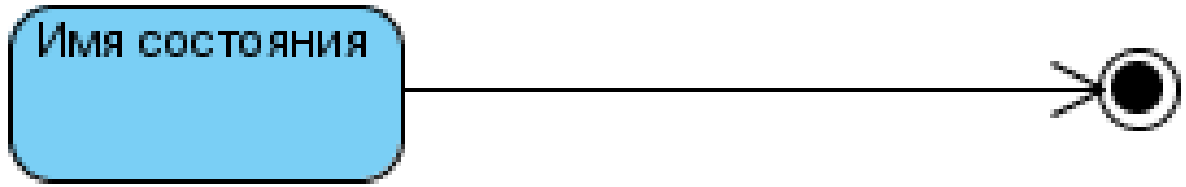


Конечное состояние

Конечное (финальное) состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояния).

В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени.

Графически конечное состояние в UML обозначается в виде закрашенного кружка, помещенного в окружность, в которую может только входить стрелка, соответствующая переходу:



Переход

Простой переход (simple transition) – отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим.

Пребывание моделируемого объекта в первом состоянии может сопровождаться выполнением некоторых действий, а переход во второе состояние будет возможен после завершения этих действий, а также после удовлетворения некоторых дополнительных условий.

В этом случае говорят, что переход срабатывает, или происходит срабатывание перехода.

Переход

На переходе указывается имя события.

Срабатывание перехода может зависеть не только от наступления некоторого события, но и от выполнения определенного условия, называемого *сторожевым условием*.

Объект переходит из одного состояния в другое в том случае, если произошло указанное событие и сторожевое условие приняло значение «истина».

Переход

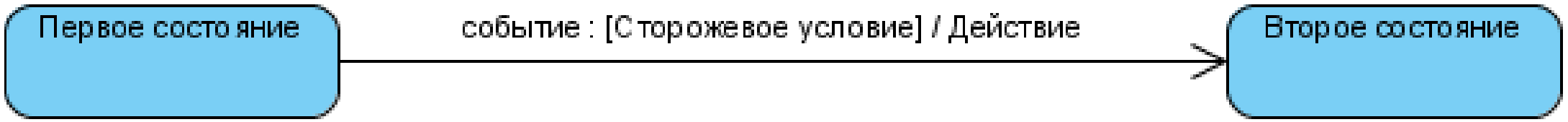
На диаграмме состояний переход изображается сплошной линией со стрелкой, которая направлена в целевое состояние.

Каждый переход может помечаться строкой текста следующего формата:

<*сигнатура события*> ‘[’<*сторожевое условие*>’]’ <*выражение действия*>

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

<*имя события*>’(’<список параметров, разделенных запятыми>’)’



Событие

В языке UML события играют роль стимулов, которые инициируют переходы из одних состояний в другие.

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*, т.е. таким, который специфицирует событие триггер.

Если строка текста рядом со стрелкой не указана, то соответствующий переход называется *нетриггерным*.

В этом случае из контекста диаграммы состояний должно быть ясно, после окончания какой деятельности он срабатывает.

Сторожевое условие

Сторожевое условие (guard condition) всегда записывается в прямых скобках после события-триггера и представляет собой некоторое булевское выражение.

Если сторожевое условие принимает значение «истина», то соответствующий переход может сработать, в результате чего объект перейдет в целевое состояние.

Если же сторожевое условие принимает значение «ложь», то переход не может сработать, и при отсутствии других переходов объект не может перейти в целевое состояние.

Вычисление истинности сторожевого условия происходит только после возникновения ассоциированного с ним события-триггера, инициирующего соответствующий переход.

Сторожевое условие

Графически фрагмент логики моделирования почтовой программы представлен в виде следующей диаграммы состояний:

Выражение действия

Выражение действия (action expression) выполняется в том и только в том случае, когда переход срабатывает.

Представляет собой атомарную операцию, выполняемую сразу после срабатывания соответствующего перехода до начала действий в целевом состоянии.

Атомарность действия означает, что оно не может быть прервано никаким другим действием до тех пор, пока не закончится его выполнение.

Составное состояние и подсостояние

Составное состояние (composite state) – такое сложное состояние, которое состоит из других вложенных в него состояний.

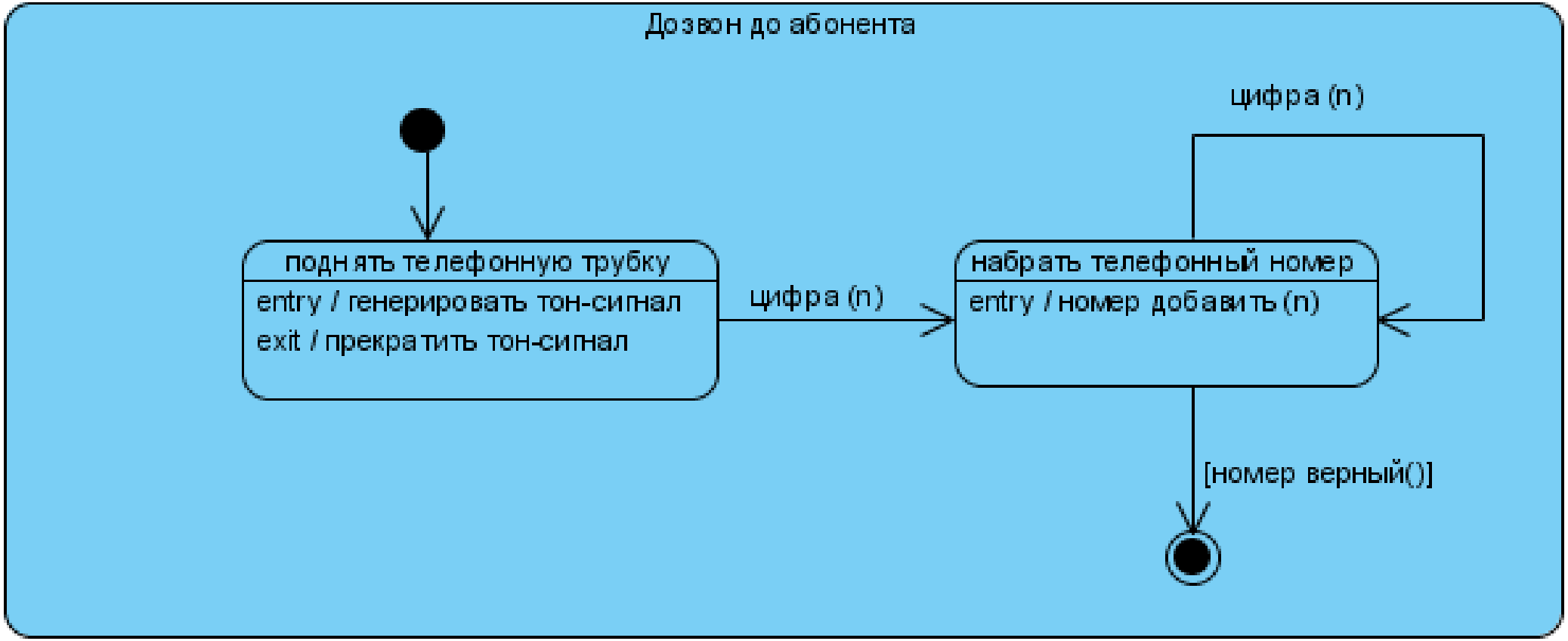
Последние будут выступать по отношению к первому как *подсостояния* (substate).

Составное состояние может содержать два или более параллельных подавтомата или несколько последовательных подсостояний.

Количество уровней вложенности составных состояний не фиксировано в языке UML.

Составное состояние и подсостояние

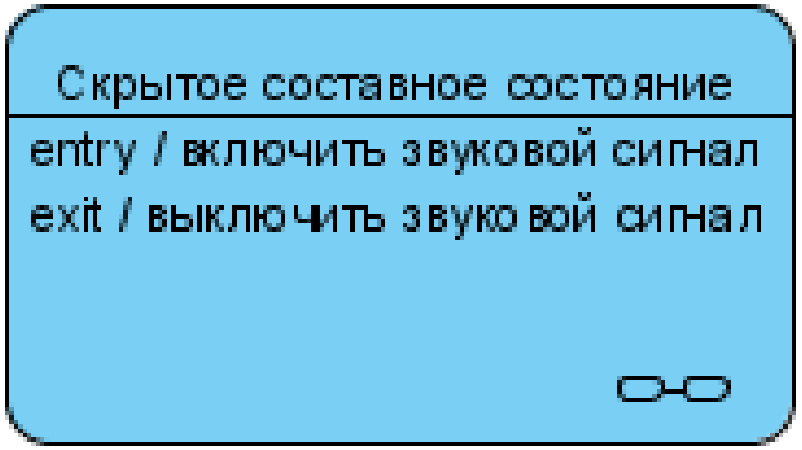
Пример: состояние дозвона до абонента как одно из возможных состояний телефонного аппарата:



Составное состояние и подсостояние

В некоторых случаях бывает желательно скрыть внутреннюю структуру составного состояния.

В подобной ситуации допускается не раскрывать на исходной диаграмме состояний данное составное состояние, а указать в правом нижнем углу специальный символ-пиктограмму:



Историческое состояние

Формализм обычного автомата не позволяет учитывать предысторию в процессе моделирования поведения объектов.

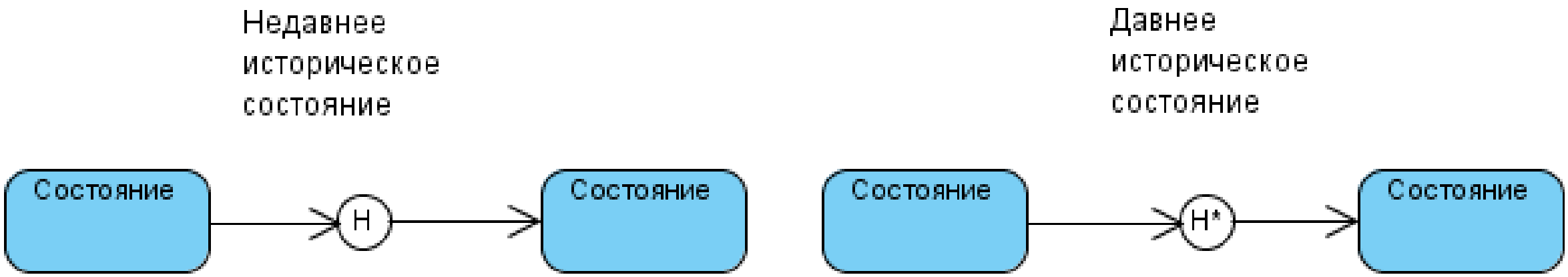
Для того, чтобы можно было учитывать ту часть деятельности, которая была выполнена на момент выхода из некоторого состояния и не начинать все сначала, в языке UML существует историческое состояние.

Историческое состояние (history state) применяется в контексте составного состояния.

Оно используется для для запоминания того из последовательных подсостояний, которое было текущим в момент выхода из составного состояния

Историческое состояние

Существует две разновидности исторического состояния: недавнее и давнее:



Историческое состояние

Недавнее историческое состояние (shallow history state) обладает следующей семантикой:

• оно является первым подсостоянием в составном состоянии, и переход извне в это составное состояние должен вести непосредственно в это историческое состояние;

• при первом попадании в недавнее историческое состояние оно не хранит никакой истории.

Если в некоторый момент времени происходит выход из вложенного состояния (например, внешнее событие), то это историческое состояние запоминает то из подсостояний, которое являлось текущим на момент выхода.

Историческое состояние

При следующем входе в это же составное состояние историческое подсостояние 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.

Давнее историческое состояние (deep history state) служит для запоминания всех подсостояний любого уровня вложенности для текущего подавтомата.

Сложные переходы

Современные программные системы могут реализовывать очень сложную логику поведения отдельных своих компонентов.

Для этих целей в языке UML имеются дополнительные обозначения и свойства, которыми могут обладать отдельные переходы на диаграмме состояний.

Переходы между параллельными состояниями

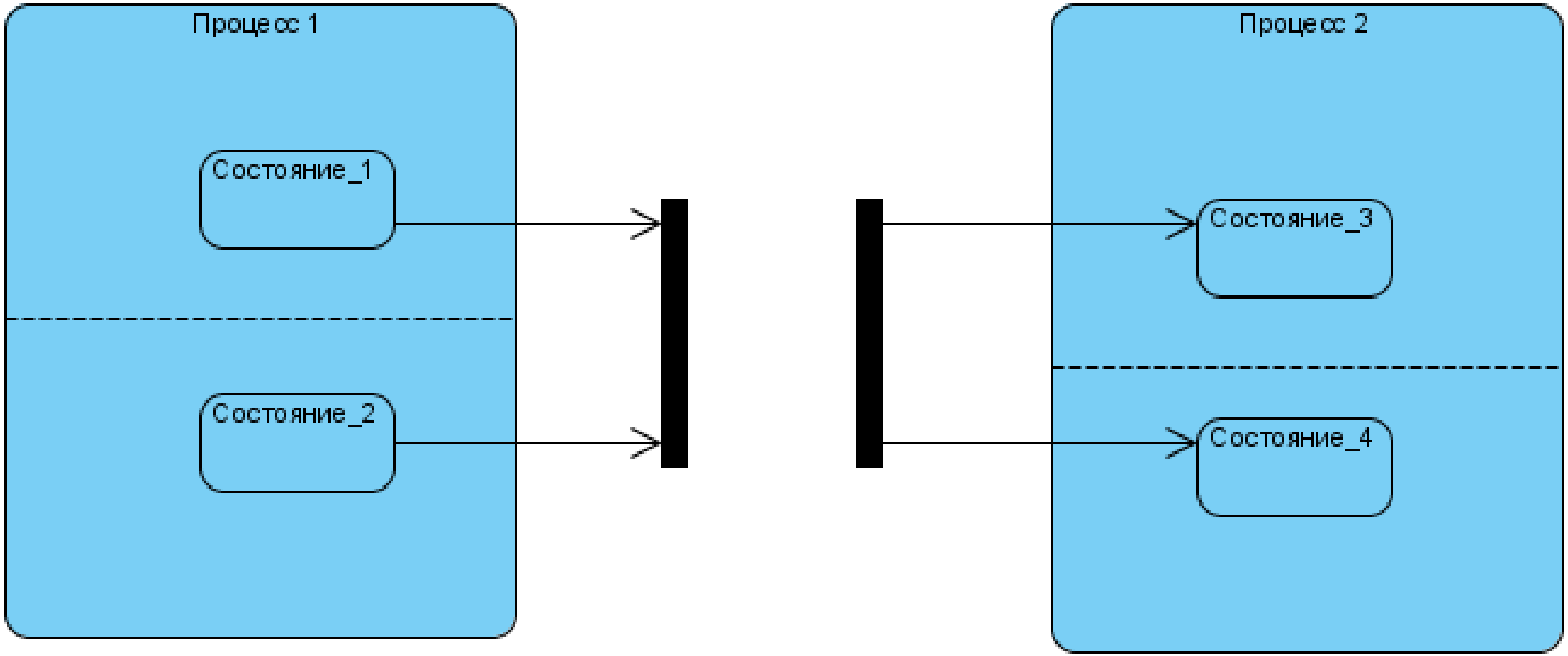
Переход может иметь несколько состояний-источников и несколько целевых состояний. Такой переход получил специальное название – *параллельный* переход.

Если параллельный переход имеет две или более входящих дуг, то его называют *соединением* (join).

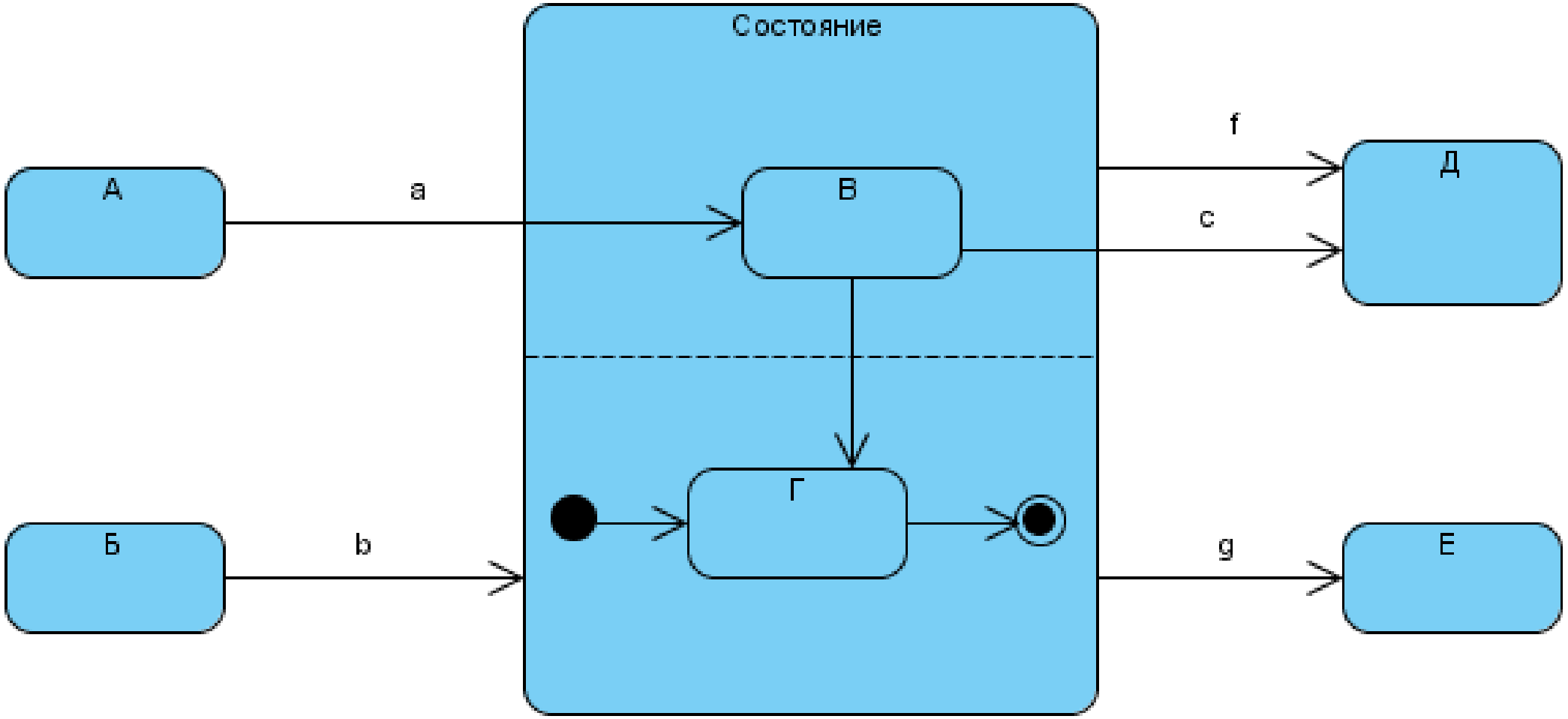
Если он имеет две или более исходящих из него дуг, то его называют *ветвлением* (fork).

Переходы между параллельными состояниями

Пример параллельного перехода из параллельных состояний и параллельного перехода в параллельное состояние:



Переходы между составными состояниями



Пример: диаграмма состояний процесса функционирования телефонного аппарата

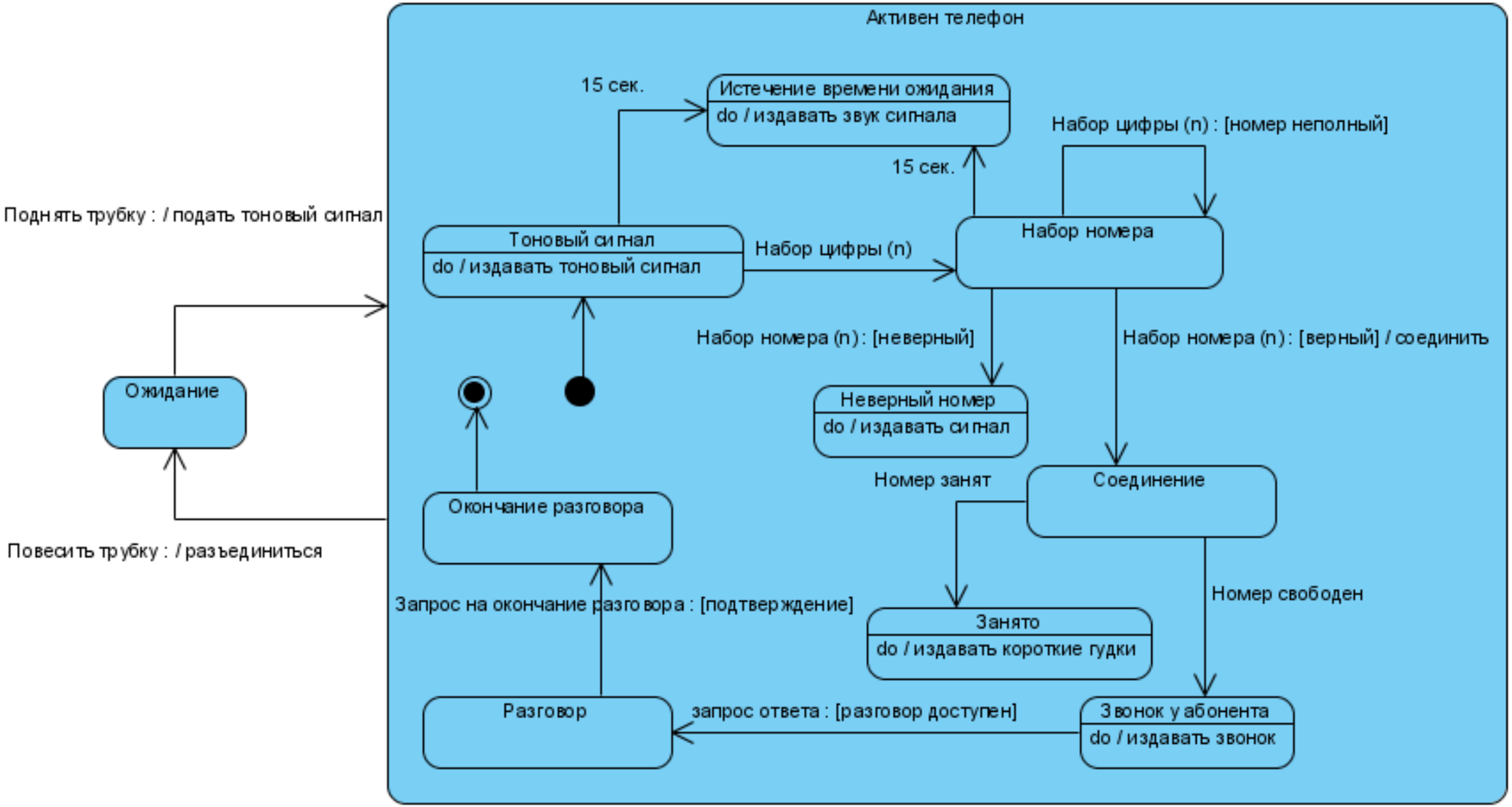


Диаграмма деятельности (Activity diagram)

Назначение диаграммы деятельности

Для моделирования процесса выполнения операций в языке UML используются *диаграммы деятельности*.

Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии.

Диаграммы деятельности можно считать частным случаем диаграмм состояний.

В языке UML *деятельность* (activity) – совокупность отдельных вычислений, выполняемых автоматом.

Отдельные элементарные вычисления могут приводить к некоторому результату или действию (action).

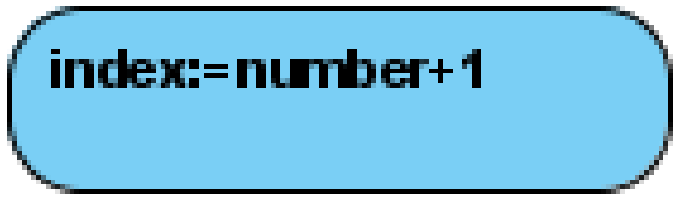
На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой.

Состояние действия

Состояние действия (action state) – специальный случай состояния с некоторым входным действием и по крайней мере одним выходящим из состояния переходом.

Использование состояния действия заключается в моделировании одного шага выполнения алгоритма или потока управления.

Графически состояние действия изображается так:



Состояние действия

Действие может быть записано на естественном языке, некотором псевдокоде или языке программирования.

Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояния.

Диаграмму деятельности принято располагать так, чтобы действия следовали сверху вниз.

Переходы

При построении диаграммы деятельности используются только *нетриггерные* переходы, т.е. такие, которые срабатывают сразу после завершения деятельности или выполнения соответствующего действия.

Этот переход переводит деятельность в последующее состояние сразу, как только закончится действие в предыдущем состоянии.

На диаграмме такой переход изображается сплошной линией со стрелкой.

Если из состояния действия выходит единственный переход, то он может не помечаться.

Если же таких переходов несколько, то сработать может только один из них – для каждого из таких переходов должно быть явно записано сторожевое условие в прямых скобках.

Такая ситуация получила название ветвления, а для ее обозначения применяется специальный символ.

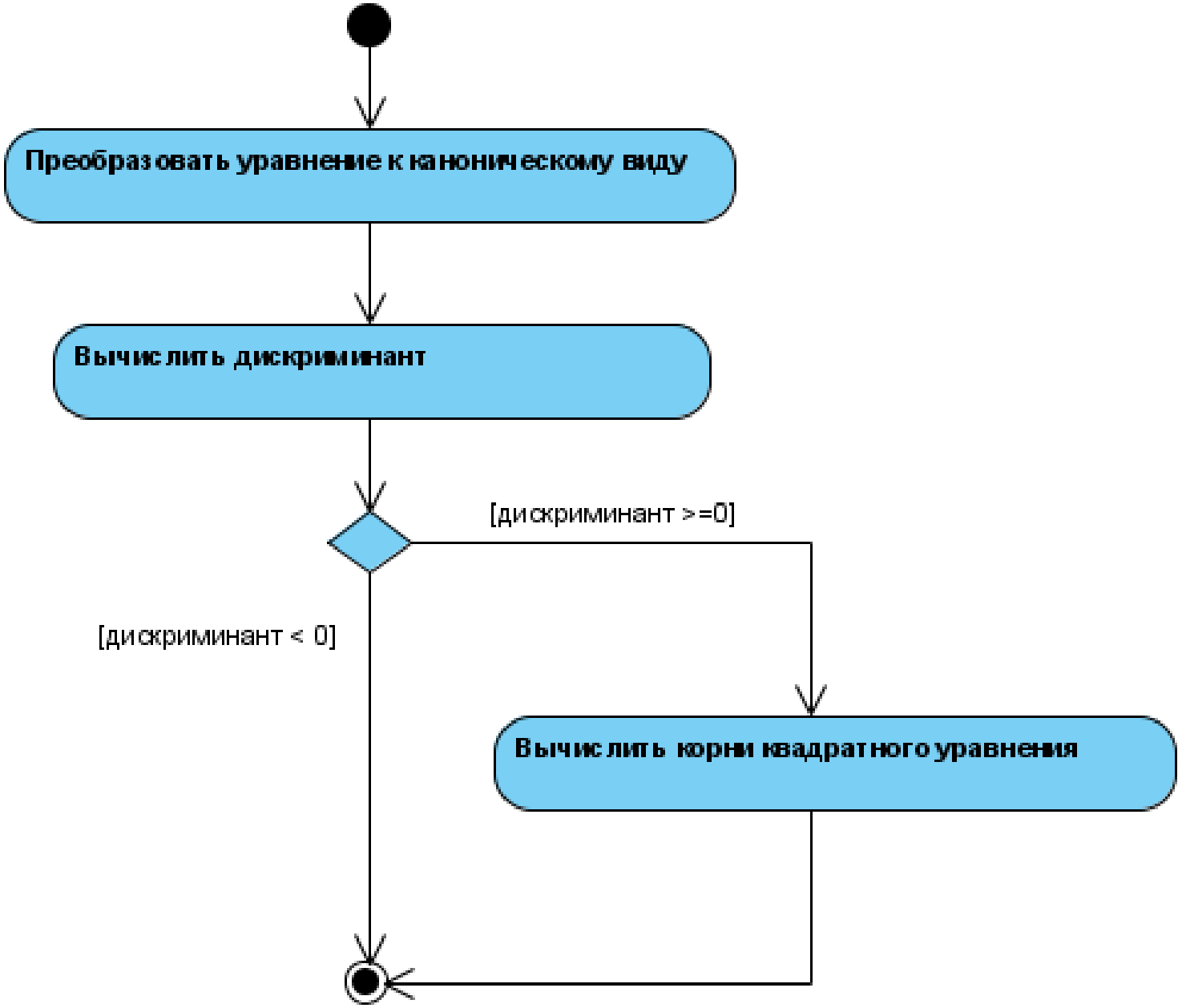
Переходы

Графически ветвление на диаграмме деятельности обозначается небольшим ромбом, внутри которого нет никакого текста.

В этот ромб может входить только одна стрелка от того состояния действия, после выполнения которого поток управления должен быть продолжен по одной из взаимно исключающих ветвей.

Принято входящую стрелку присоединять к верхней или левой вершине символа ветвления.

Пример ветвления на диаграмме деятельности:



Переходы

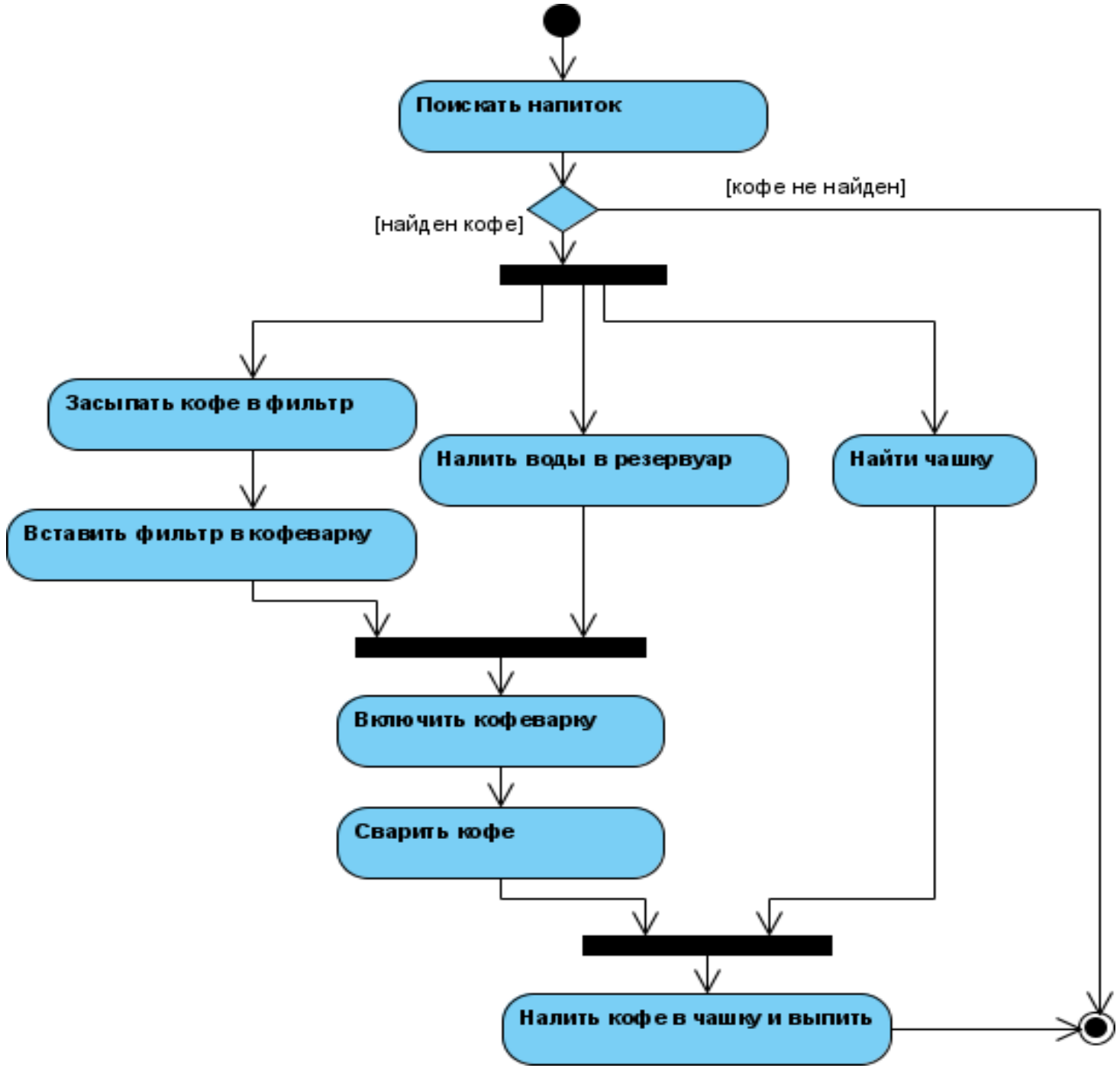
Один из недостатков обычных блок-схем или структурных схем алгоритмов связан с проблемой изображения параллельных ветвей отдельных вычислений.

В языке UML для этой цели используется специальный символ для разделения и слияния параллельных вычислений или потоков управления.

Символ *разделения* (concurrent fork) имеет один входящий переход и несколько выходящих.

Слияние (concurrent join) имеет несколько входящих переходов и один выходящий.

Пример параллельных процессов на диаграмме деятельности:



Дорожки

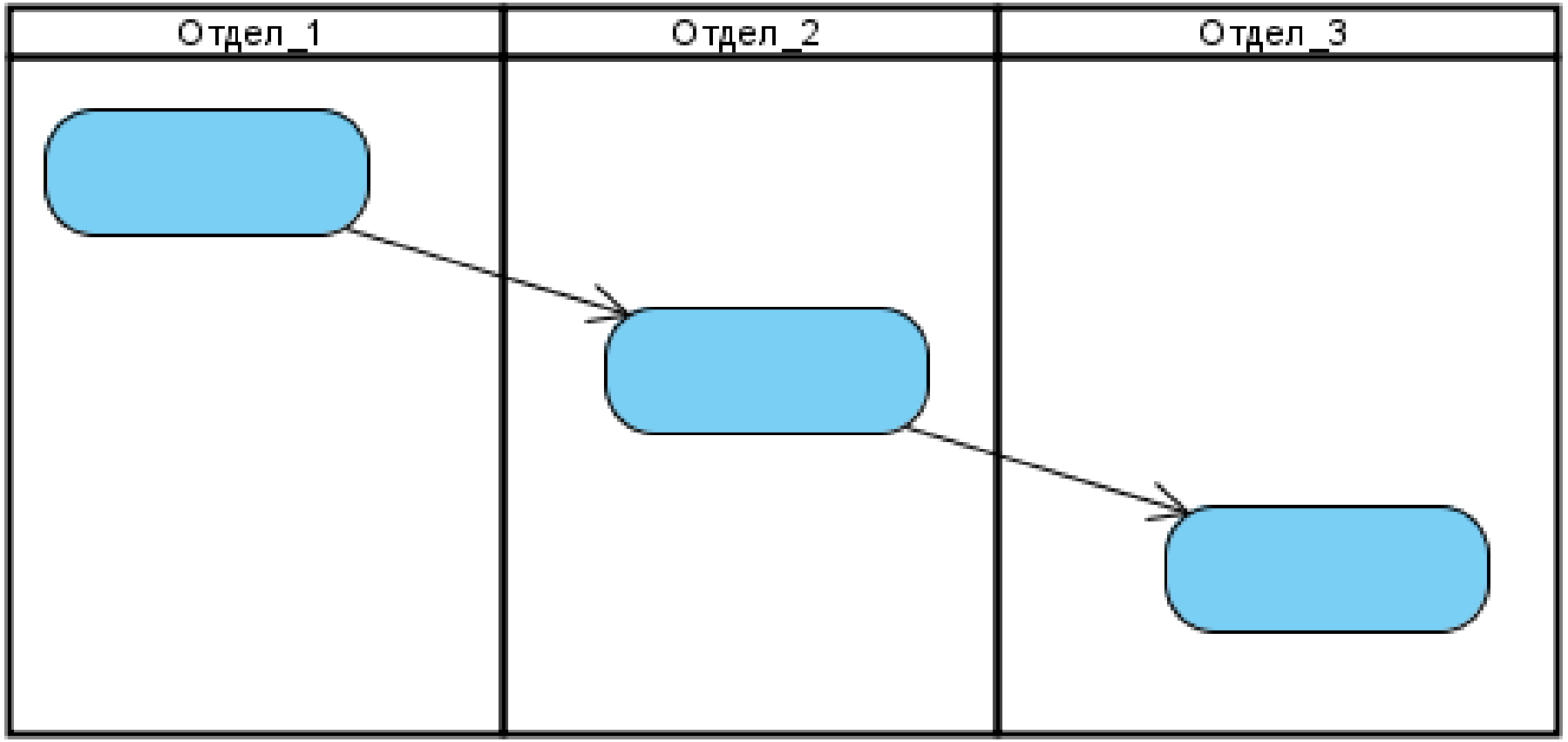
Диаграммы деятельности могут быть использованы не только для спецификации алгоритмов вычисления или потоков управления, но и для моделирования бизнес-процессов.

В этом случае действия желательно ассоциировать с конкретным подразделением компании.

В этом случае подразделение несет ответственность за реализацию отдельных действий.

Дорожки

Для моделирования указанных особенностей в языке UML используется специальная конструкция – *дорожки* (swimlanes):



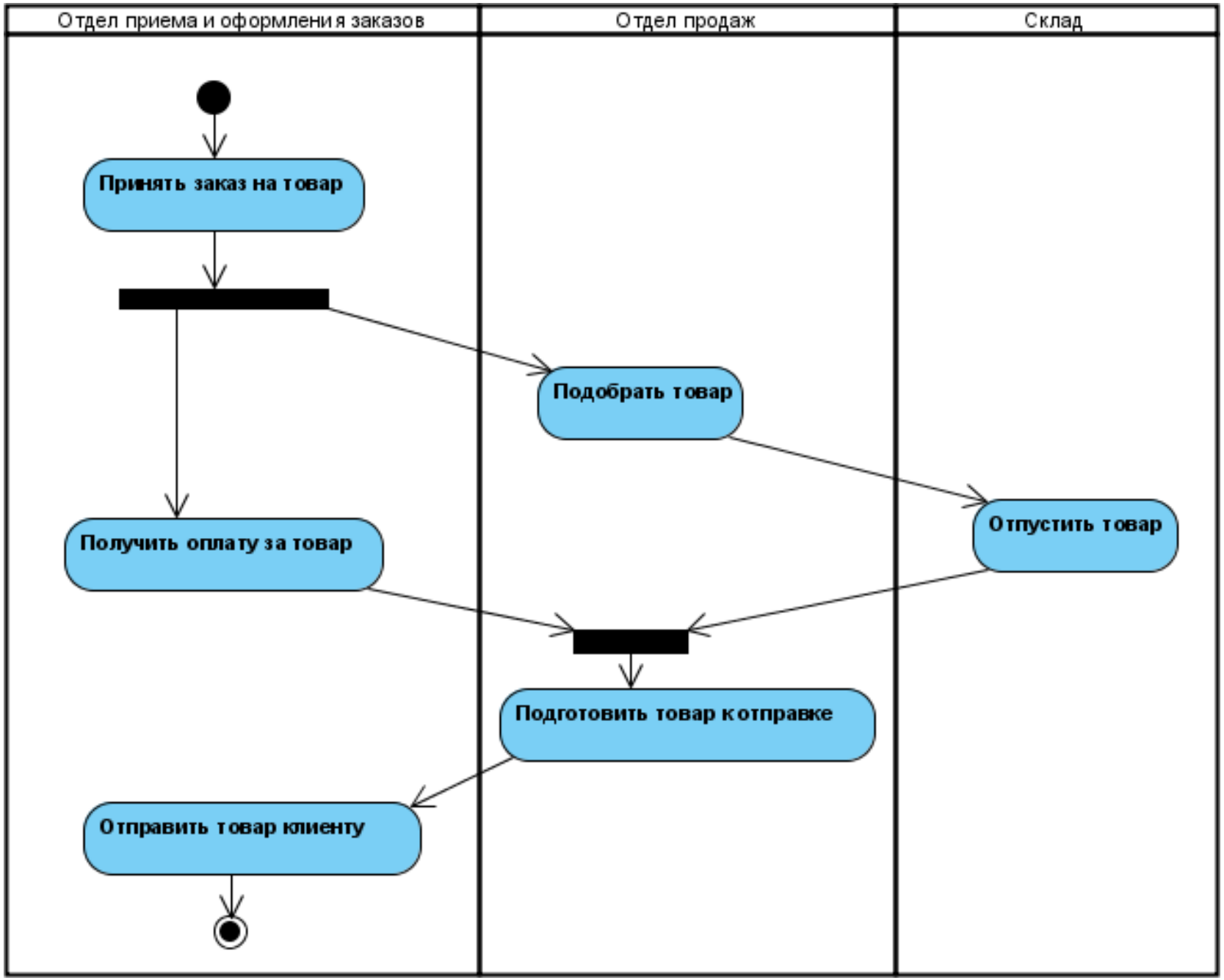
Дорожки

Название подразделений явно указываются в верхней части дорожки.

Пересекать линию дорожки могут только переходы.

Порядок следования дорожек не несет какой-либо семантической информации и определяется соображениями удобства.

Пример фрагмента диаграммы деятельности торговой компании:



Объекты

Действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий.

Для графического представления объектов используется прямоугольник класса.

После имени может указываться характеристика состояния объекта в прямых скобках.

Объекты присоединяются к состояниям действия отношением зависимости пунктирной линией со стрелкой.

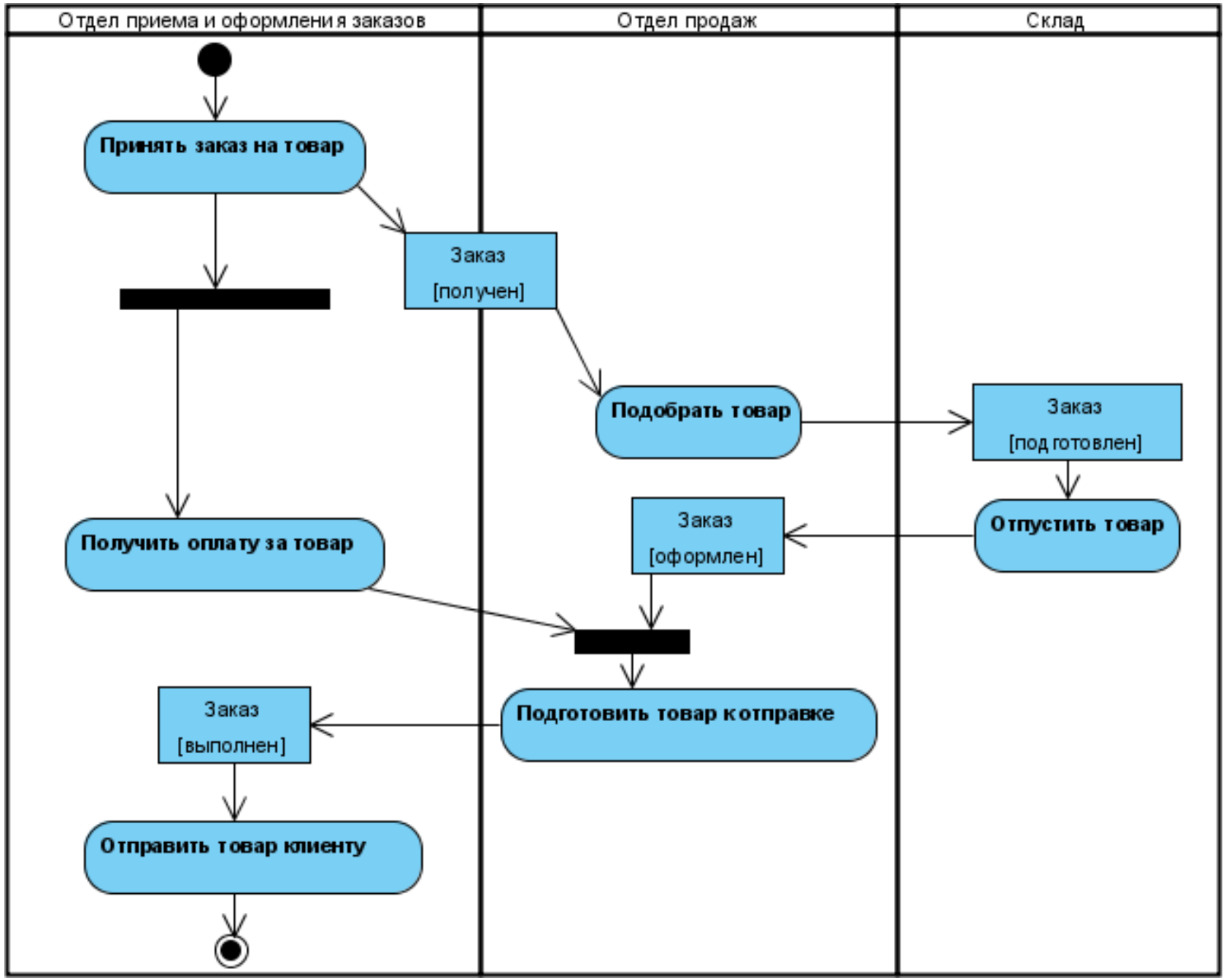
Объекты

На диаграмме деятельности с дорожками расположение объекта может иметь дополнительный смысл.

Если объект расположен на границе двух дорожек, то это может означать, что переход к следующему состоянию действия в соседней дорожке связан с готовностью некоторого документа.

Если объект целиком расположен внутри дорожки, то и состояние этого объекта целиком определяется действиями данной дорожки.

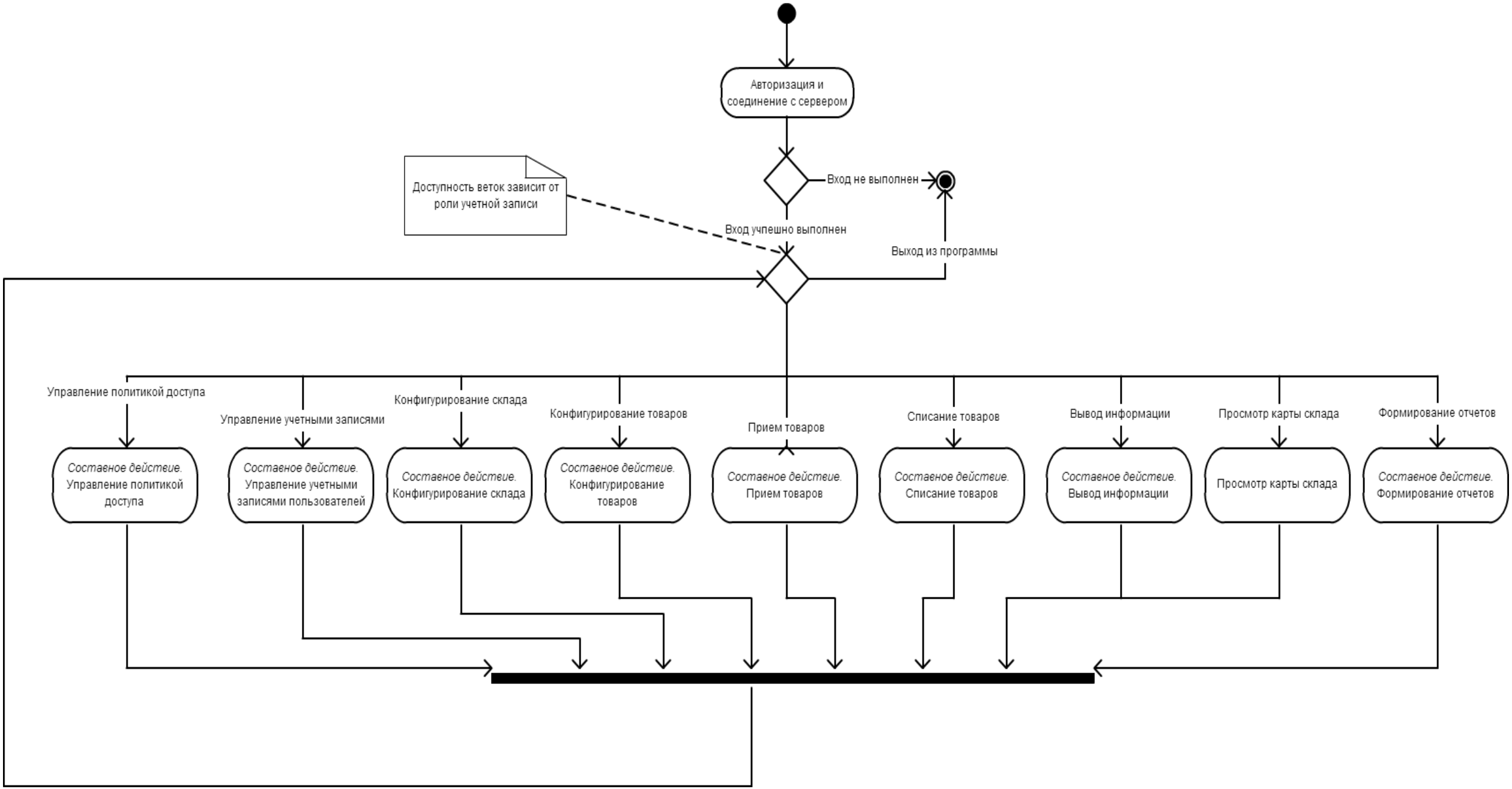
Модифицированный пример с торговой компанией:

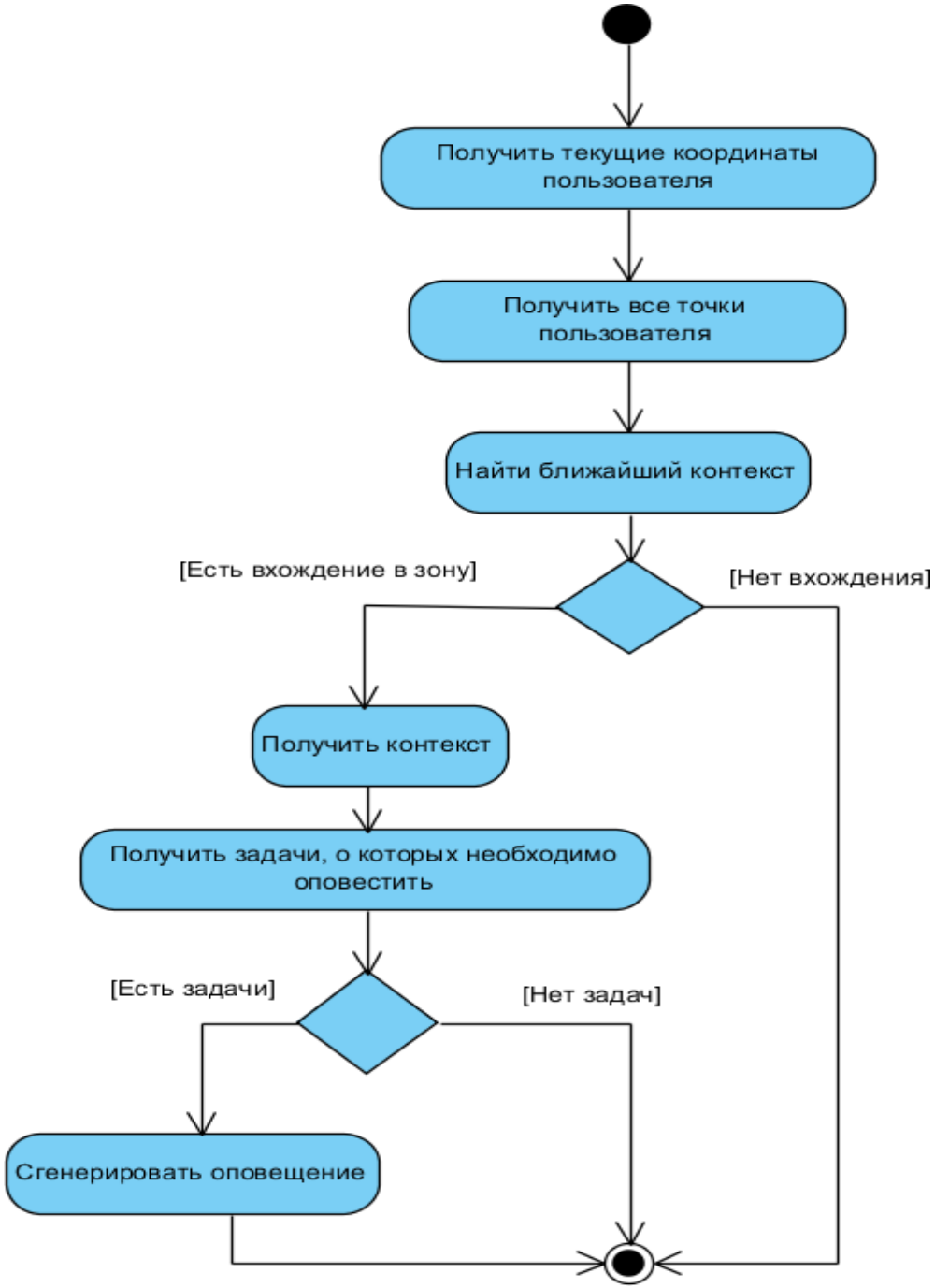


Рекомендации по построению диаграмм деятельности

Диаграмма деятельности строится для отдельного класса, варианта использования, отдельной операции класса или целой подсистемы.

На начальных этапах проектирования построение диаграммы деятельности начинают с выделения под-деятельностей, которые в совокупности образуют деятельность подсистем.


Авторизация и соединение с сервером
Доступность веток зависит от роли учетной записи
Вход не выполнен
Вход учпешно выполнен
Выход из программы
Управление политикой доступа
Управление учетными записями
Конфигурирование склада
Конфигурирование товаров
Прием товаров
Списание товаров
Вывод информации
Просмотр карты склада
Формирование отчетов
Составное действие. Управление политикой доступа
Составное действие. Управление учетными записями пользователей
Составное действие. Конфигурирование склада
Составное действие. Конфигурирование товаров
Составное действие. Прием товаров
Составное действие. Списание товаров
Составное действие. Вывод информации
Просмотр карты склада
Составное действие. Формирование отчетов


Получить текущие координаты пользователя
Получить все точки пользователя
Найти ближайший контекст
[Есть вхождение в зону]
[Нет вхождения]
Получить контекст
Получить задачи, о которых необходимо оповестить
[Есть задачи]
[Нет задач]
Сгенерировать оповещение