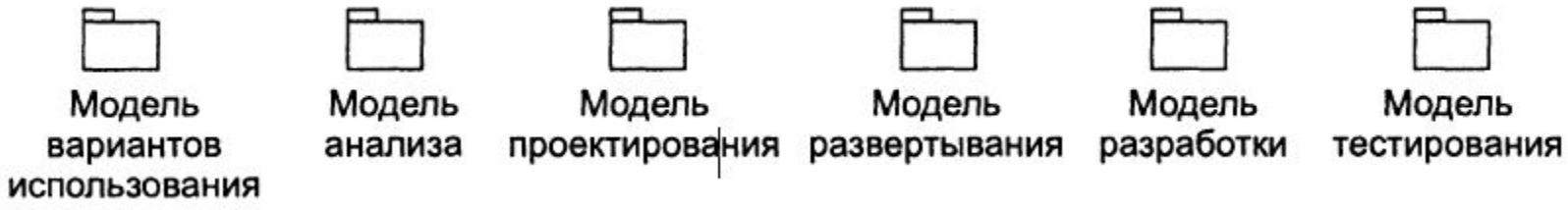
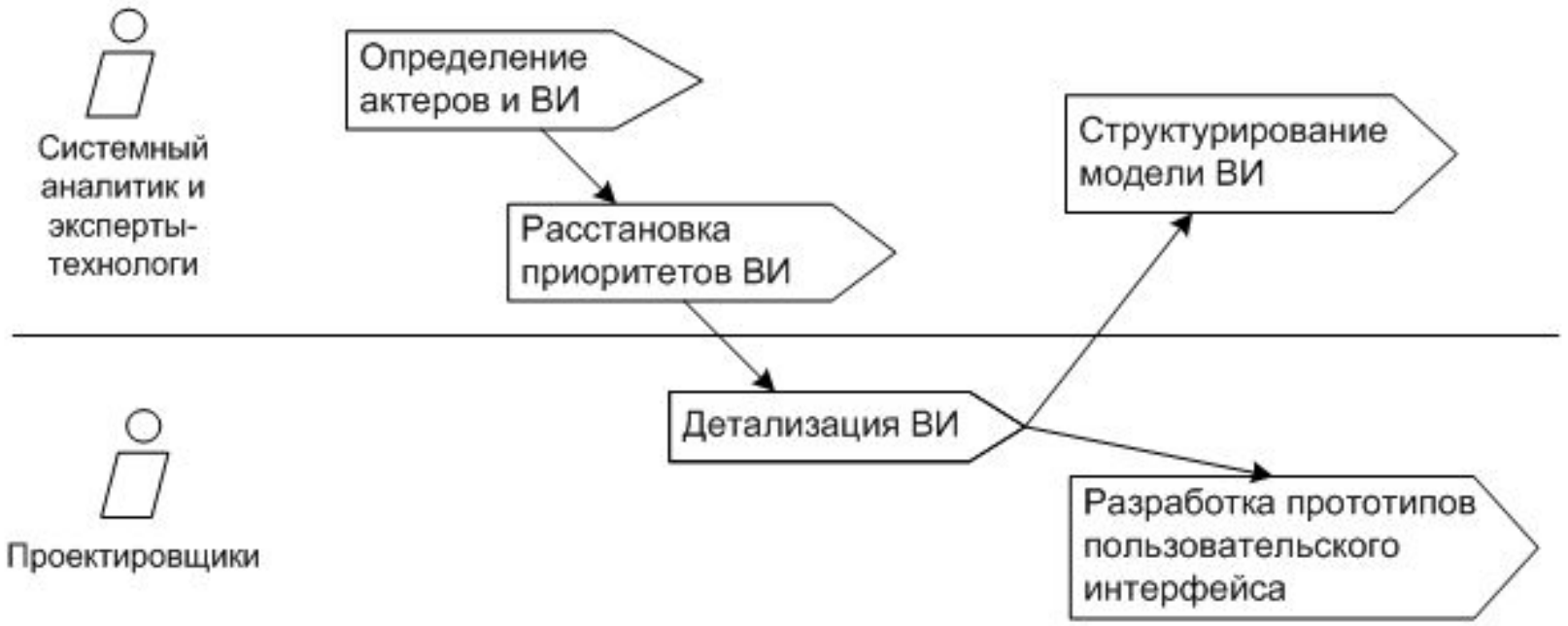
UML 2

Основной набор моделей Унифицированного процесса



Связи между моделями

Последовательность действий при построении модели вариантов использования

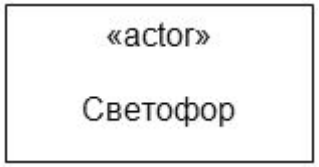


Цели ВИ

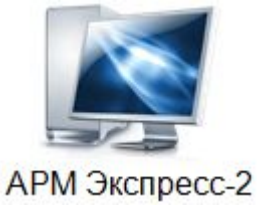
* - унификация элементов модели;
* - выделение общих и совместно применяемых частей вариантов использования;
* - обеспечение семантической (смысловой) согласованности между диаграммами и их элементами

Примеры отображения актеров

«проволочный человечек»



класс с текстовым стереотипом «actor»



произвольная иконка

Примеры вариантов использования

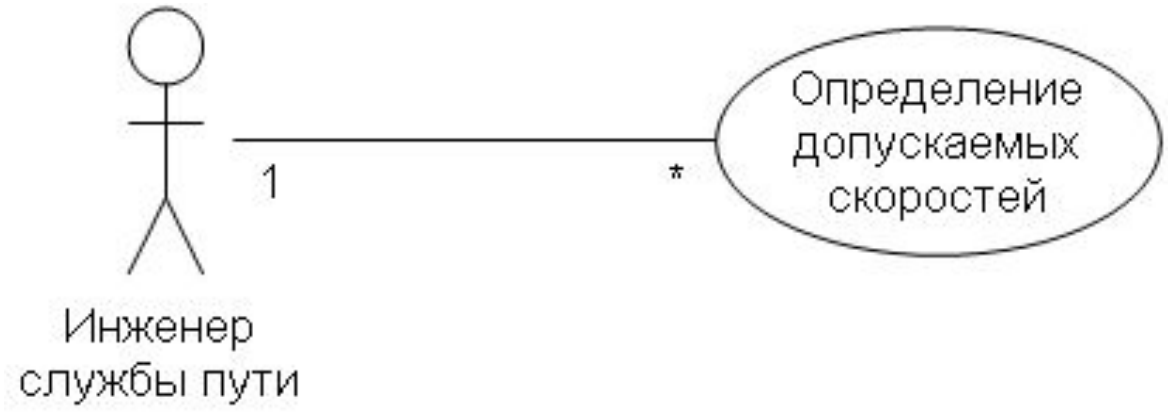
Пример примечания

Отношения четырех видов

* - ассоциация;
* - обобщение;
* - включение;
* - расширение.

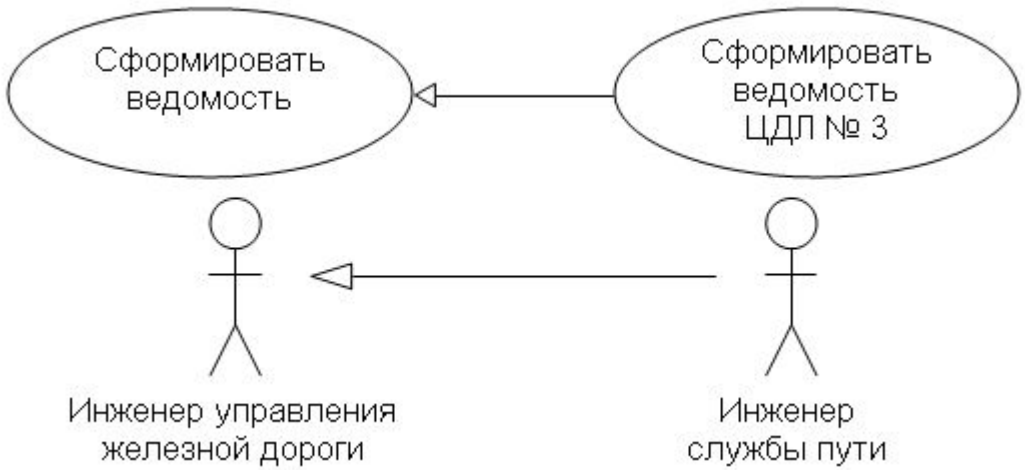
Отношение ассоциации

* служит для обозначения взаимодействия актера с вариантом использования



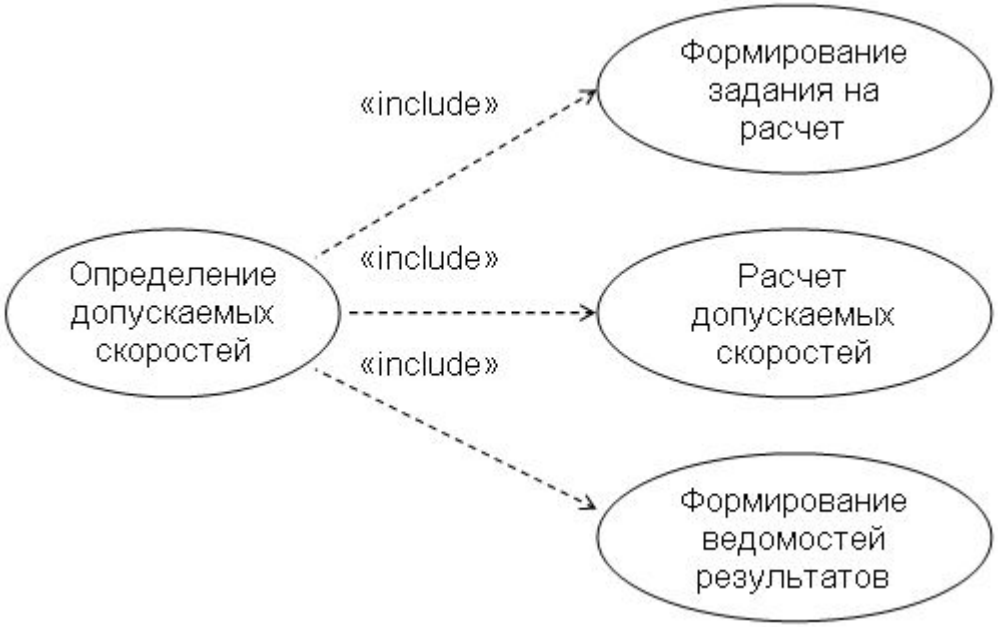
Отношение обобщения

* служит для указания того факта, что некоторая сущность А может быть обобщена до сущности В. В этом случае сущность А будет являться специализацией сущности В. На диаграмме данный вид отношения можно отображать только между **однотипными** сущностями (между двумя вариантами использования или двумя актерами).



Отношение включения

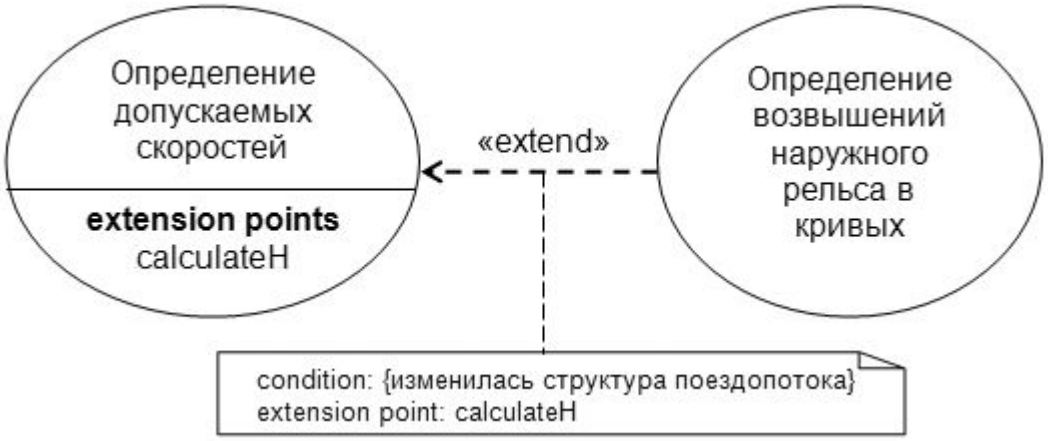
* указывает, что некоторое заданное поведение одного варианта использования ***обязательно*** включается в качестве составного компонента в последовательность поведения другого варианта использования



Стрелка включения должна быть направлена от базового (составного) варианта к включаемому и помечена стереотипом «include» или «uses»

отношение расширения

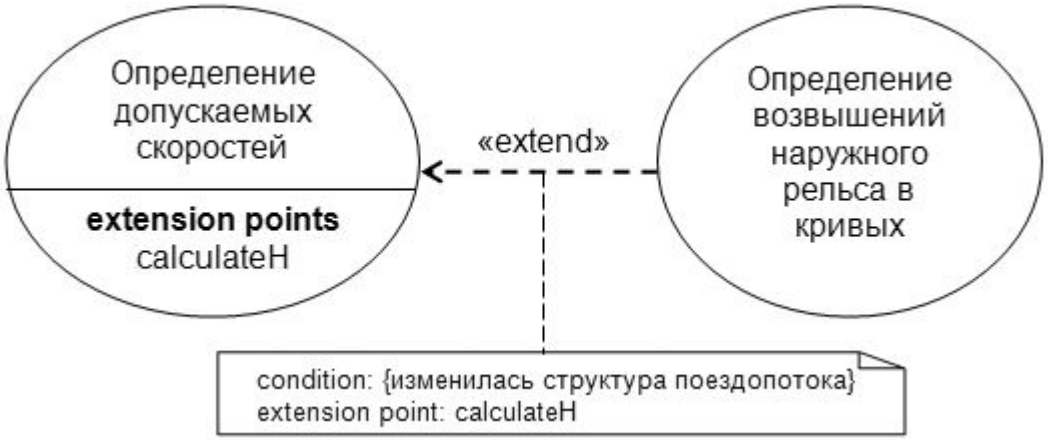
* определяет ***потенциальную*** *возможность* включения поведения одного варианта использования в состав другого. Т. е. дочерний вариант использования может как вызываться, так и не вызываться родительским



Стрелка расширения должна быть направлена от включаемого варианта к базовому и помечена стереотипом «extend»

точки расширения

* Варианты использования, которые расширяют базовый, подключаются к нему (активируются при его выполнении) через так называемые **точки расширения** (extension points).
* Каждая точка расширения маркируется и **условием** (condition) **активации.** Обычно перечень точек расширения указывается в базовом варианте использования ниже горизонтальной линии.



Правила и рекомендации по разработке диаграмм вариантов использования

1. Рекомендуется вначале построить **контекстную диаграмму,** на которой отображаются основные варианты использования (функции) системы, а затем для каждого из них построить **диаграммы декомпозиции** (детализации).
2. Контекстная диаграмма может представлять собой **несвязный граф** (в отличие от IDEF0 и DFD).
3. Чрезмерная детализация вариантов использования не требуется. Вариант использования – это относительно крупный блок функциональности системы.

4. Отдельная диаграмма (контекстная или декомпозиции) не должна быть перенасыщена элементами. Рекомендуется отображать на диаграмме не более **15** вариантов использования.

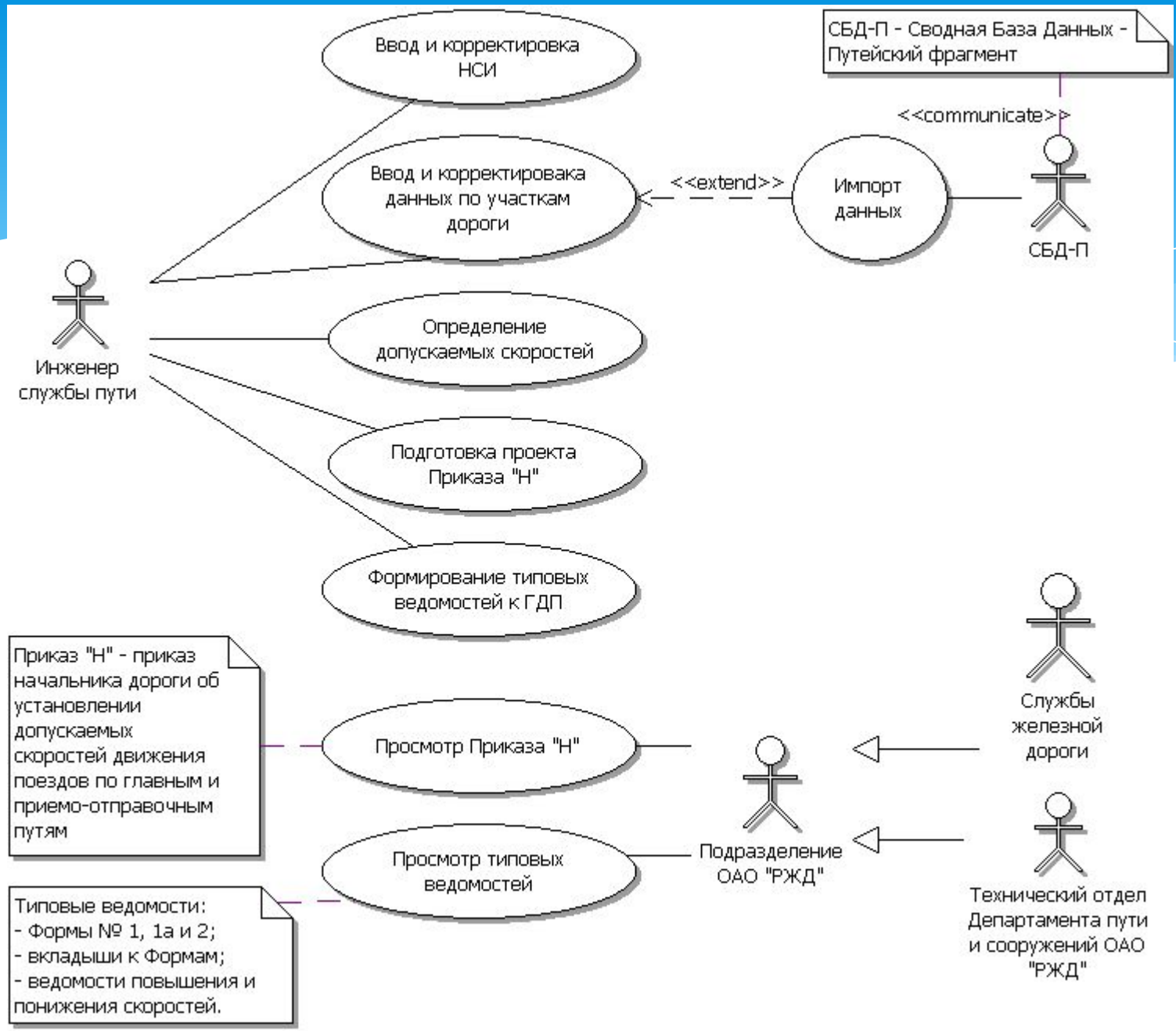
5. Располагать элементы следует так, чтобы была видна **логическая последовательность выполнения** вариантов использования и было **минимум пересечений** между отношениями.

6. Перед построением диаграммы необходимо задокументировать потоки событий в системе. ***Поток событий*** – это процесс обработки данных, реализуемый в рамках одного или нескольких вариантов использования. Описание потока включает информацию о том, какие обязанности возлагаются на актеров, а какие – на систему:

- **краткое описание поведения**, реализуемого в варианте использования;
- **предусловия** – условия, которые должны быть соблюдены, прежде чем вариант использования может быть задействован. (завершение выполнения другого варианта использования или наличие у пользователя прав доступа);

- **основной поток событий** описывает, что должно происходить во время выполнения варианта использования в наиболее распространенном (типовом) случае. В этом случае дочерние варианты использования **связаны** с базовым отношением **включения;**
- **альтернативные потоки событий** описывают исключительные ситуации (ввод неправильного пароля, необходимость выполнения дополнительных действий). Дочерние варианты использования при разработке диаграммы связываются с базовым **отношением расширения;**
- **постусловия** – условия, которые должны быть выполнены после завершения варианта использования (обязательное сохранение результатов расчета в базе данных на сервере)

7. На диаграммах не следует отображать особенности реализации вариантов использования и внутренней организации системы, связанные со спецификой используемых программных и аппаратных средств. Данные диаграммы в первую очередь предназначены для **совместного с заказчиком определения функциональных требований к системе**. Поэтому понимать (интерпретировать) отображенное на диаграммах и заказчик и разработчик должны одинаково.


Ввод и корректировка НСИ
СБД-П - Сводная База Данных - Путейский фрагмент
<<communicate>>
Ввод и корректировака данных по участкам дороги
<<extend>>
Импорт данных
СБД-П
Инженер службы пути
Определение допускаемых скоростей
Подготовка проекта Приказа "Н"
Формирование типовых ведомостей к ГДП
Приказ "Н" - приказ начальника дороги об установлении допускаемых скоростей движения поездов по главным и приемо-отправочным путям
Просмотр Приказа "Н"
Службы железной дороги
Подразделение ОАО "РЖД"
Просмотр типовых ведомостей
Технический отдел Департамента пути и сооружений ОАО "РЖД"
Типовые ведомости:
- Формы № 1, 1а и 2;
- вкладыши к Формам;
- ведомости повышения и понижения скоростей.


Формирование детальной ведомости
Формирование ведомости "Основная допускаемая скорость"
Формирование ведомости "Дополнительные ограничения скорости"
<<include>>
<<include>>
<<include>>
Формирование задания на расчет
Расчет допускаемых скоростей
Формирование ведомостей результатов
<<include>>
<<include>>
<<include>>
Определение допускаемых скоростей
Инженер службы пути
<<extend>>
Определение возвышений наружного рельса в кривых

Способы детализации вариантов использования

Возможные «канонические» варианты

- с помощью диаграмм автоматов (состояний);
- с помощью диаграмм деятельности (аналог блок-схем);
- с помощью диаграмм взаимодействия (последовательности и коммуникации).

Диаграммы автоматов (состояний)

Диаграммы автоматов (state machine)

* используются **для описания поведения**, реализуемого в рамках варианта использования, или **поведения экземпляра сущности** (класса, объекта, компонента, узла или системы в целом) .
* Моделируется через описание **возможных состояний** экземпляра сущности и **переходов между ними** на протяжении его жизненного цикла, начиная от создания и заканчивая уничтожением.
* Диаграмма являет собой **связный ориентированный граф**
  * Вершины являются состояния,
  * Дуги служат для обозначения переходов из состояния в состояние.

Состояние (state)

* понимается ситуация в ходе **жизни экземпляра** сущности, когда эта **ситуация удовлетворяет некоторому условию,**
* **экземпляр выполняет некоторые операции** или **ждет наступления некоторого события.** (для объекта его состояние может быть задано в виде набора конкретных значений атрибутов, при этом изменение этих значений будет приводить к изменению состояния моделируемого объекта.)

Операции

* ***Действие*** (action) – это **атомарная операция,** выполнение которой не может быть прервано, приводящая к **смене состояния** или **возвращающая значение** (операции создания или уничтожения объекта, расчет факториала)
* ***Деятельность*** (activity) – это **составная** (неатомарная) операция, реализуемая экземпляром в конкретном состоянии, выполнение которой может быть прервано. (процедуры расчета допускаемых скоростей или шифрования данных)

Событие (event)

* Спецификация существенного факта, который может произойти в конкретный момент времени.
* **Внешние события** передаются между системой и актерами (например, нажатие кнопки или посылка сигнала от датчика передвижений).
* **Внутренние события** передаются между объектами внутри системы.

виды событий

- посылка сообщения (message):
  - ***Вызов*** (call) – спецификация факта посылки синхронного сообщения между объектами, предписывающего выполнение операции (действия или деятельности) объектом, которому посылается сообщение. После посылки вызова объект-отправитель передает управление объекту-получателю и после выполнения последним операции получает управление обратно. (закрасить фигуру красным фоном fill(red) или рассчитать допускаемые скорости calculateVdop())
  - ***Сигнал*** (signal) – спецификация факта посылки асинхронного сообщения между объектами. Исключения, которые поддерживаются в большинстве современных языков программирования, являются наиболее распространенным видом внутренних сигналов.
  - ***Любое сообщение*** (any receive);

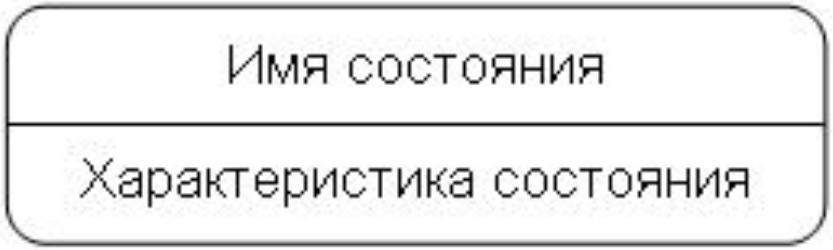
- ***Событие времени*** (time) – спецификация факта, обозначающего наступление конкретного момента времени (absolute time) или истечение определенного промежутка времени (relative time).

обозначается с помощью ключевых слов **«at»** (at 9:00:00) и **«after»** (after 2 seconds).

- ***Изменение состояния***(change) – спецификация логического условия, соответствующего изменению состояния экземпляра сущности.

обозначается с помощью ключевого слова **«when»** (when A < B) или **сторожевого условия** ([A < B]).

Способы отображения состояний



Характеристика состояния

может содержать описание выполняемых операций, перед которыми указывается одна из стандартных ***меток***:

- **entry** (вход) – действие при входе, выполняемое вне зависимости от того, по какому переходу был выполнен вход в состояние. (создать соединение с базой данных entry / createConnect())
- **exit** (выход) – действие при выходе, выполняемое вне зависимости от того, по какому переходу был выполнен выход из состояния (закрыть соединение с базой данных exit / closeConnect())
- **do** (выполнять) – деятельность в состоянии. Находясь в состоянии, экземпляр сущности **может бездействовать** и ждать наступления некоторого события, а **может выполнять** длительную операцию. (рассчитать допускаемые скорости do / calculateVdop())

Допускается указывать несколько операций в виде отдельных строк, каждая из которых начинается с метки «do», или в виде одной строки, операции в которой отделены друг от друга точкой с запятой.

Переход (transition)

Отношение между двумя состояниями, показывающее возможный путь изменения состояния экземпляра сущности. Считается, что в **состоянии экземпляр сущности находится продолжительное время, а переход выполняется мгновенно.**

Переход отображается в виде **однонаправленной ассоциации между двумя состояниями.** При смене состояний говорят, что **переход срабатывает.** До срабатывания перехода экземпляр сущности находится в состоянии, называемом ***исходным***, а после его срабатывания – в ***целевом.***

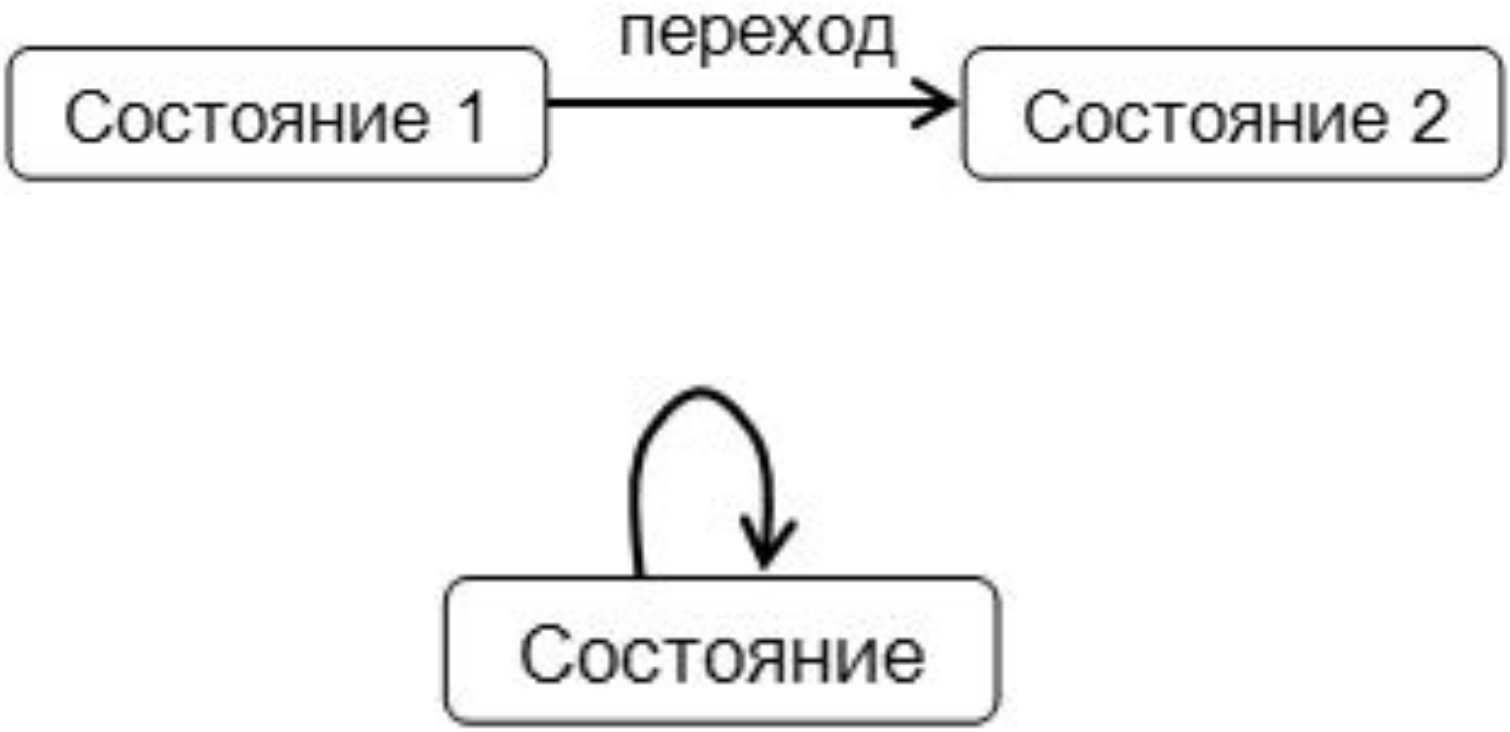
Виды переходов

* ***Нетриггерный*** **(**переход по завершении), срабатывает неявно, когда все основные операции (с метками entry, do и exit) в исходном состоянии успешно завершают свою работу. Данный вид перехода обозначается **стрелкой без надписи.**
* ***Триггерный*** необходимо наступление некоторого **события, которое записывается над стрелкой.**
* ***«событие [сторожевое условие] / действие»***
* действие представляет собой **атомарную операцию,** выполняемую сразу после срабатывания соответствующего перехода и до начала каких бы то ни было операций в целевом состоянии. Разрешается указывать **не одно, а несколько обособленных действий,** отделенных друг от друга точкой с запятой. Обязательное требование – все действия в списке должны четко различаться между собой и следовать в порядке их записи.

Примеры спецификации переходов

- mouseClick();
- mouseClick() / setFocus();
- mouseClick() [isEnabled()] / setFocus;
- at 14:00:00 или [getTime() = 14:00] – текущее время на компьютере равно 14 часам;
- «столкновение»
- «выход из строя».

Простой и рефлексивный переходы



Составные состояния

* ***составные состояния*** (composite state), состоящие из вложенных в них ***подсостояний*** (substate).

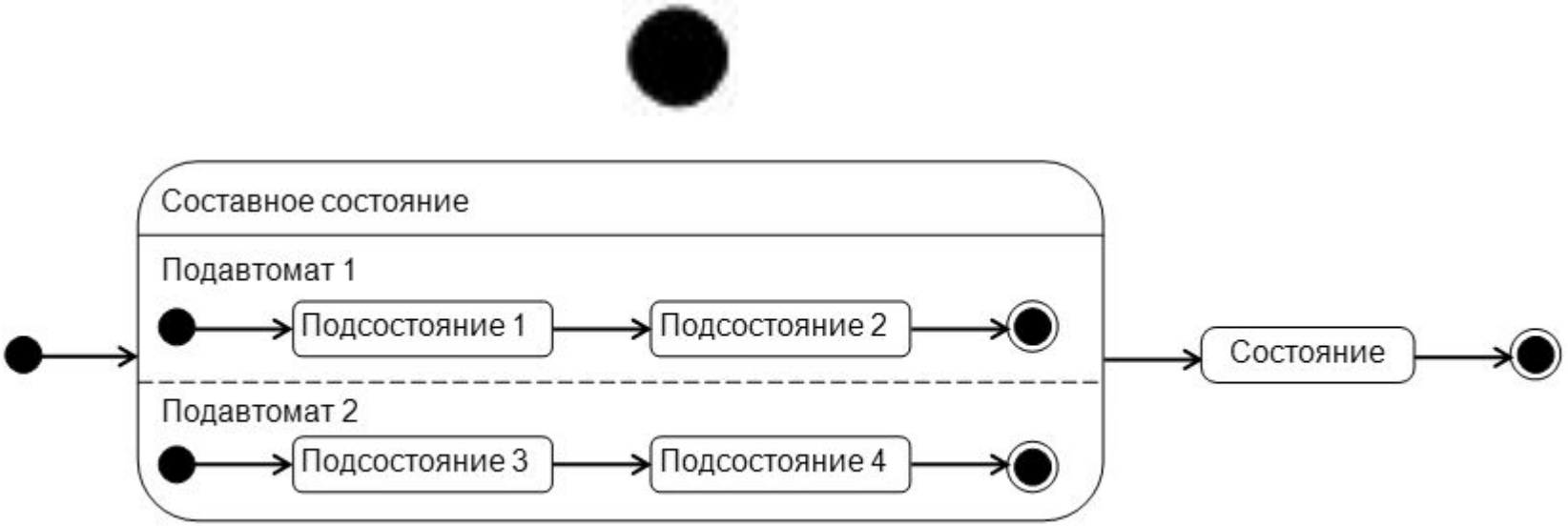
* Составное состояние, которое может использоваться в разных контекстах, в т.ч. и для разных диаграмм (автоматов), называются ***подавтоматами*** (submachine state).
* Составное состояние может быть разбито на ***зоны*** (regions), - ***параллельными подавтоматами*** (concurrent substates).
* Если на диаграмме имеется составное состояние с вложенными параллельными подавтоматами**, экземпляр сущности может одновременно находиться в нескольких подсостояниях**, но не более чем **по одному из каждого подавтомата.**
* Если какой-либо из подавтоматов пришел в свое конечное состояние раньше других, то он должен ожидать, пока другие подавтоматы не придут в свои конечные состояния.

Составное состояние с вложенными параллельными подавтоматами

Составное состояние со скрытой внутренней структурой

Псевдосостояния. Начальное (initial)

* Начальное состояние автомата, начальное подсостояние составного состояния или параллельного подавтомата.
* Из начального состояния могут **только исходить переходы.**



Псевдосостояния. Конечное (final)

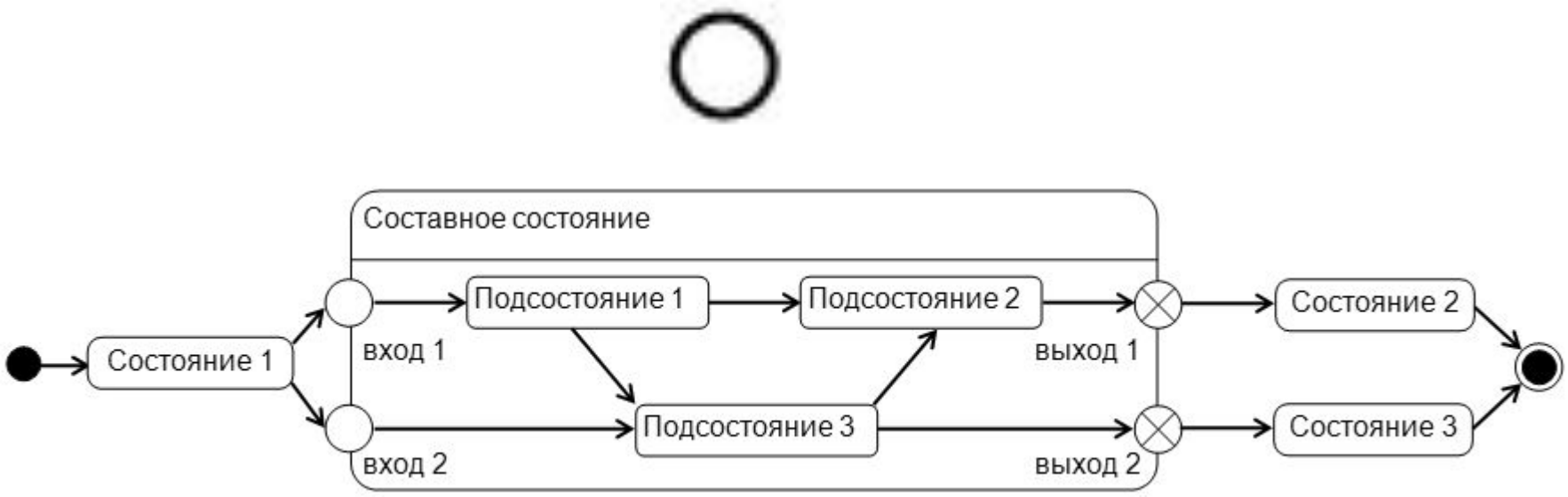
* Конечное состояние автомата, конечное подсостояние составного состояния или параллельного подавтомата.
* В конечное состояние могут **только входить переходы.**
* В стандарте UML 2.5 считается состоянием, а не псевдосостоянием.

Немедленное завершение (terminate)

* Аналогично конечному состоянию, подразумевает **немедленное прекращение деятельности** и **уничтожение экземпляра сущности**, для которой построен автомат.

Точка входа (entry point)

* Точка входа в автомат или составное состояние. Может быть несколько. Допускается крепление к границе составного состояния.

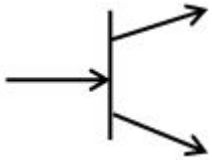


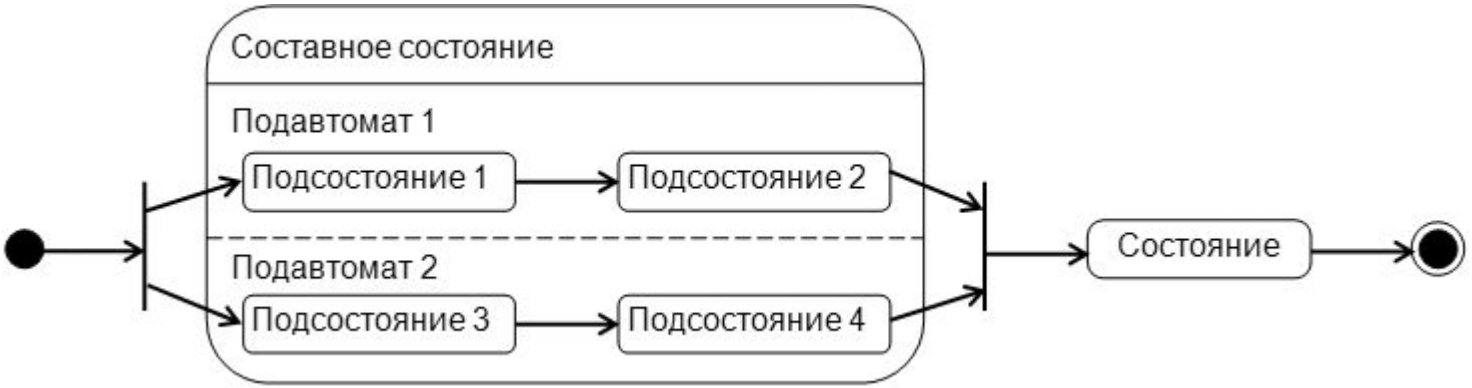
Точка выхода (exit point)

* Точка выхода из автомата или составного состояния. Может быть несколько. Допускается крепление к границе составного состояния.

Ветвление (англ. fork)

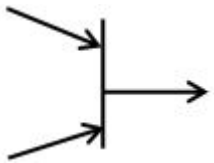
* Ветвление переходов в параллельные подавтоматы.





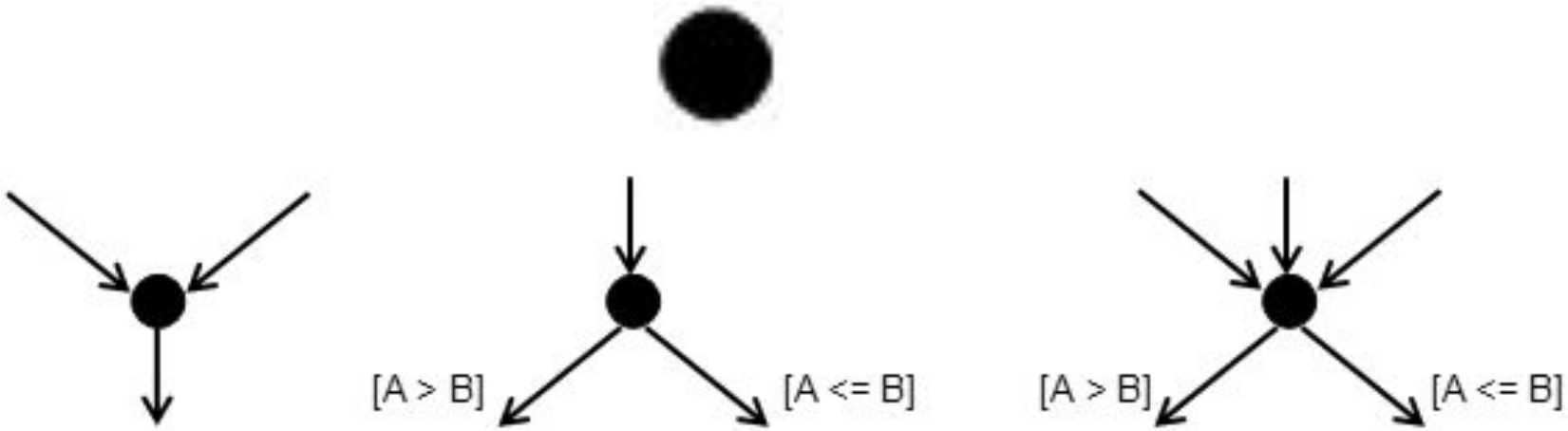
Соединение (join)

* Соединение переходов из параллельных подавтоматов. Выполняет функцию синхронизации выхода из параллельных подавтоматов составного состояния.



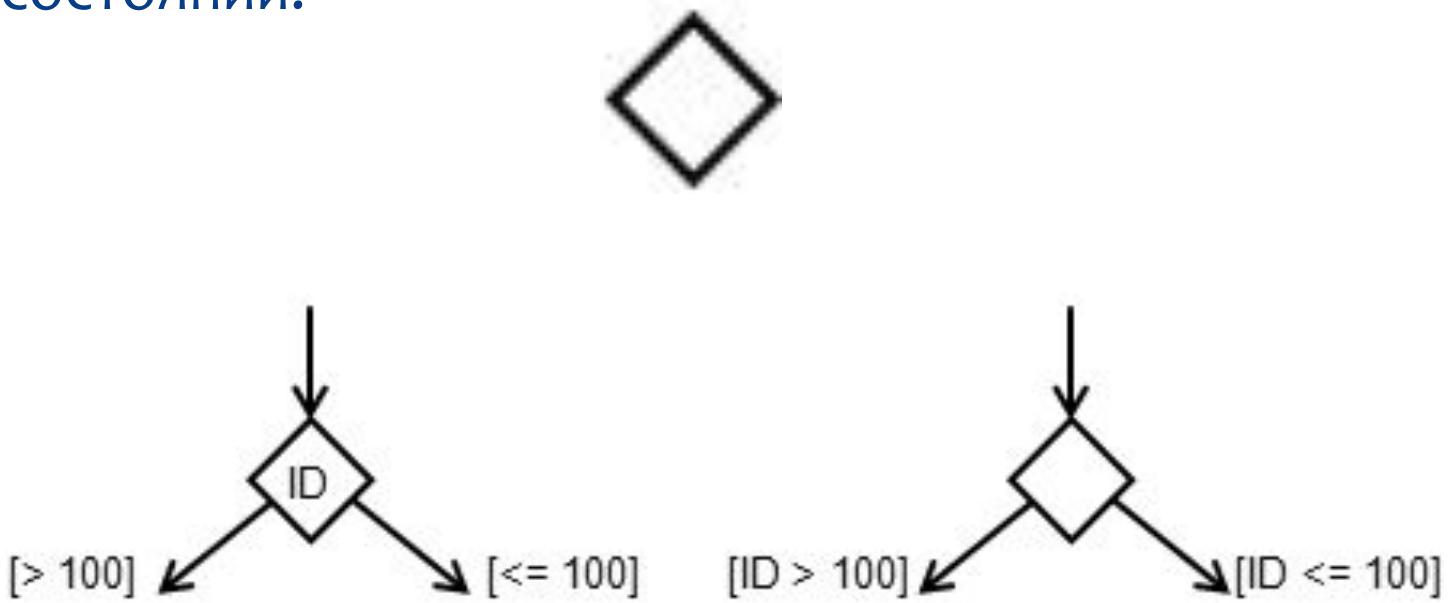
Переход (junction)

* Соединение и ветвление переходов для последовательных состояний. В случае ветвления для каждой исходящей из перехода ассоциации должно быть задано сторожевое условие.



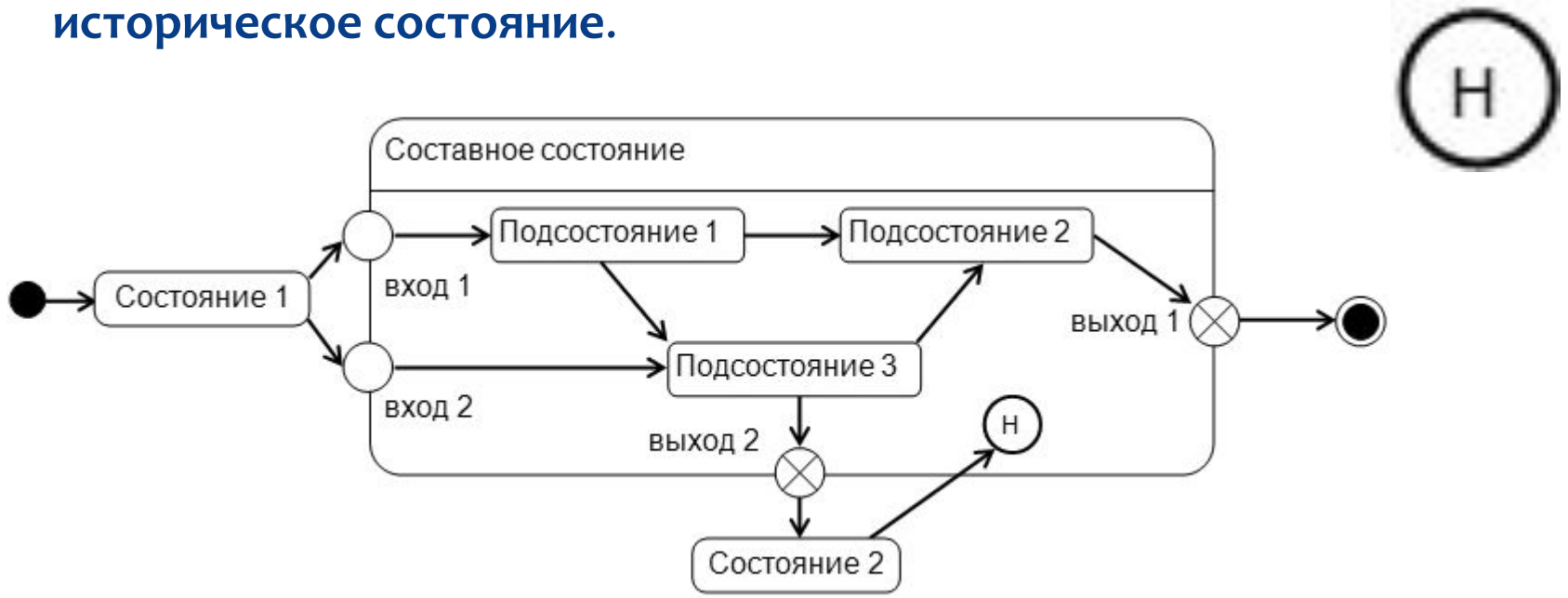
Выбор (choice)

* Ветвление переходов для последовательных состояний.



Поверхностное историческое (shallow history)

* Указывается внутри составного состояния и подразумевает **запоминание текущей конфигурации составного состояния** при выходе из него. Переход в историческое состояние восстанавливает запомненную конфигурацию составного состояния и продолжает работу составного состояния с того момента, когда его прервали в прошлый раз. Внутри составного состояния может быть **только одно историческое состояние.**



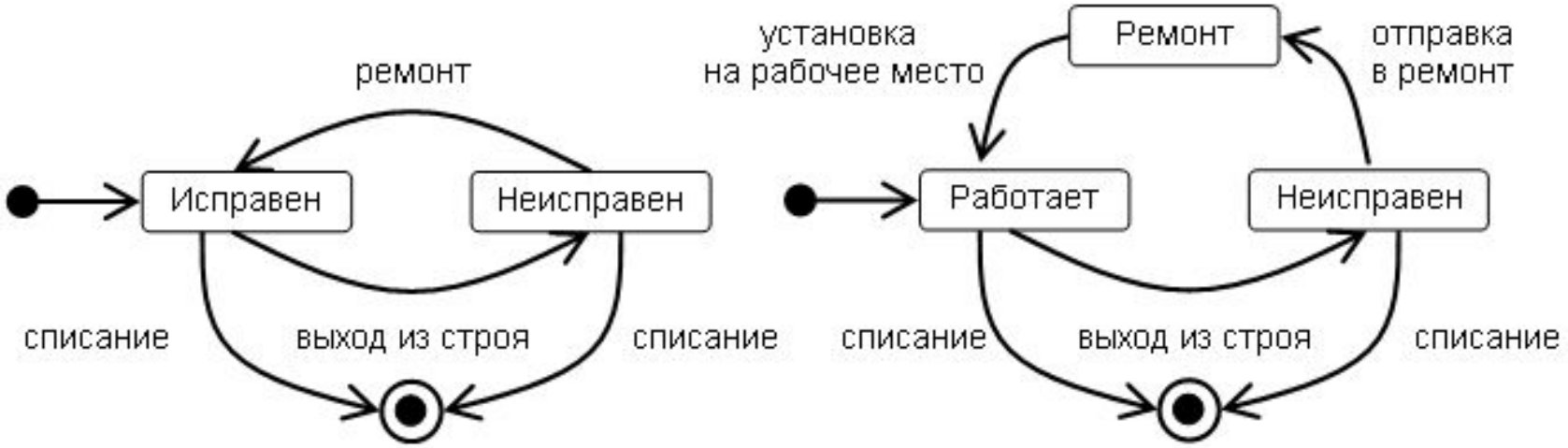
Глубинное историческое (deep history)

* Аналогично поверхностному историческому состоянию, но **распространяется на все уровни вложенности подсостояний.**

Правила и рекомендации по разработке диаграмм автоматов

1. При выделении состояний и переходов **длительность срабатывания переходов должна быть существенно меньшей, чем нахождение моделируемого объекта в соответствующих состояниях.**

Каждое из состояний должно характеризоваться **определенной устойчивостью во времени.**



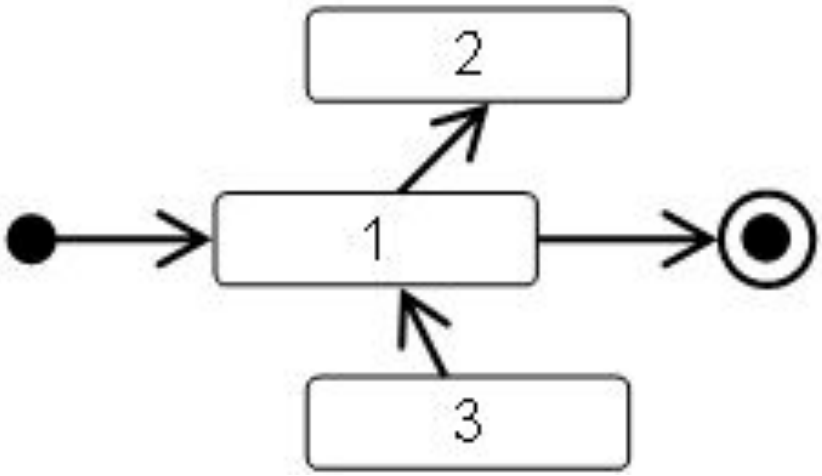
2. Автомат (диаграмма) **должен начинаться знаком начального состояния** и **заканчиваться знаком конечного**.

Начальное состояние указывается только один раз, а конечных может быть несколько в целях минимизации пересечений переходов.

Для подавтоматов **рекомендуется** придерживаться этого же правила или использовать **точки входа / выхода**. Допускается не указывать начальных / конечных состояний и точек входа / выхода для составных состояний или подавтоматов, когда начальное подсостояние (подсостояния) очевидны.

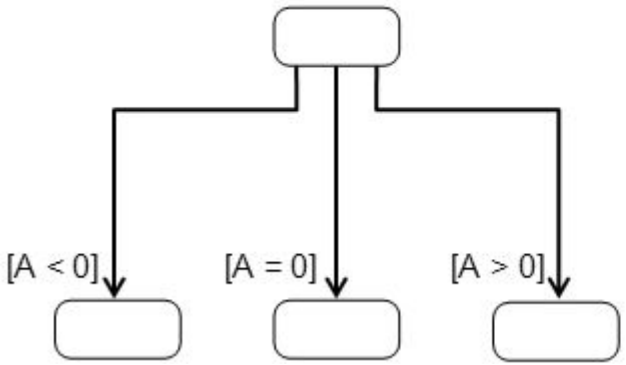
3. Для облегчения восприятия диаграммы рекомендуется **использовать декомпозицию** со скрытием составных состояний.

4. Диаграмма не должна содержать изолированных состояний и переходов. Переходы и их спецификация должны быть заданы таким образом, чтобы на графе каждое состояние было потенциально достижимо из начального и из любого состояния было потенциально достижимо конечное.

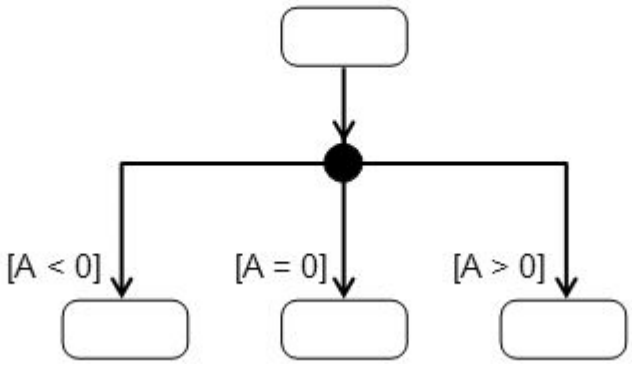


* Триггерные переходы по условию на диаграмме можно показать тремя способами.

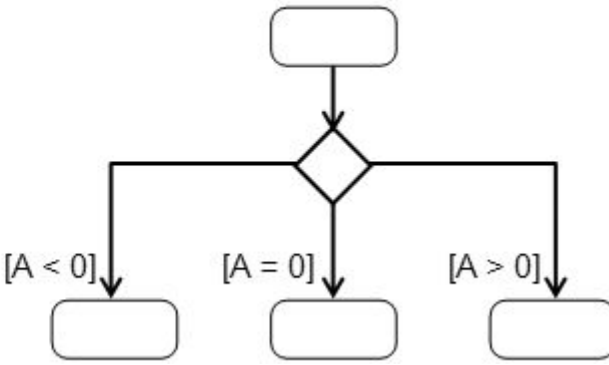
а)



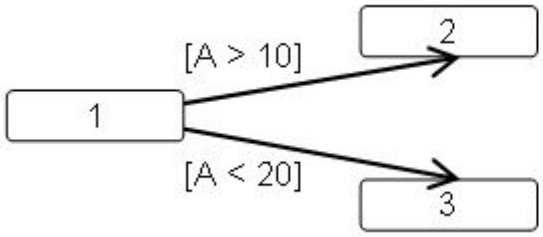
б)

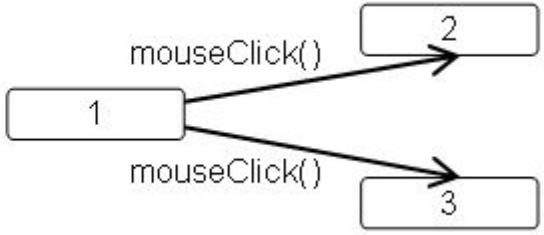


в)



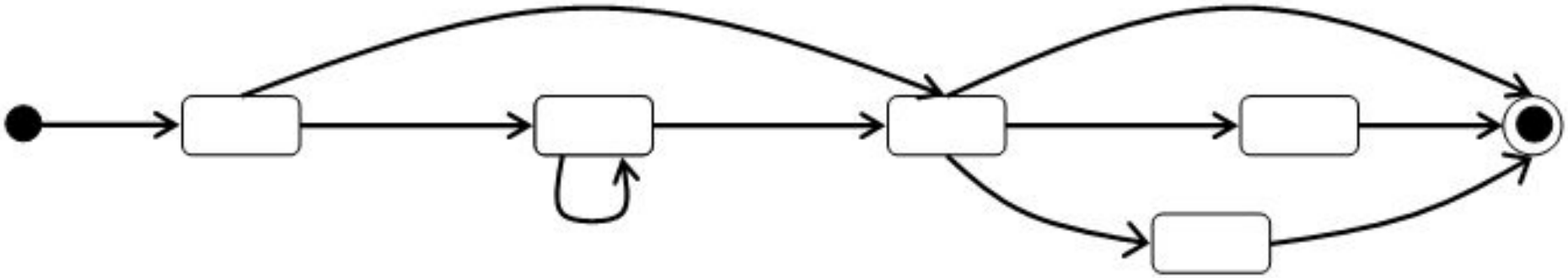
6. В каждый момент времени автомат или подавтомат должен находиться только в одном состоянии. Это означает, что спецификация переходов из одного состояния не должна допускать потенциальной возможности перехода в два и более состояний.

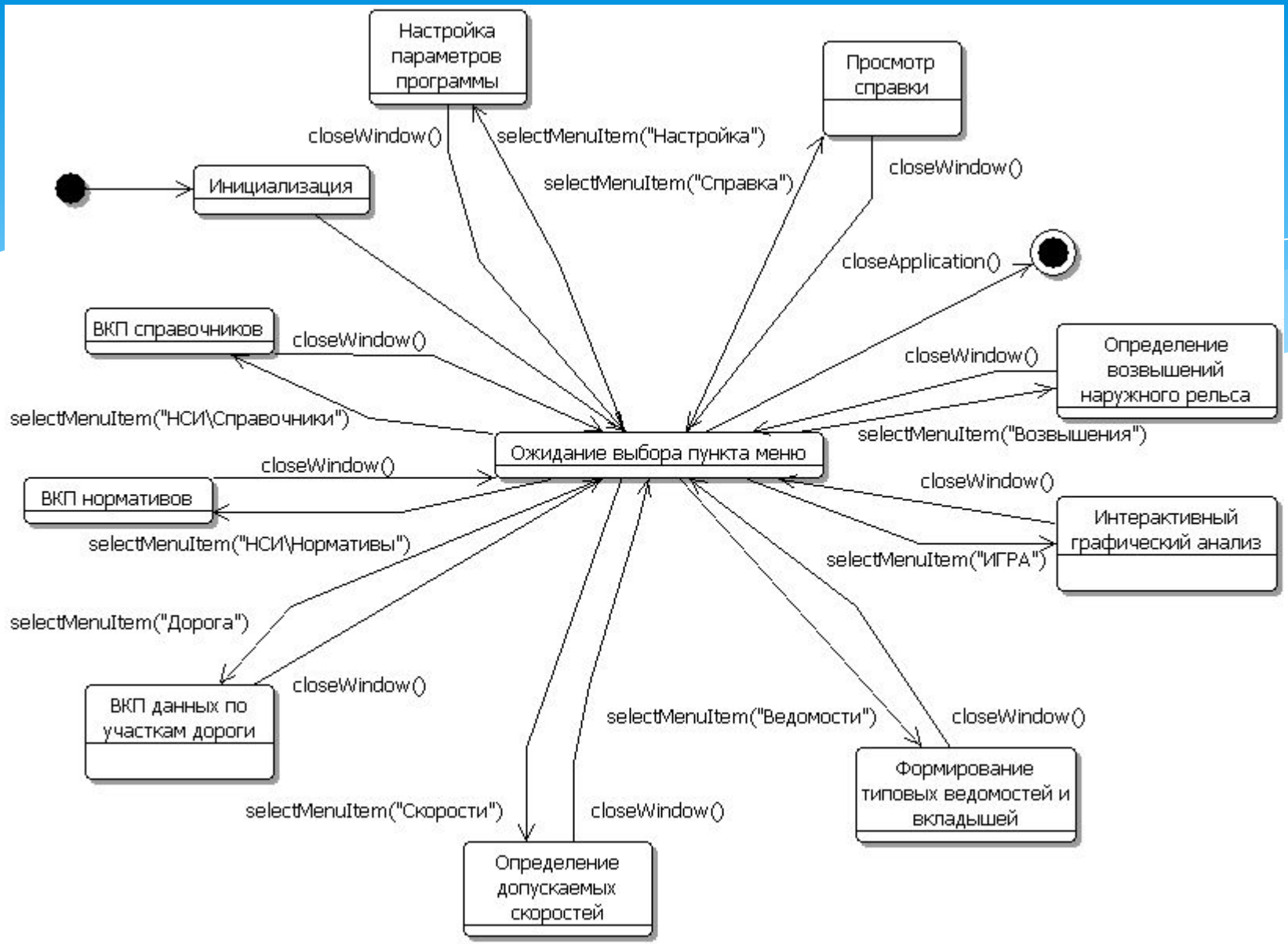


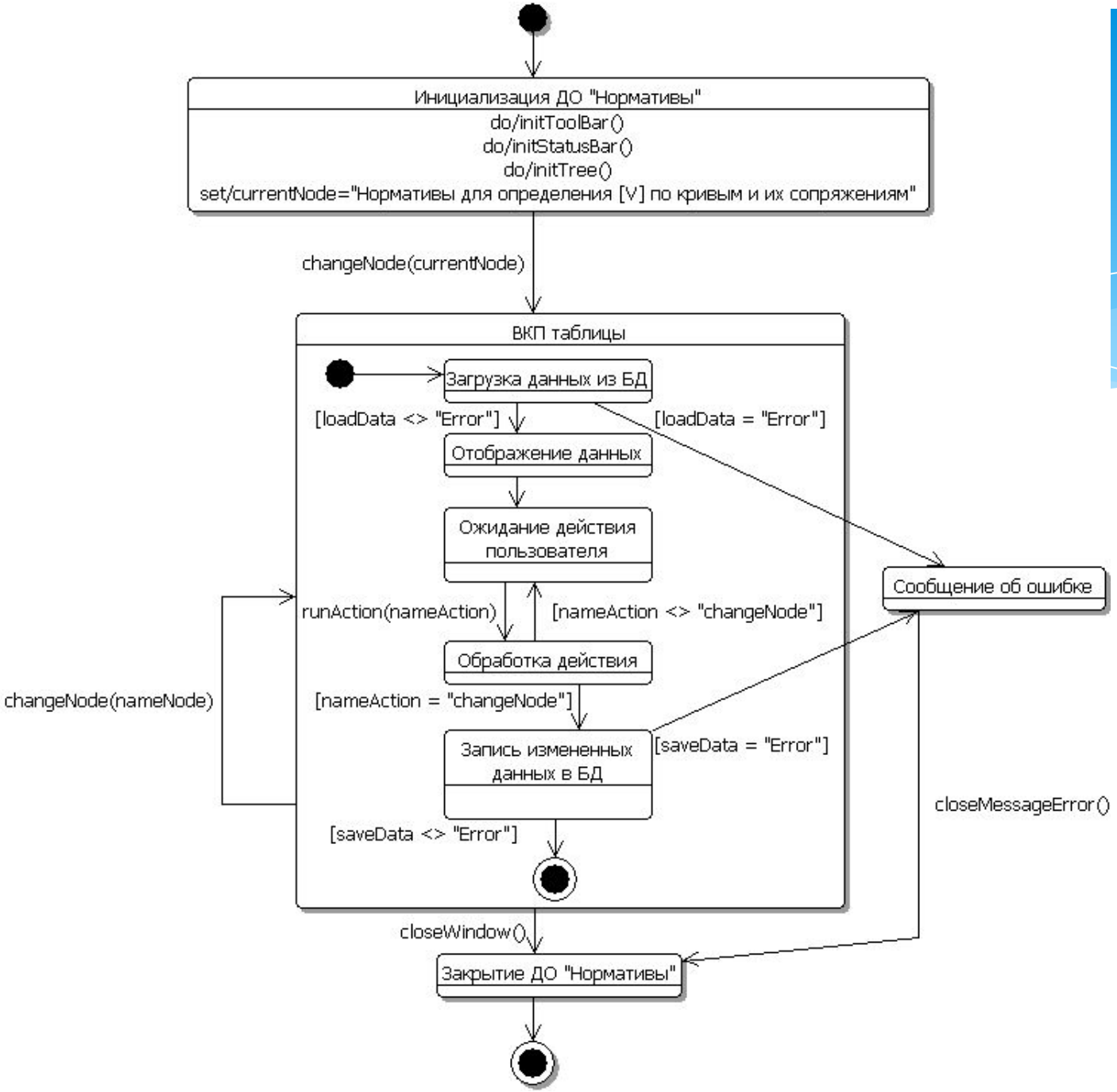


Исключением из этого правила является параллельный переход в подсостояния параллельных подавтоматов одного составного состояния.

Пример детализации варианта использования




Настройка параметров программы
Просмотр справки
closeWindow()
selectMenuItem("Настройка")
selectMenuItem("Справка")
closeWindow()
Инициализация
closeApplication()
ВКП справочников
closeWindow()
closeWindow()
Определение возвышений наружного рельса
selectMenuItem("НСИ\Справочники")
selectMenuItem("Возвышения")
Ожидание выбора пункта меню
closeWindow()
closeWindow()
ВКП нормативов
Интерактивный графический анализ
selectMenuItem("НСИ\Нормативы")
selectMenuItem("ИГРА")
selectMenuItem("Дорога")
closeWindow()
ВКП данных по участкам дороги
selectMenuItem("Ведомости")
closeWindow()
Формирование типовых ведомостей и вкладышей
selectMenuItem("Скорости")
closeWindow()
Определение допускаемых скоростей

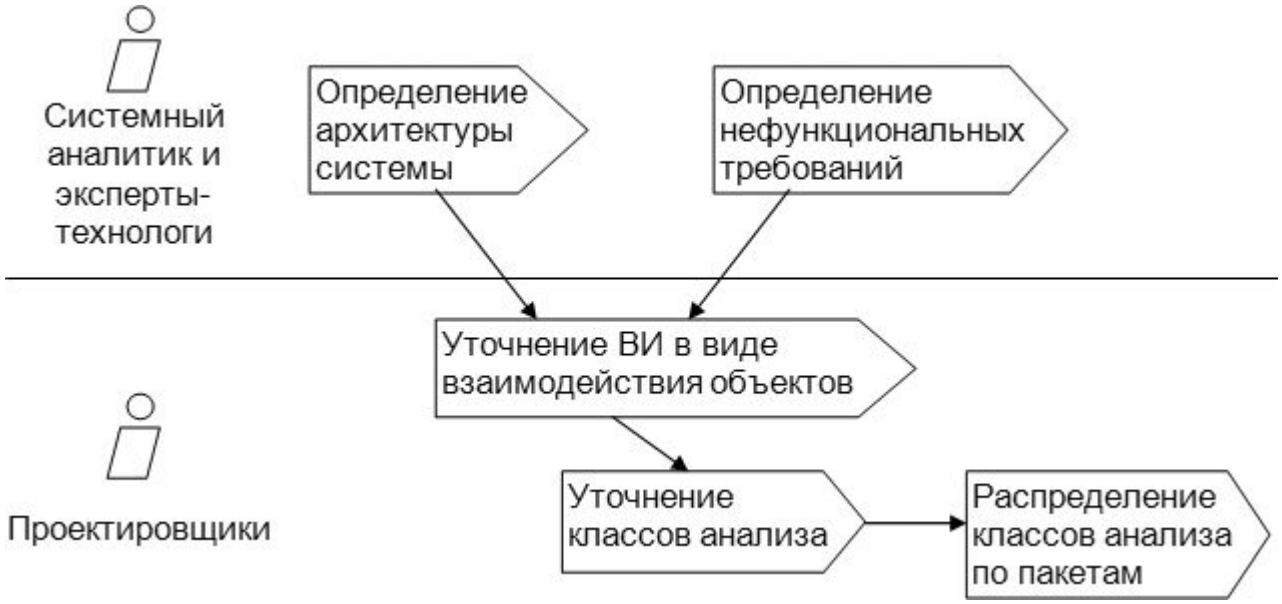

Инициализация ДО "Нормативы"
do/initToolBar()
do/initStatusBar()
do/initTree()
set/currentNode="Нормативы для определения [V] по кривым и их сопряжениям"
changeNode(currentNode)
ВКП таблицы
Загрузка данных из БД
[loadData <> "Error"]
[loadData = "Error"]
Отображение данных
Ожидание действия пользователя
runAction(nameAction)
[nameAction <> "changeNode"]
Обработка действия
changeNode(nameNode)
[nameAction = "changeNode"]
Запись измененных данных в БД
[saveData = "Error"]
[saveData <> "Error"]
Сообщение об ошибке
closeMessageError()
closeWindow()
Закрытие ДО "Нормативы"

Модель анализа

Задачи модели

- выявление внутренней архитектуры (определения подсистем и основных классов);
- поиск альтернативных вариантов реализации системы (подсистем) и выбора основного;
- уточнение всех требований (функциональных и нефункциональных).

Обобщенная схема технологического процесса «Анализ требований»

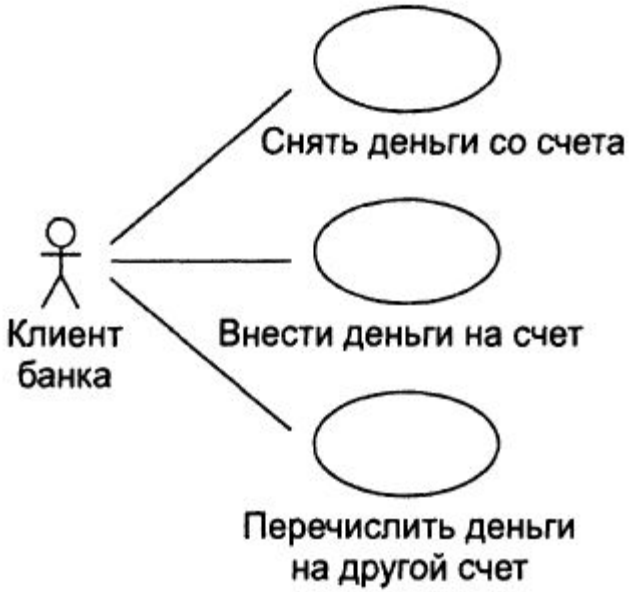


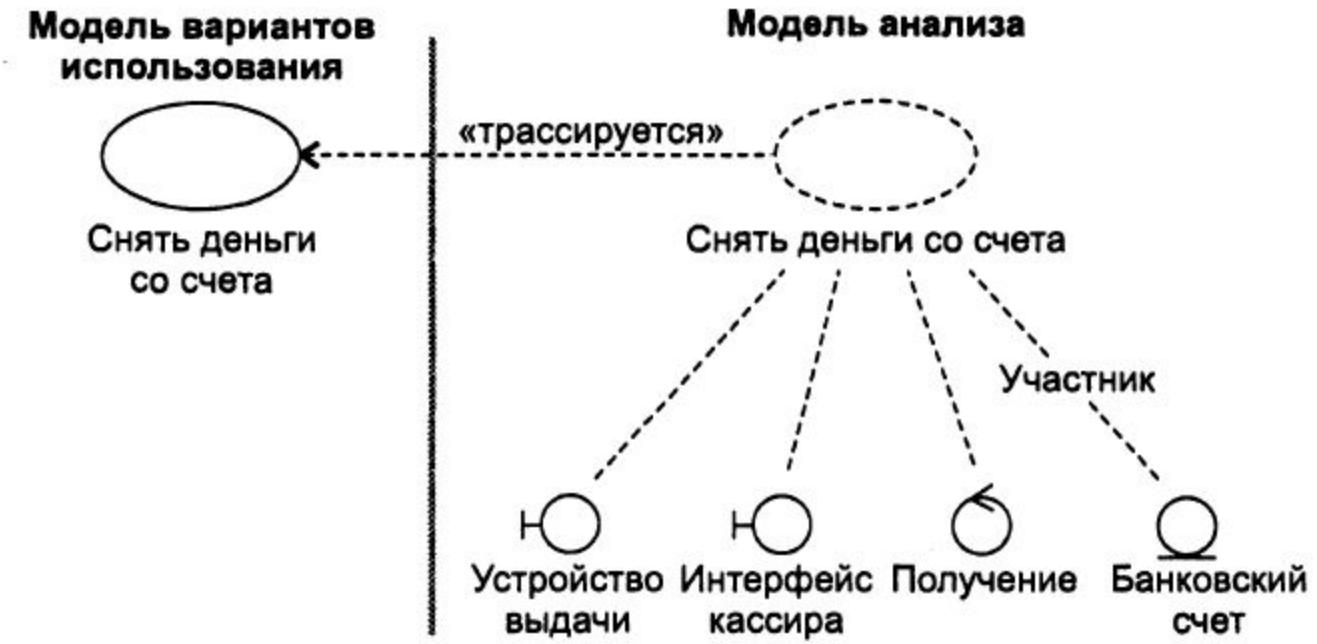
Диаграммы (*основные артефакты*)

- классов анализа;
- последовательности;
- коммуникации.

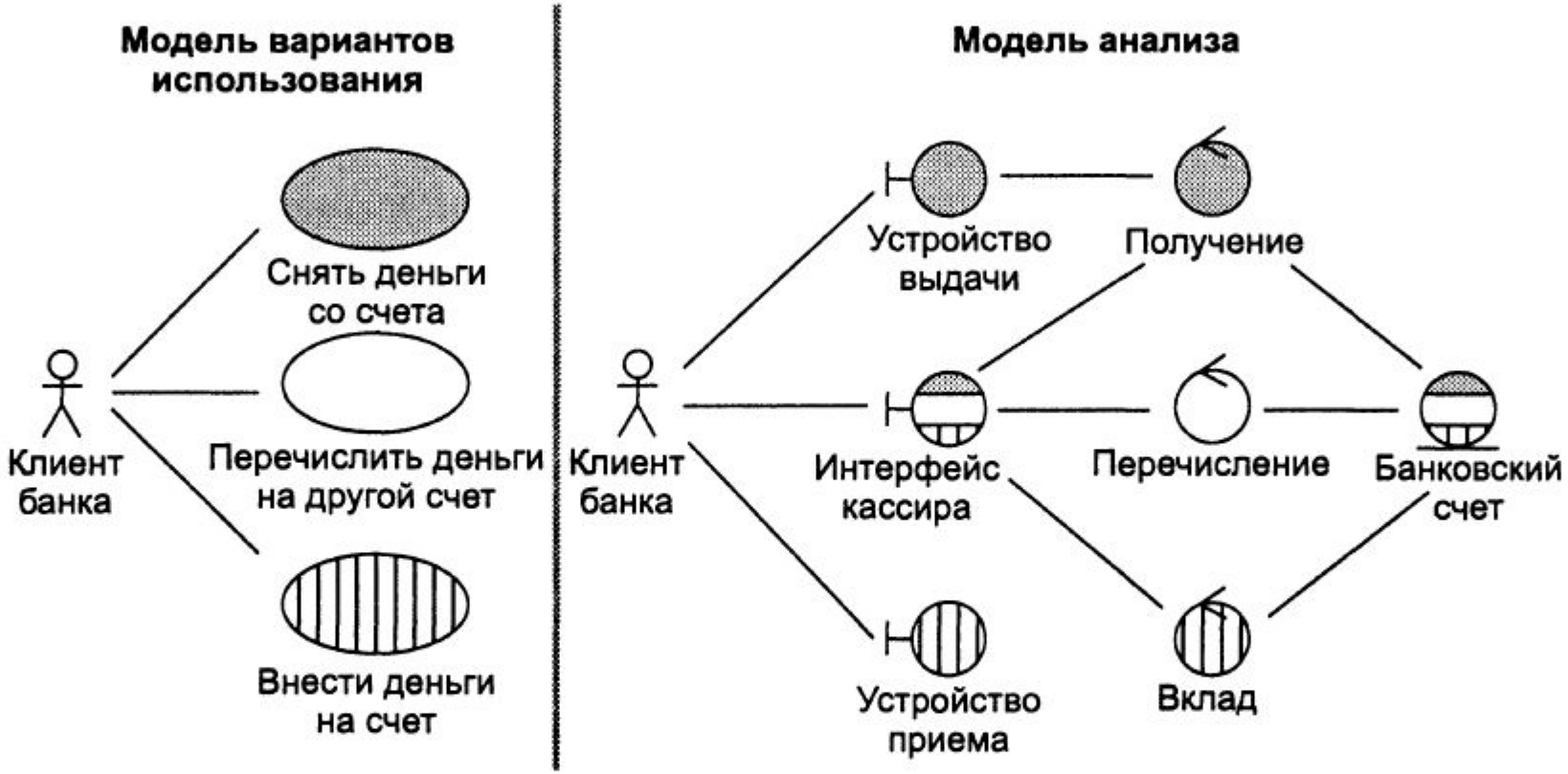
группы требований (параметры качества)

- ***практичности*** – характеризуют легкость освоения и эксплуатации системы, интуитивность и эргономичность пользовательского интерфейса, согласованность пользовательского интерфейса, документации и обучающих материалов;
- ***надежности*** – характеризуют частоту появления и серьезность ошибок, возможности устранения ошибок и восстановления после сбоев, ремонтопригодность, срок службы и т. д.;
- ***производительности*** – накладывают дополнительные ограничения на функциональные требования. Например, возможность параллельного выполнения операций или требования, задающие частоту, скорость, время отклика, выделяемый объем памяти и т. д. для выполнения конкретных операций;
- ***возможности поддержки*** – определяют порядок консультаций пользователей в процессе эксплуатации системы, распространения новых версий системы и документации к ней, обновления централизованно распространяемой нормативно-справочной информации и т.д.


Снять деньги со счета
Клиент банка
Внести деньги на счет
Перечислить деньги на другой счет


Модель вариантов использования
Модель анализа
«трассируется»
Снять деньги со счета
Снять деньги со счета
Участник
Устройство выдачи
Интерфейс кассира
Получение
Банковский счет

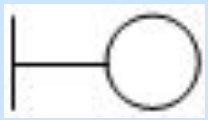
Реализация каждого из вариантов использования в структуре классов анализа



Три вида классов анализа

* граничный;
* управляющий;
* сущности.

Графический стереотип

Граничный класс	Управляющий класс	Класс сущности
		
Диалоговое окно «Нормативы»	Расчет Vдоп	План

Стандартное обозначение со строкой-стереотипом

Граничный класс	Управляющий класс	Класс сущности
«boundary» Диалоговое окно «Нормативы»	«control» Расчет Vдоп	«entity» План

граничный класс

* используется для моделирования взаимодействия между системой и актерами (пользователями, внешними системами или устройствами). Взаимодействие часто включает в себя получение или передачу информации, запросы на предоставление услуг и т. д.
* Граничные классы являются абстракциями диалоговых окон, форм, панелей, коммуникационных интерфейсов, интерфейсов периферийных устройств, интерфейсов API (англ. application program interface – интерфейс прикладных программ) и т. д. Каждый граничный класс должен быть связан как минимум с одним актером;

управляющий класс

* отвечает за координацию, взаимодействие и управление другими объектами, выполняет сложные вычисления, управляет безопасностью, транзакциями и т. п.

класс сущности

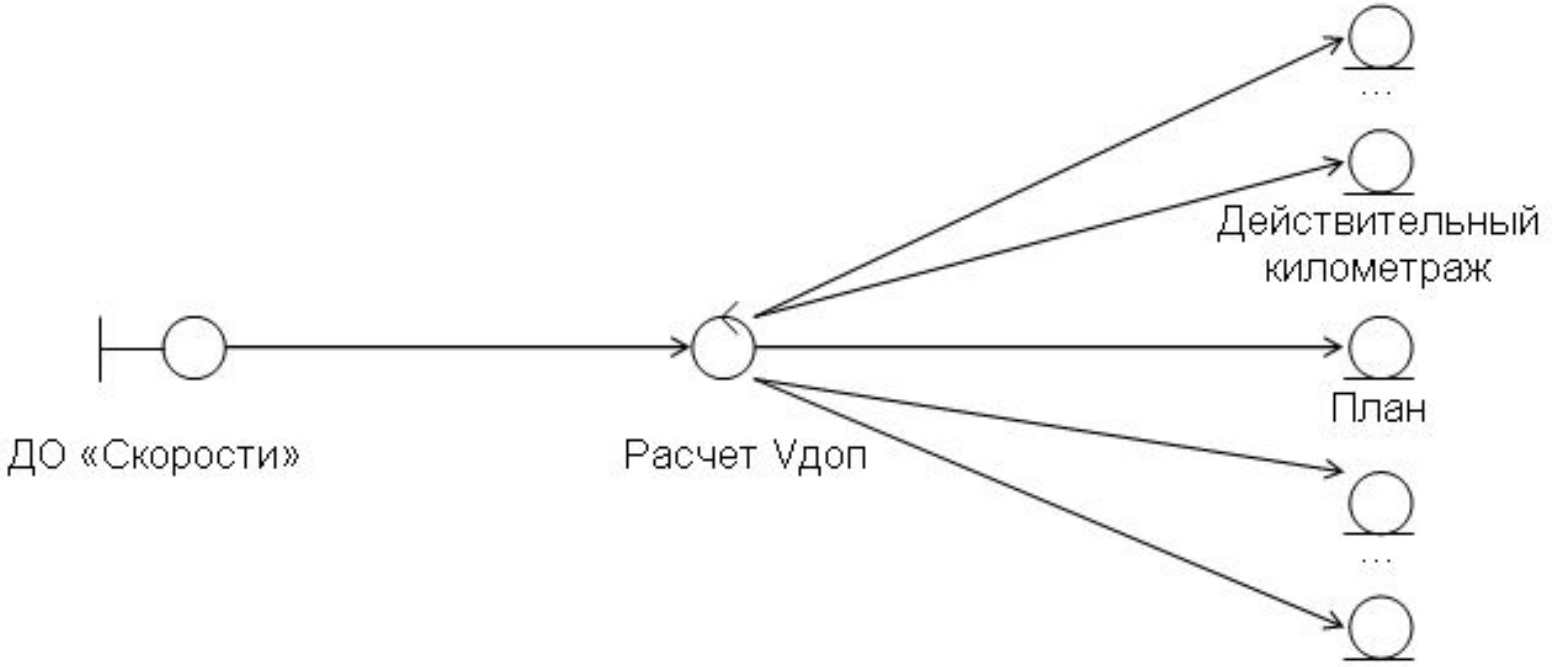
* используется для моделирования долгоживущей, нередко сохраняемой информации. Классы сущности являются абстракциями основных понятий предметной области – людей, объектов, документов и т. д., как правило, хранимых в табличном или ином виде.

Отношения

- ассоциаций;
- агрегаций;
- композиций;
- обобщения;
- зависимостей.

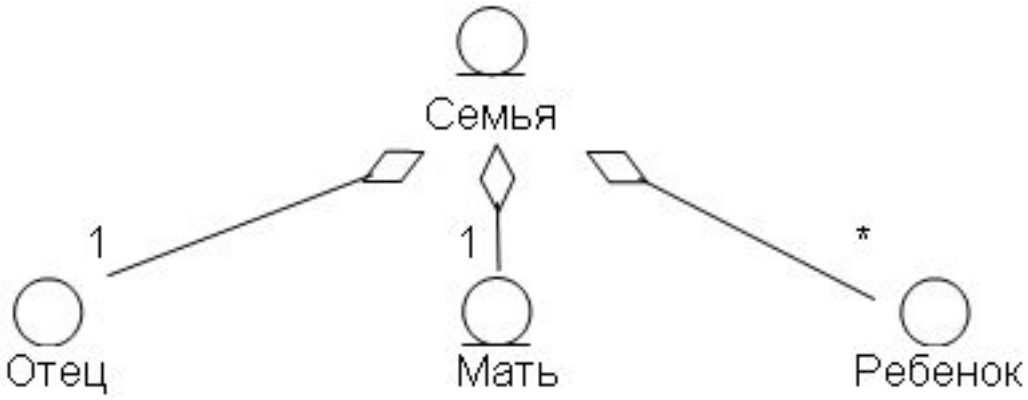
Отношение ассоциации

применительно к диаграмме классов анализа показывает, что объекты одного класса содержат информацию о существовании (наличии в памяти) объектов другого класса и между ними имеется некоторая логическая или семантическая связь.



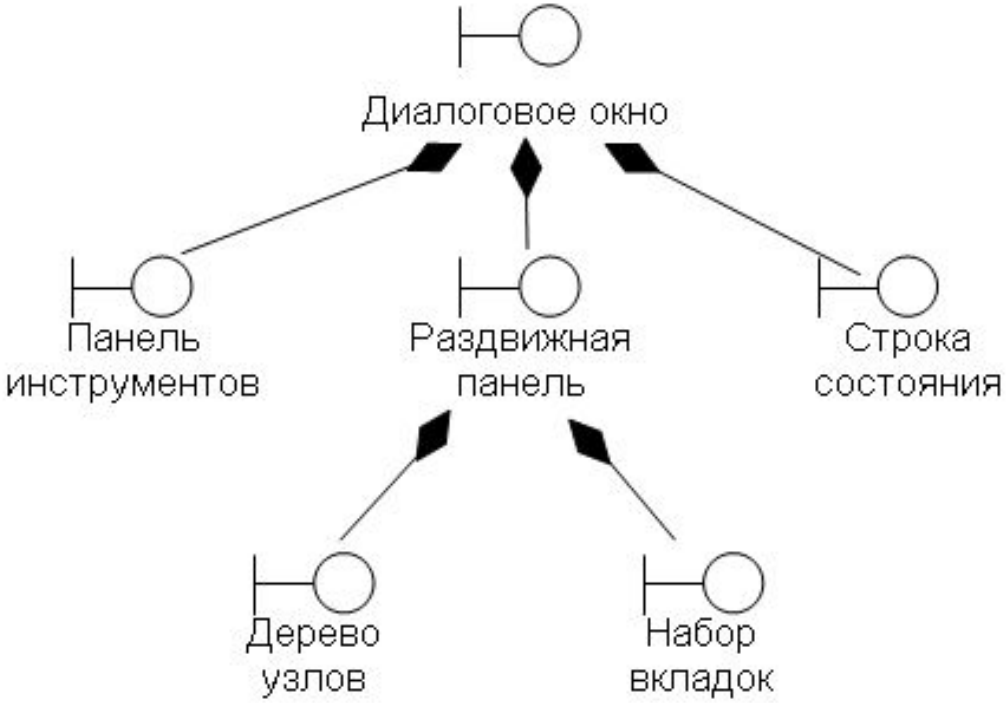
Отношение агрегации

указывает на отношение «часть–целое». отношение, как и ассоциация, означает, что «объект–целое» содержит ссылку на «объект–часть». «Объект–часть» также может содержать ссылку на «объект–целое». Агрегация может указываться только между классами одного типа.



Отношение композиции

аналогично агрегации, в которой «части» не могут существовать отдельно от «целого». Применительно к классам (объектам) это означает, что при уничтожении «объекта–целого» должны быть уничтожены все связанные с ним «объекты–части». При этом допускается создание «объектов–частей» намного позже или уничтожение намного ранее «объекта–целого».

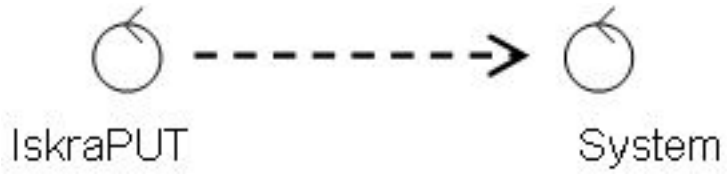


Отношение обобщения

является обычным таксонометрическим отношением между более общим (абстрактным) классом (родителем или предком) и его частным случаем (дочерним классом или потомком). Отношение обобщения может быть только между классами одного вида.

Отношение зависимости

Отношение зависимости применительно к диаграмме классов анализа означает, что в спецификации или теле методов объектов одного класса (зависимого) выполняется обращение к атрибутам, методам или непосредственно к объектам другого класса (независимого).



1. При выделении классов анализа следует учитывать тот факт, что они являются обобщенными (укрупненными) сущностями, которые в дальнейшем подлежат уточнению и возможному разбиению на несколько более мелких классов.

2. Для выделения классов сущностей необходимо определить все реальные либо воображаемые объекты, имеющие существенное значение для рассматриваемой предметной области, информация о которых подлежит хранению. При этом из спецификаций вариантов использования следует выделить все объекты, которые могут существовать независимо от других. (объект «билет» является независимой сущностью, потому что любой билет существует независимо от того, знаем мы его номер, стоимость или нет.) Т. е. при выделении классов-сущностей действуют те же правила, что при построении концептуальной модели БД).

* 3. Для каждого актера следует предусмотреть, как минимум, один граничный класс в целях организации интерфейса между ним и системой. Аналогично для каждого класса сущности, как правило, должен быть граничный класс – ведь по каждому объекту класса сущности должна быть предусмотрена возможность просмотра, ввода и/или корректировки информации через определенную форму ввода/вывода или чтения/записи через определенный интерфейс.

* 4. Для управления, обеспечения взаимодействия и координации работы объектов, реализующих одну из функций системы (обычно, вариант использования), необходимо предусмотреть, как минимум, один управляющий класс. Как правило, взаимодействие между граничным классом и классом сущности происходит через управляющий класс.

* 5. В целях облегчения восприятия специфики связей между классами рекомендуется использовать отношения агрегации, композиции и обобщения.

* 6. При разработке диаграммы основное внимание должно быть уделено определению и детализации классов сущностей, управляющих и граничных классов, обеспечивающих взаимодействие с внешними системами. Граничные классы, обеспечивающие взаимодействие с пользователями, не требуют излишней детализации до уровня отдельного поля ввода или ниспадающего списка, так как современные среды программирования обладают богатыми возможностями по быстрому созданию пользовательского интерфейса.

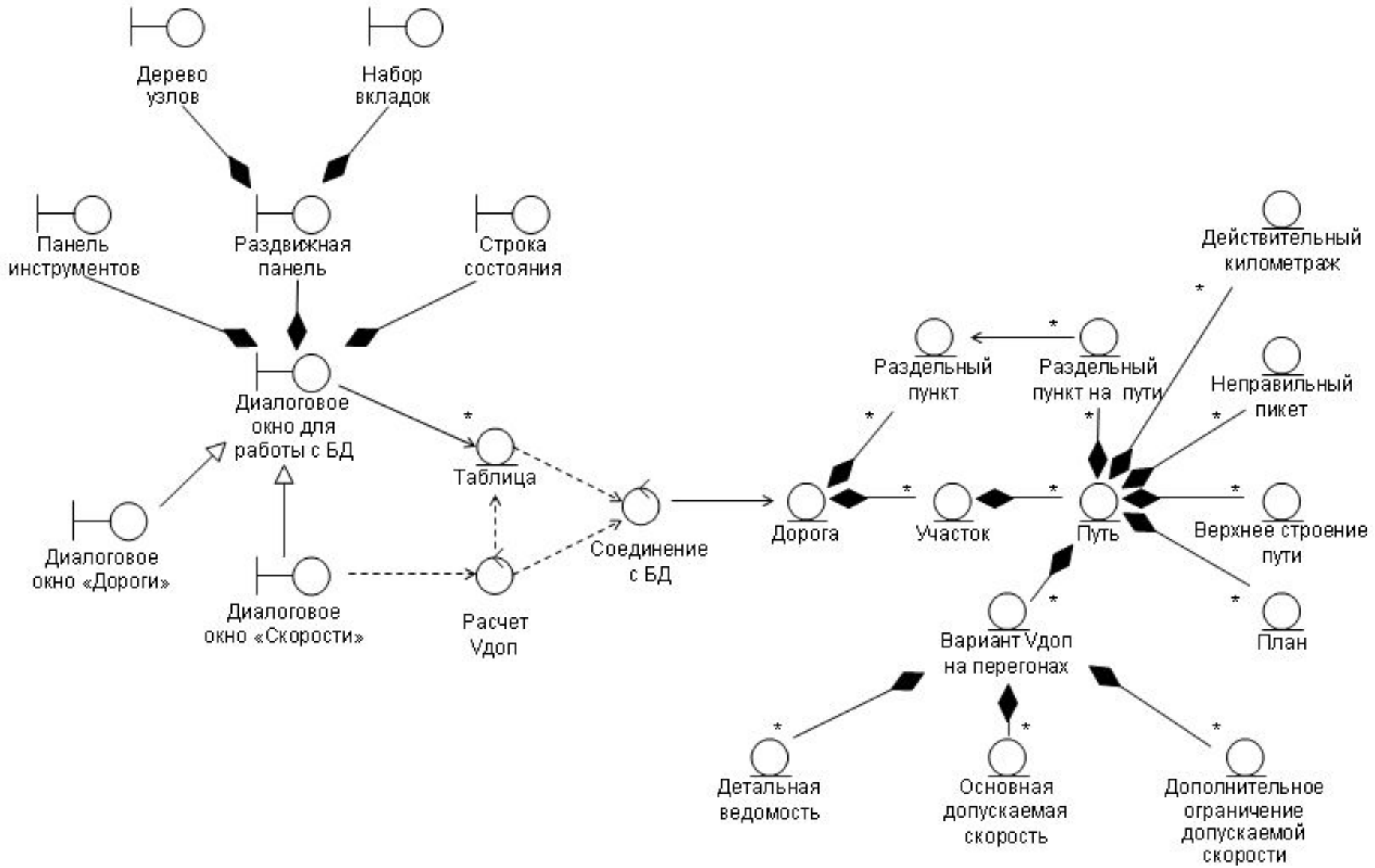

Дерево узлов
Набор вкладок
Панель инструментов
Раздвижная панель
Строка состояния
Диалоговое окно для работы с БД
Таблица
Диалоговое окно «Дороги»
Диалоговое окно «Скорости»
Расчет Удоп
Соединение с БД
Дорога
Раздельный пункт
Раздельный пункт на пути
Участок
Путь
Действительный километраж
Неправильный пикет
Верхнее строение пути
План
Вариант Удоп на перегонах
Детальная ведомость
Основная допускаемая скорость
Дополнительное ограничение допускаемой скорости

Диаграмма кооперации

· экземпляры актеров и классов, участвующих в реализации варианта использования;

· ассоциации между экземплярами актеров и классов;

· сообщения, передаваемые между экземплярами актеров и классов.

Имя объекта : Имя класса	**Вася : Программист**	
: Имя класса	: Программист	анонимный объект
Имя объекта	Вася	имя класса известно
Имя объекта :	Вася :	объект-сирота. Считается, что имя класса неизвестно

Взаимодействие

* ***Сообщение*** (англ. message) – это спецификация факта передачи информации между сущностями с ожиданием выполнения определенных действий со стороны принимающей сущности. Сущность, отправляющую сообщение, называют ***клиентом***, а принимающую – ***сервером***.

Сообщения

	**синхронное сообщение**
	асинхронное сообщение
	возвращающее сообщение (возврат управления)

Спецификация сообщения

* · предшествующие сообщения / [сторожевое условие] номер сообщения : стереотип;
* · предшествующие сообщения / [сторожевое условие] номер сообщения : переменная := имя сообщения (список аргументов)

* ***Предшествующие сообщения*** (их номера или идентификаторы) записываются через запятые и указывают, что данное сообщение не может быть передано, пока не будут посланы все предшествующие сообщения своим адресатам.
* ***Сторожевое условие*** – обычное булевское выражение, означающее возможность посылки сообщения. Используется для ветвления потока сообщений.
* ***Порядковый номер*** указывает на последовательность посылки сообщений. Например, {1, 2, 3, 3.1, 3.2, 3.3, 4, 5}. Сообщения с номерами {1, 2, 3, 4, 5} посылаются объектом, инициализирующим взаимодействие, а сообщения {3.1, 3.2, 3.3} – другим объектом, после получения им сообщения с номером 3.

Стандартные стереотипы

* · «call» (англ. – вызвать) – синхронное сообщение, требующее выполнения операции принимающего объекта;
* · «create» (англ. – создать) – синхронное сообщение, требующее создания объекта;
* · «destroy» (англ. – уничтожить) – синхронное сообщение с требованием уничтожить соответствующий объект;
* · «send» (англ. – послать) – асинхронное сообщение, обозначающее посылку сигнала серверу;
* · «return» (англ. – возвратить) – возвращающее сообщение.

* ***Переменная (атрибут)***, которая будет содержать значение, возвращаемое в результате обработки сообщения.
* ***Имя сообщения*** (обязательный параметр) – имя вызываемой операции объекта-получателя.
* ***Список аргументов*** – список аргументов, разделенных запятыми и передаваемых для выполнения операции.

Диаграмма кооперации