Анализ и проектирование на UML

Направление подготовки
«Информационные системы и технологии»

Максим Валерьевич Хлопотов,
старший преподаватель кафедры ИС

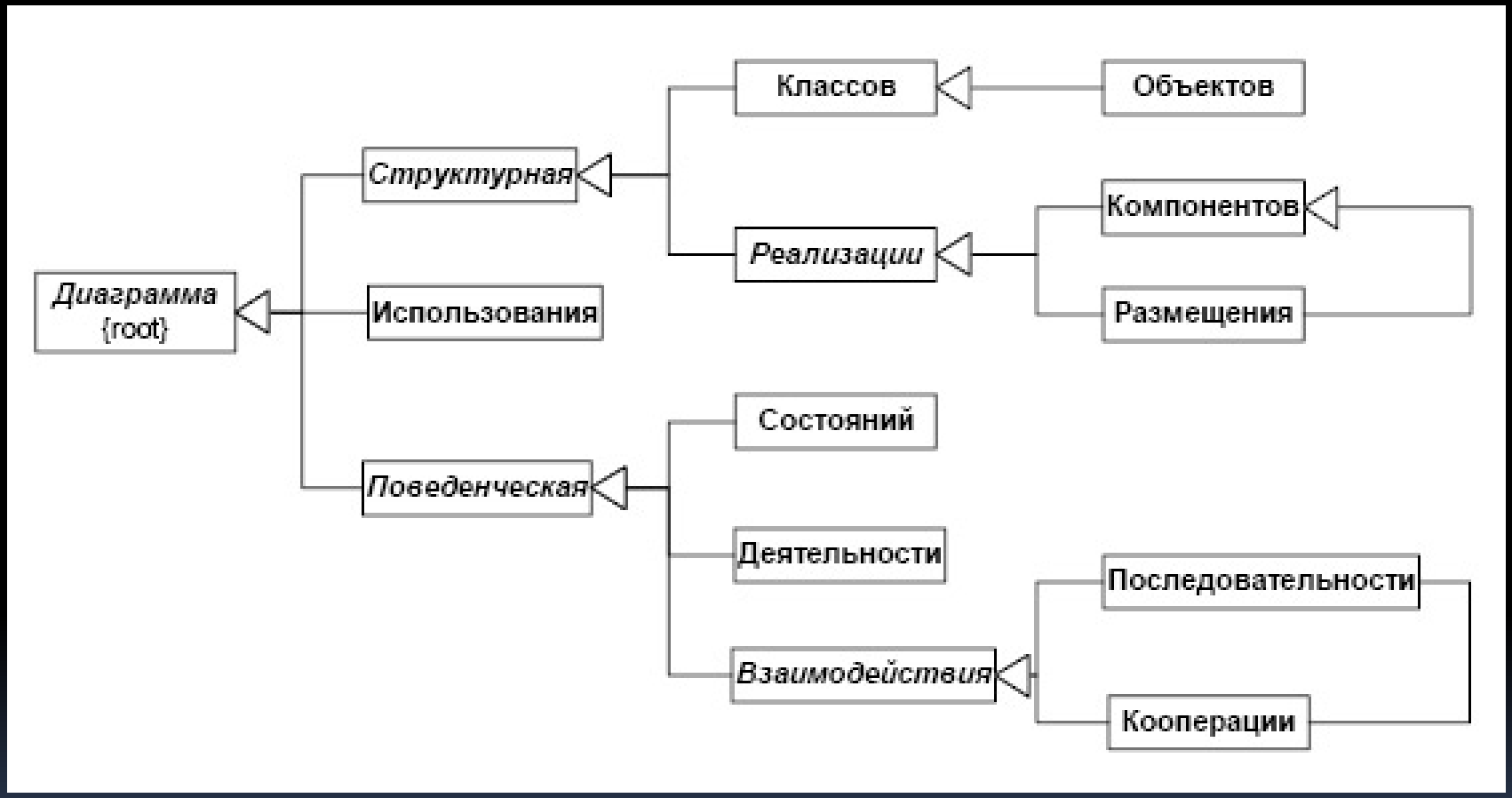
Темы лекционных занятий

1. Введение в UML
2. **Моделирование использования**
3. Моделирование структуры
4. Моделирование поведения
5. Дисциплина моделирования

Назначение UML

UML — это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем.

Иерархия диаграмм UML



Представления

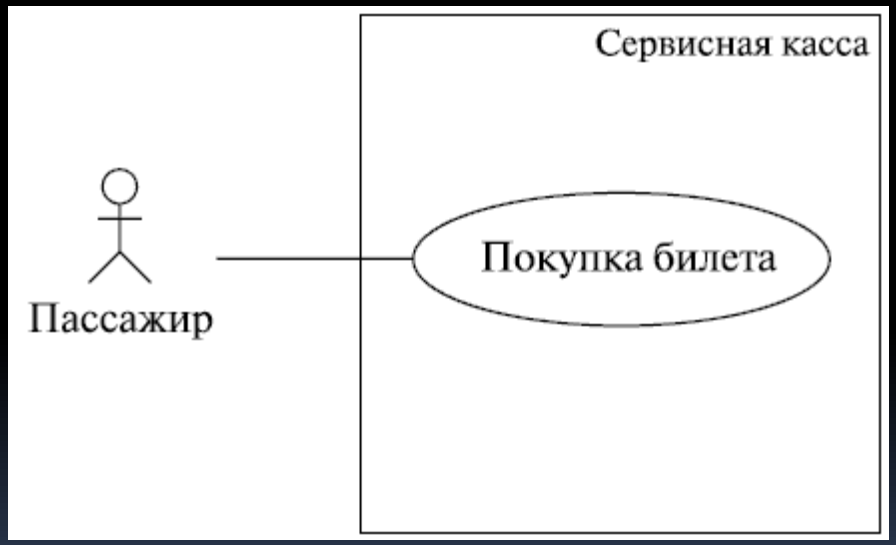
Выделим три представления:
- представление использования (что делает система полезного?);
- представление структуры (из чего состоит система?);
- представление поведения (как работает система?).

Диаграммы UML

Диаграмма использования — это наиболее общее представление функционального назначения системы. Диаграмма использования призвана ответить на главный вопрос моделирования: что делает система во внешнем мире?

Пример

действующее лицо (эктор), ассоциация, вариант использования (прецедент), рамки системы



Диаграммы UML

Диаграмма классов — основной способ описания структуры системы. Это не удивительно, поскольку UML сильно объектно-ориентированный язык, и классы являются основным "строительным материалом" системы.

Пример

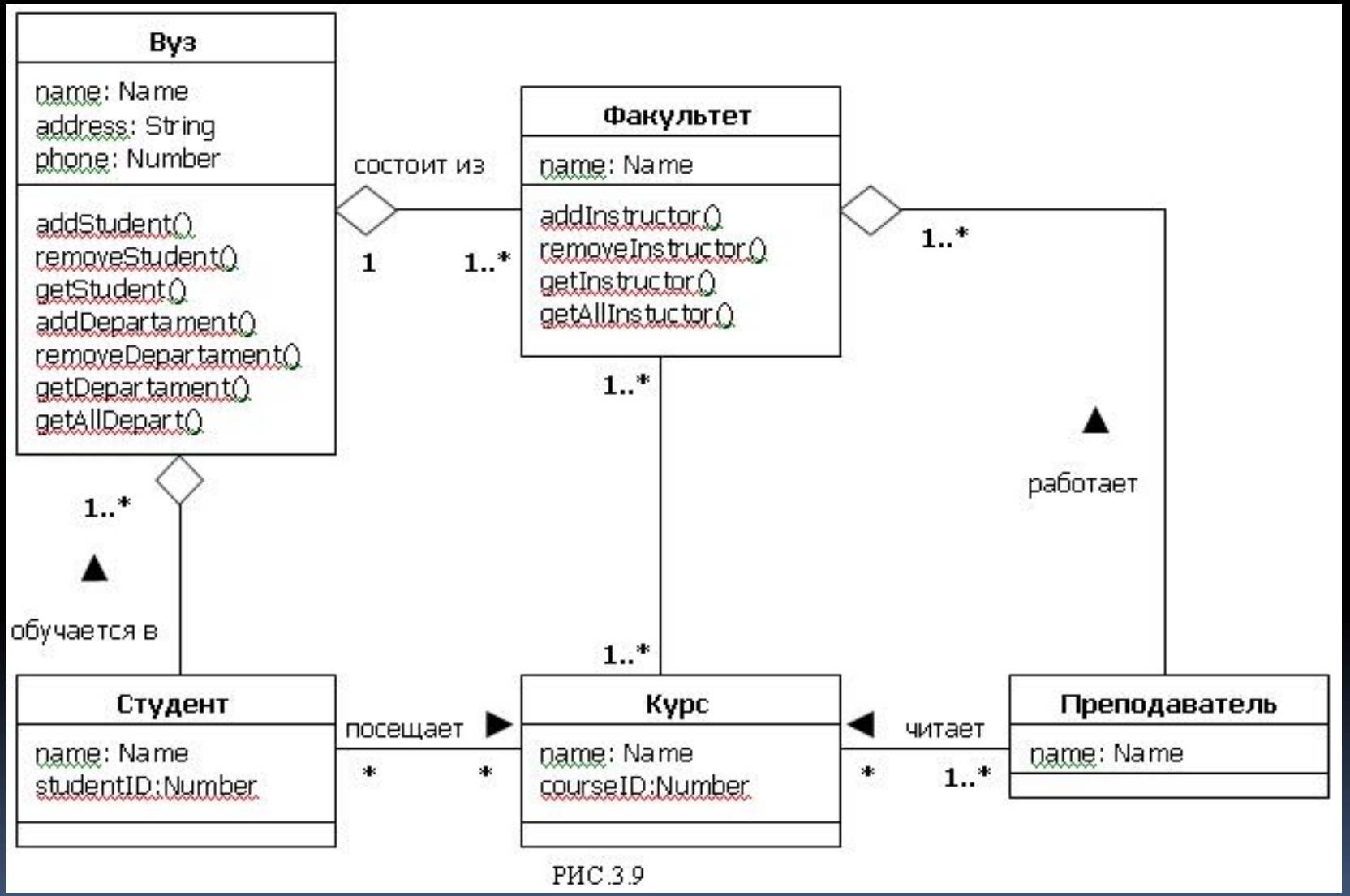


РИС 3.9

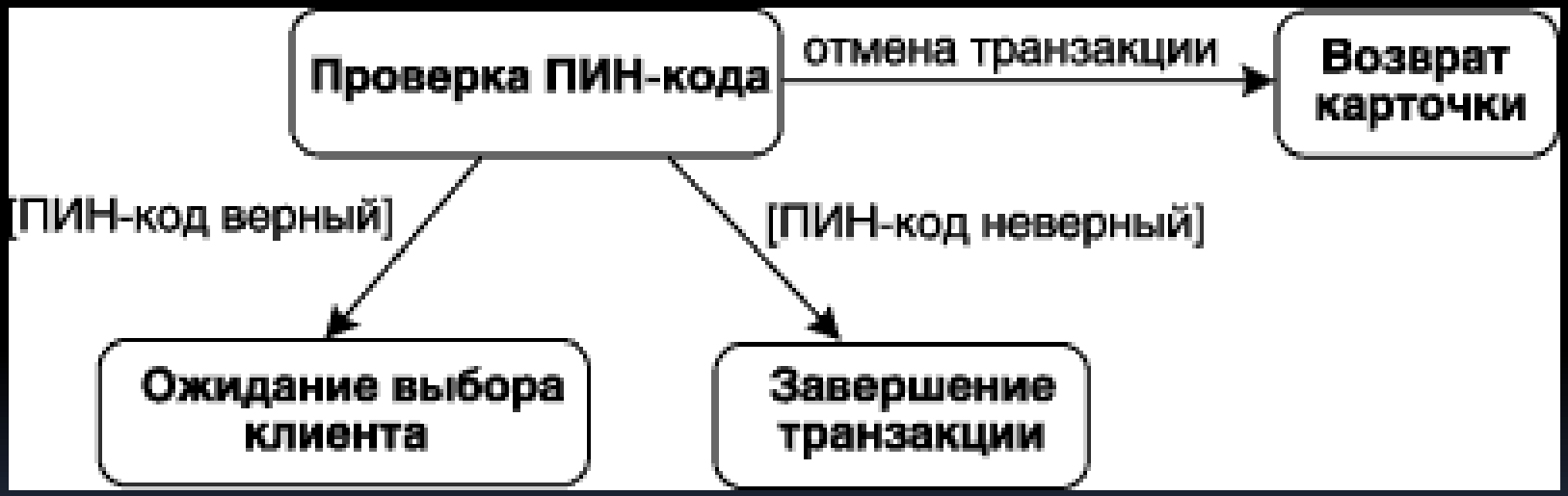
Диаграммы UML

Диаграмма объектов — это частный случай диаграммы классов. Диаграммы объектов имеют вспомогательный характер — по сути это примеры, показывающие, какие имеются объекты и связи между ними в некоторый конкретный момент функционирования системы.

Диаграммы UML

Диаграмма состояний — это основной способ детального описания поведения в UML. В сущности, диаграммы состояний представляют собой граф состояний и переходов конечного автомата, нагруженный множеством дополнительных деталей и подробностей.

Пример
(изменение состояний банкомата при проверке ПИН-кода)



Диаграммы UML

Диаграмма деятельности — это, фактически, блок-схема алгоритма, в которой модернизированы обозначения, а семантика согласована с современным объектно-ориентированным подходом.

Пример

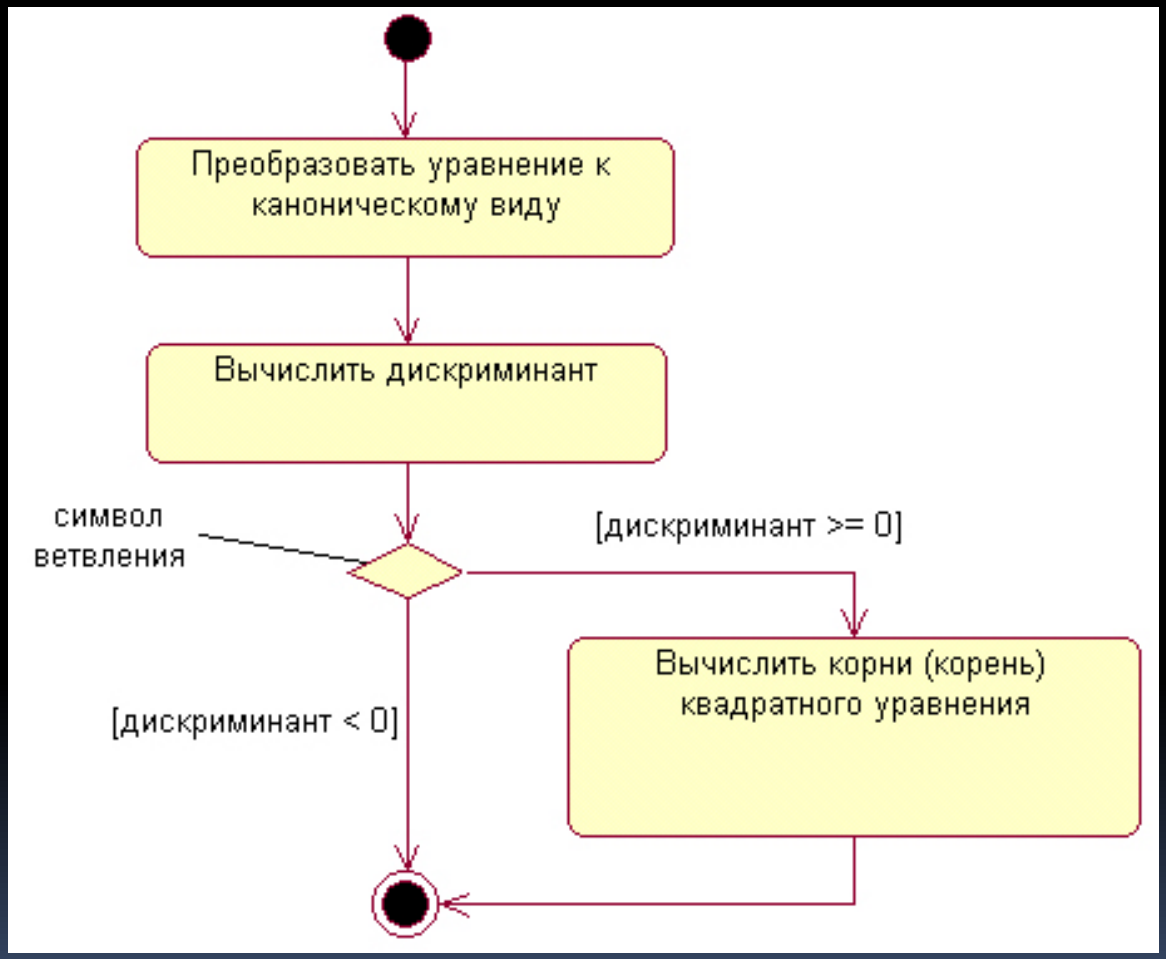
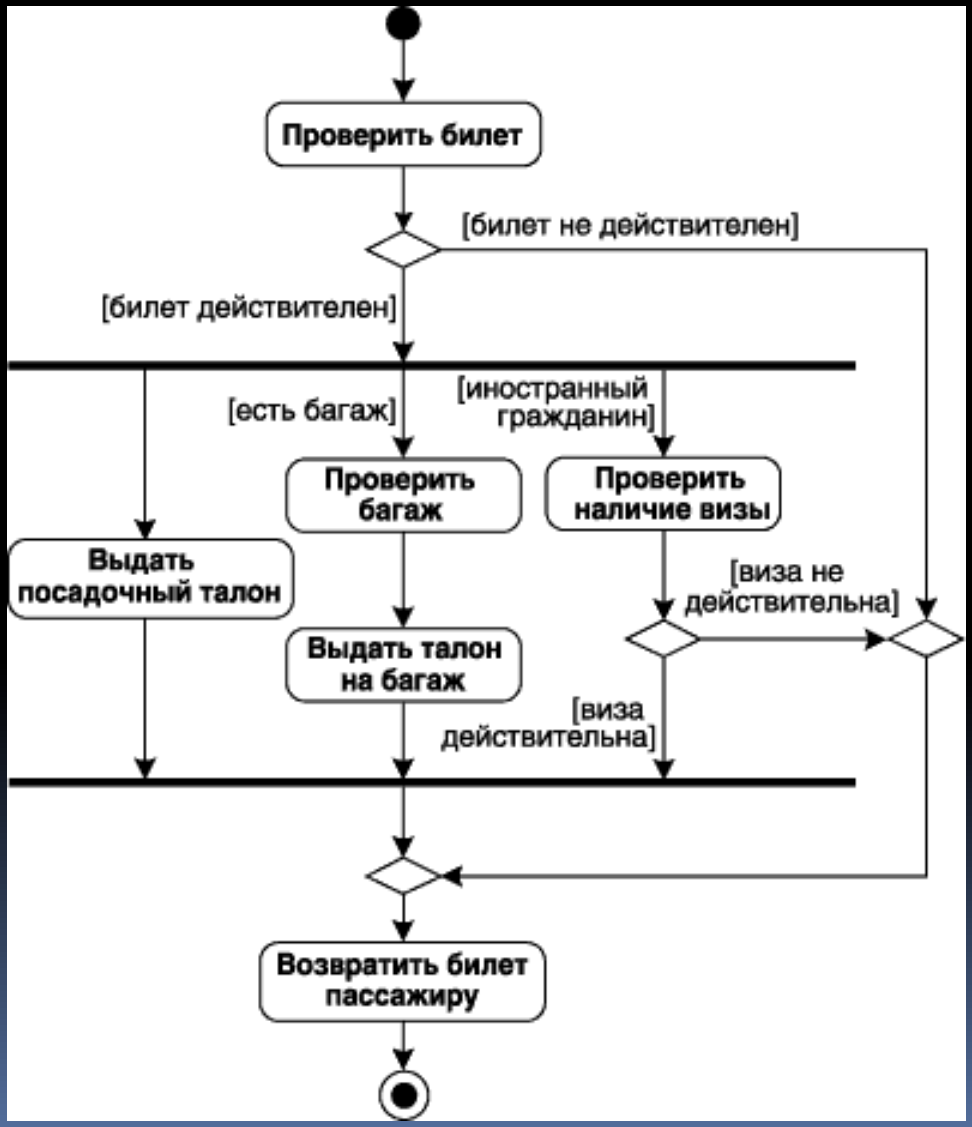

Преобразовать уравнение к каноническому виду
Вычислить дискриминант
символ ветвления
[дискриминант >= 0]
Вычислить корни (корень) квадратного уравнения
[дискриминант < 0]

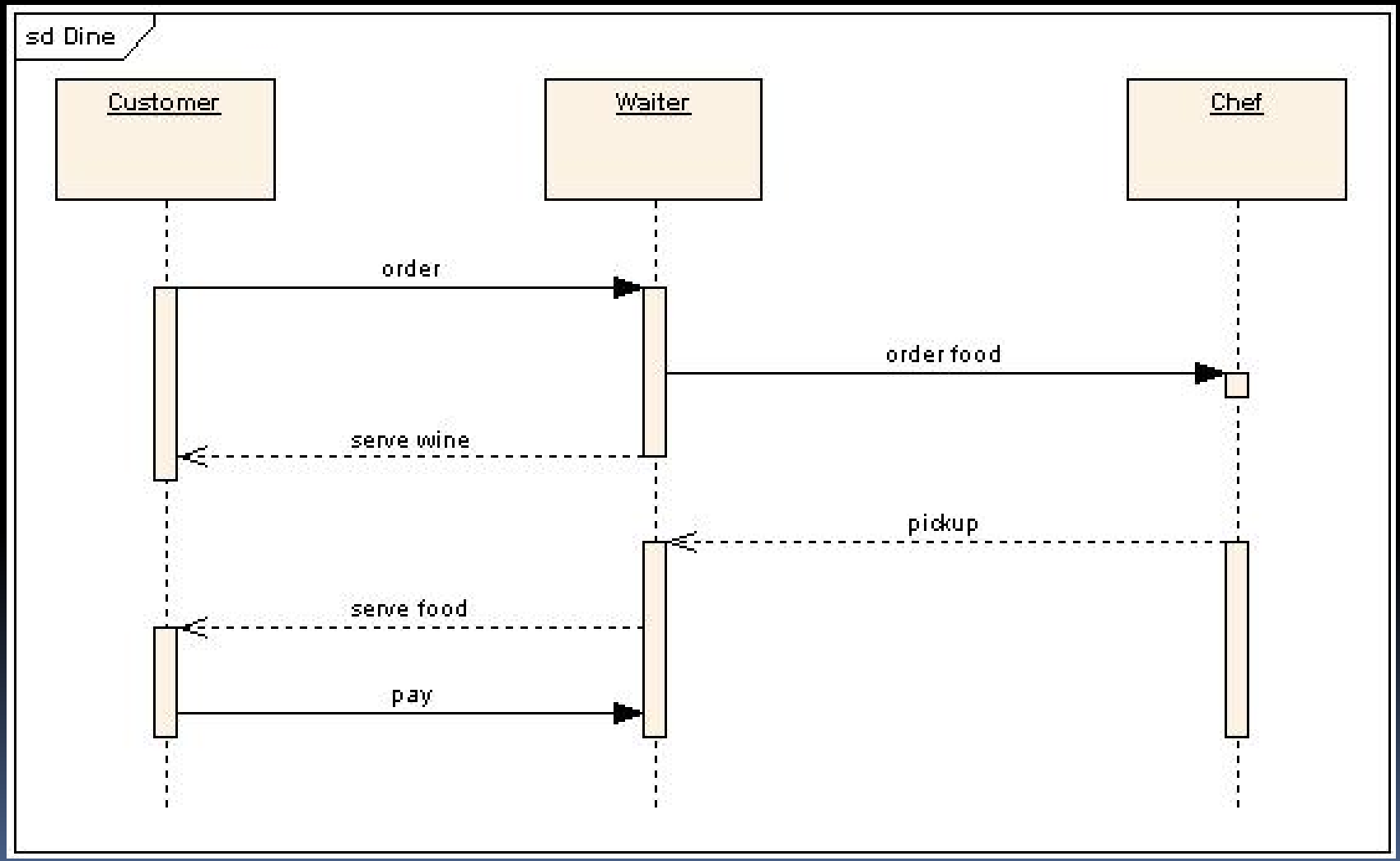
Диаграмма деятельности (пример)



Диаграммы UML

Диаграмма последовательности — это способ описать поведение системы "на примерах". Фактически, диаграмма последовательности — это запись протокола конкретного сеанса работы системы (или фрагмента такого протокола). В объектно-ориентированном программировании самым существенным во время выполнения является посылка сообщений взаимодействующими объектами.

Пример


sd Dine
Customer
Waiter
Chef
order
order food
serve wine
pickup
serve food
pay

Диаграммы UML

Диаграмма кооперации (в UML 2 – диаграмма коммуникации) семантически эквивалентна диаграмме последовательности.
Фактически, это такое же описание последовательности обмена сообщениями взаимодействующих объектов, только выраженное другими графическими средствами.

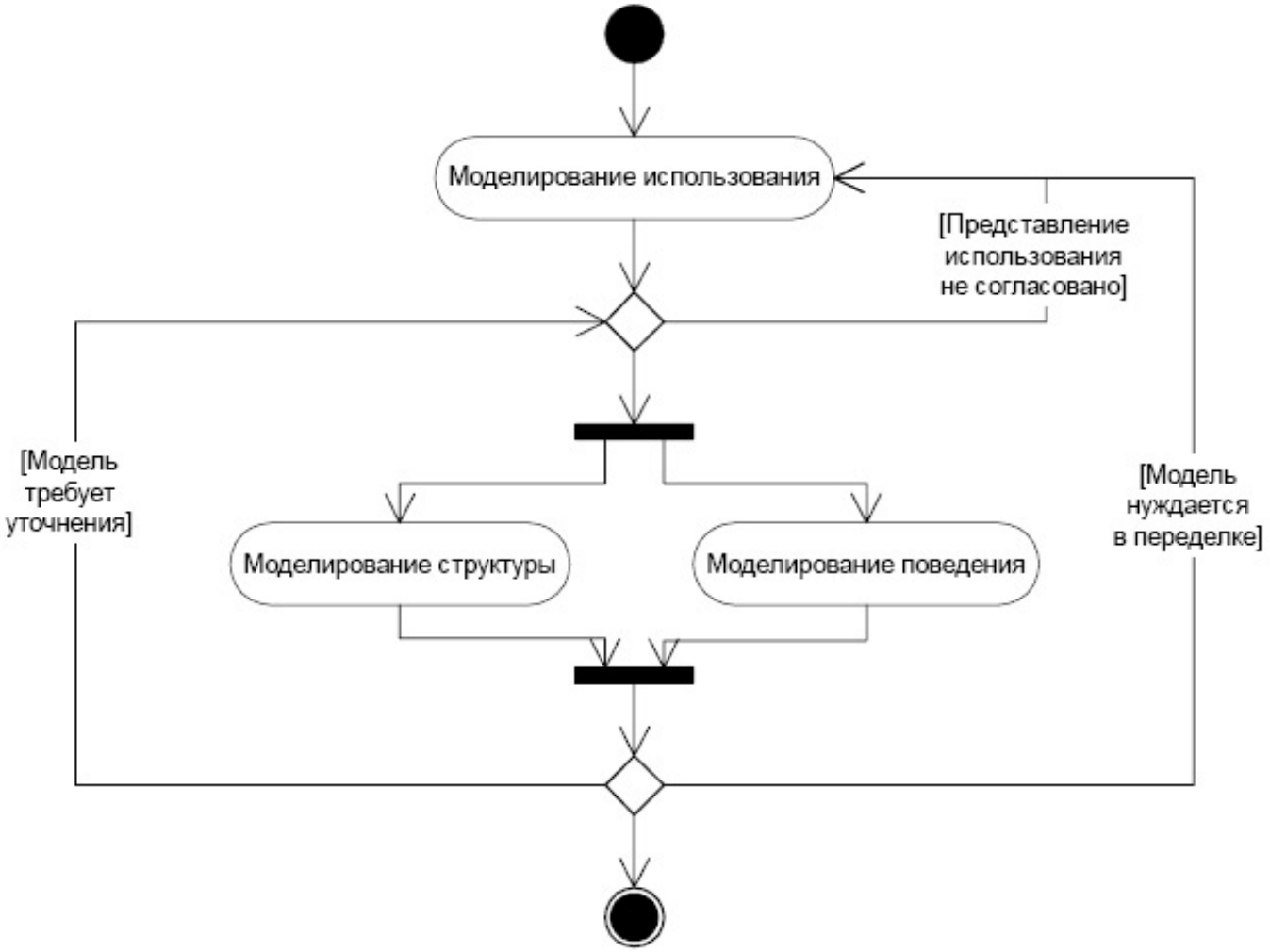
Диаграммы UML

Диаграмма компонентов — это, фактически, список артефактов, из которых состоит моделируемая система, с указанием некоторых отношений между артефактами. Наиболее существенным типом артефактов программных систем являются программы. Таким образом, на диаграмме компонентов основной тип сущностей — это компоненты (как исполнимые модули, так и другие артефакты), а также интерфейсы (чтобы указывать взаимосвязь между компонентами) и объекты (входящие в состав компонентов).

Диаграммы UML

Диаграмма размещения (диаграмма развёртывания) немногим отличается от диаграммы компонентов. Фактически, наряду с отображением состава и связей компонентов здесь показывается, как физически размещены компоненты на вычислительных ресурсах во время выполнения.

Процесс моделирования



Моделирование использования

Наш язык и мышление устроены так, что самой простой, понятной и четкой формой изложения мыслей являются так называемые простые утверждения.

Простое утверждение имеет следующую грамматическую форму: подлежащее — сказуемое — прямое дополнение. В логических терминах: субъект — предикат — объект.

Например: начальник увольняет сотрудника, директор создает отдел.

Моделирование использования

По сути, именно простые утверждения и записаны на диаграмме использования.

Действующее лицо — это субъект, а вариант использования — предикат (вместе с объектом).

Моделирование использования предполагает явное формулирование требований к системе на самом начальном этапе разработки.

Преимущества

- **Простые утверждения.**

Моделирование использования позволяет записать исходное техническое задание в строгой и формальной, но в тоже время очень простой и наглядной графической форме, как совокупность простых утверждений относительно того, что делает система для пользователей.

Использование такой формы не гарантирует от ошибок (вряд ли гарантия от ошибок вообще возможна), но благодаря простоте и наглядности формы их легче заметить.

Преимущества

- **Абстрагирование от реализации.**

Моделирование использования предполагает формулирование требований к системе абсолютно независимо от ее реализации.

Представление использования описывает только, что делает система, но не как это делается и не зачем это нужно делать.

Преимущества

- **Декларативное описание.**

Каждый вариант использования описывает (именует) некоторое множество последовательностей действий, доставляющих значимый для пользователя результат.

Никакого императивного описания представление использования не содержит, в модели нет указаний на то, какой вариант использования должен выполняться раньше, а какой позже, то есть нет описания алгоритма, а значит, нет алгоритмических ошибок.

Преимущества

- **Выявление границ.**

Представление использование определяет границы системы и постулирует существование во внешнем мире использующих её агентов.

Описание системы в виде черного ящика с определенными интерфейсами кажется очень похожим на представление использования, но здесь есть важное различие, которое часто упускается из вида. На диаграмме использования одинокие и покинутые действующие лица и варианты использования обнаруживаются с первого взгляда.

Диаграмма использования

Диаграмма использования является основным средством моделирования использования в UML.

На диаграмме использования применяются следующие типы сущностей:

- действующие лица;
- варианты использования;
- примечания;
- пакеты.

Диаграмма использования

Между этими сущностями устанавливаются следующие типы отношений:

- ассоциация между действующим лицом и вариантом использования;
- обобщение между действующими лицами;
- обобщение между вариантами использования;
- зависимости (двух стереотипов) между вариантами использования;
- зависимости между пакетами.

Диаграмма использования

С синтаксической точки зрения *действующее лицо* — это стереотип классификатора, который обозначается специальным значком. Для действующего лица указывается только имя, идентифицирующее его в системе. Семантически действующее лицо — это множество логически взаимосвязанных ролей.

Роль в UML — это интерфейс, поддерживаемый данным классификатором в данной ассоциации.

С прагматической точки зрения главным является то, что действующие лица находятся вне проектируемой системы (или рассматриваемой части системы).

Действующие лица

Вопрос о выделении действующих лиц при составлении модели — один из самых сложных.

Неудачный выбор действующих лиц может отрицательно повлиять на всю модель в целом.

Формального метода идентификации действующих лиц не существует.

Действующие лица

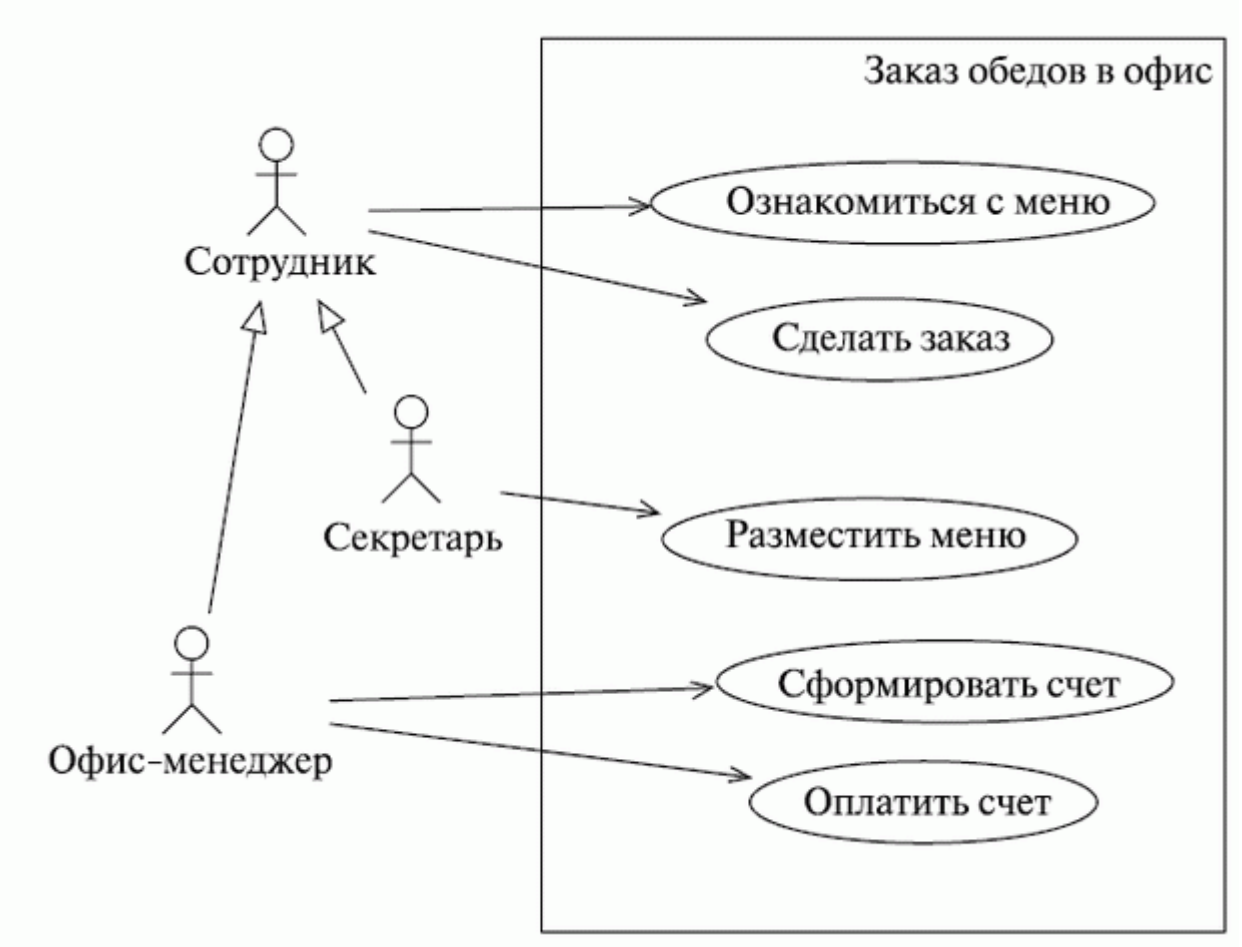
Выделение категорий пользователей происходит, как правило, неформально: из соображений здравого смысла и собственного опыта.

Имеет смысл отнести пользователей к разным категориям, если наблюдаются следующие признаки: пользователи участвуют в разных (независимых) бизнес-процессах; пользователи имеют различные права на выполнение действий и доступ к информации; пользователи взаимодействуют с системой в разных режимах: от случая к случаю, регулярно, постоянно.

Действующие лица

В качестве имен действующих лиц рекомендуется использовать существительное (возможно с определяющим словом), а в качестве имен вариантов использования — глагол (возможно, с дополнением).

Пример нотации



Варианты использования

Выделение вариантов использования — ключ ко всему дальнейшему моделированию. На этом этапе определяется функциональность системы, то есть, что она должна делать.

Нотация для варианта использования — это просто имя, помещенное в овал (или помещенное под овалом — такой вариант тоже допустим).

Другими словами, функции, выполняемые системой, на уровне моделирования использования никак не раскрываются — им только даются имена.

Варианты использования

Семантически вариант использования — это описание множества возможных последовательностей действий (событий), приводящих к значимому для действующего лица результату.

Прагматика варианта использования состоит в том, что среди всех последовательностей действий, могущих произойти при работе приложения, выделяются такие, в результате которых получается явно видимый и достаточно важный для действующего лица результат.

Варианты использования

Выбор вариантов использования сильно влияет на качество модели. Формальные методы выбора предложить трудно — помогают только опыт и чутьё.

Некоторые пункты ТЗ естественным образом переводятся в варианты использования.

Примечание

Примечания можно и нужно употреблять на всех типах диаграмм, а не только на диаграммах использования.

При моделировании проектировщик часто может сказать о моделируемой системе больше, чем это позволяют сделать строгая, но ограниченная нотация UML. В таких случаях наиболее подходящим средством для внесения в модель дополнительной информации является примечание.

Примечание

Примечание имеет свою графическую нотацию — прямоугольник с загнутым уголком, к котором находится текст примечания.

Примечания могут находиться в отношении соответствия с другими сущностями — эти отношения изображаются пунктирной линией без стрелок.

Примечания содержат текст, который вводит пользователь — создатель модели.

Примечание

Примечания могут иметь стереотипы. В UML определены два стандартных стереотипа для примечаний:

- requirement — описывает общее требование к системе;
- responsibility — описывает ответственность класса.

Примечания первого типа часто применяют в диаграммах использования, а примечания второго типа — в диаграммах классов.

Отношения

На диаграммах использования применяются следующие основные типы отношений:

- ассоциация между действующим лицом и вариантом использования;
- обобщение между действующими лицами;
- обобщение между вариантами использования;
- зависимости между вариантами использования.

Ассоциация

Ассоциация между действующим лицом и вариантом использования показывает, что действующее лицо тем или иным способом взаимодействует (предоставляет исходные данные, потребляет результат) с вариантом использования.

Ассоциация обозначает, что действующее лицо так или иначе, но обязательно непосредственно участвует в выполнении каждого из сценариев, описываемых вариантом использования.

Ассоциация

Ассоциация является наиболее важным и, фактически, обязательным отношением на диаграмме использования.

Если на диаграмме использования нет ассоциаций между действующими лицами и вариантами использования, то это означает, что система не взаимодействует с внешним миром.

Такие системы, равно как и их модели, не имеют практического смысла.

Обобщение

Обобщение между действующими лицами показывает, что одно действующее лицо наследует все свойства (в частности, участие в ассоциациях) другого действующего лица.

С помощью обобщения между действующими лицами легко показать иерархию категорий пользователей системы, в частности, иерархию прав доступа к выполняемым функциям и хранимым данным.

Обобщение

Обобщение между вариантами использования показывает, что один вариант использования является частным случаем (подмножеством множества сценариев) другого варианта использования.

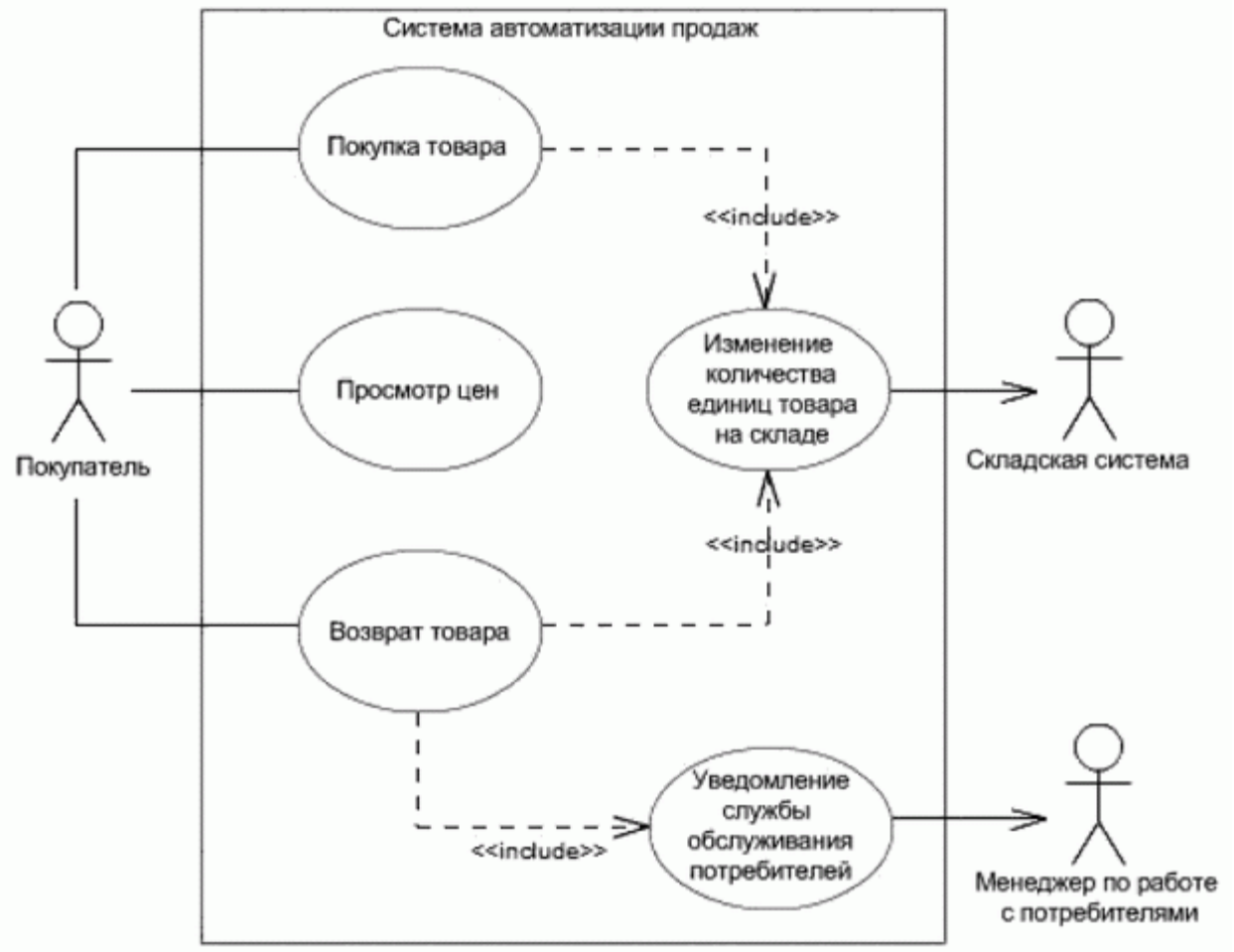
Зависимости

Зависимость между вариантами использования показывает, что один вариант использования зависит от другого варианта использования.

2 стандартных стереотипа зависимости:

- include — показывает, что сценарий независимого варианта использования включает в себя в качестве подпоследовательности действий сценарий зависимого варианта использования;
- extend — показывает, что в сценарий зависимого варианта использования может быть в определенном месте вставлен в качестве подпоследовательности действий сценарий независимого варианта использования.

Пример


Система автоматизации продаж
Покупка товара
<<include>>
Покупатель
Просмотр цен
Изменение количества единиц товара на складе
Складская система
<<include>>
Возврат товара
Уведомление службы обслуживания потребителей
<<include>>
Менеджер по работе с потребителями

Пример


Оплатить покупку
Покупатель
<<extend>>
<<extend>>
<<extend>>
Оплатить наличными
Оформить кредит
Оплатить кредитной картой

Реализация вариантов использования

После того, как построено представление использования, то есть выделены действующие лица, варианты использования и установлены отношения между ними, встает естественный вопрос: что дальше?

Представление использования, если оно тщательно продумано и детально прорисовано, является формой технического задания, содержащей достаточно информации для дальнейшего проектирования.

Реализация вариантов использования

Действующие лица находятся вне системы — с ними ничего делать не нужно.

Таким образом, переход от моделирования использования к другим видам моделирования состоит в уточнении, детализации и конкретизации вариантов использования.

В представлении использования мы показали, что делает система, теперь нужно определить, как это делается. Это обычно называется ***реализацией вариантов использования***.

Реализация вариантов использования

Вариант использования — это описание множества последовательностей действий, доставляющих значимый для действующего лица результат.

Наиболее часто используемый метод описания множества последовательностей действий состоит в указании алгоритма, выполнение которого доставляет последовательность действий из требуемого множества.

Реализация вариантов использования

- текстовые описания;
- псевдокод;
- диаграмма деятельности;
- *диаграммы взаимодействия.*

- Вариант использования должен доставлять значимый результат, значит, если результата нет, то что-то спроектировано не так, как нужно.

Пример текстового описания

Вариант использования «Увольнение по собственному желанию»

1. Сотрудник пишет заявление
2. Начальник подписывает заявление
3. Если есть неиспользованный отпуск, то бухгалтерия рассчитывает компенсацию
4. Бухгалтерия рассчитывает выходное пособие
5. Системный администратор удаляет учетную запись
6. Менеджер штатного расписания обновляет базу данных

Текстовые описания

Достоинства:

- просты, всем понятны, легко и быстро составляются.

Недостатки:

- неполны, неточны, ненаглядны

Псевдокод

- Если программа предназначена для выполнения компьютером, то она должна быть записана на сугубо формальном языке, который называют в этом случае *языком программирования*.
- Если программа предназначена исключительно для чтения и, может быть, выполнения человеком, то можно применить менее формальный (и более удобный) язык, который в этом случае обычно называют *псевдокодом*.

Псевдокод

- Обычно в псевдокод включают смесь общеизвестных ключевых слов языков программирования и неформальные выражения на естественном языке, обозначающие выполняемые действия.
- Эти выражения должны быть понятны человеку, который пишет или читает программу на псевдокоде, но совсем не обязаны быть допустимыми выражениями языка программирования.
- Текст на псевдокоде похож на код программы на языке программирования, но таковым не является.

Псевдокод

Достоинства способа:

- понятен, привычен и доступен любому.

Недостатки:

- плохо согласуется с парадигмой объектно-ориентированного программирования;
- отсутствуют наглядная визуализация, строгость и точность языка проектирования и реализации, поддержка распространенными инструментальными средствами;
- практически невозможно использовать повторно.

Диаграмма деятельности

- Описать алгоритм можно с помощью диаграммы деятельности.
- С одной стороны, диаграмма деятельности — это полноценная диаграмма UML, с другой стороны, диаграмма деятельности немногим отличается от блок-схемы

Диаграмма деятельности

Используются для моделирования процесса выполнения операций.

Частный случай диаграмм состояний.

Диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются *состояния действия* или *деятельности*, а дугами - переходы от одного *состояния действия* к другому.

Состояния деятельности и действия

- Состояние деятельности - состояние в графе деятельности, которое служит для представления процедурной последовательности действий, требующих определенного времени.
- Состояние действия - специальный случай состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом.

Переход

- *Переход* – 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.
- Переход изображается сплошной линией со стрелкой, которая выходит из исходного *состояния* и направлена в целевое *состояние*. Каждый *переход* может быть помечен строкой текста с именем события.

Сторожевое условие

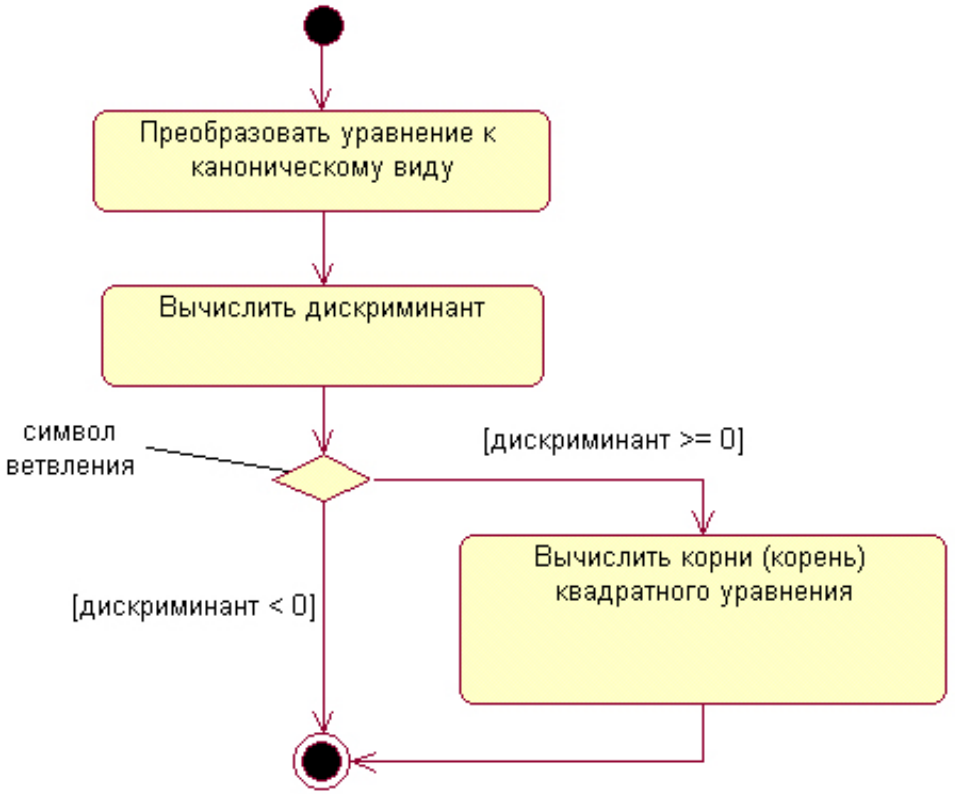
- Логическое условие, записанное в прямых скобках и представляющее собой булевское выражение – называется *сторожевое условие*.
- При этом булевское выражение должно принимать одно из двух взаимно исключающих значений: "истина" или "ложь".

Псевдосостояние

- вершина в графе, которая имеет форму состояния, но не обладает поведением. Примерами псевдосостояний в UML являются начальное и конечное *состояния*.

Ветвление

- Если из состояния действия выходит единственный переход, то его не помечают. Если переходов несколько, для каждого из них должно быть явно записано собственное сторожевое условие.
- Графически ветвление на диаграмме деятельности обозначается символом решения (decision), изображаемого в форме небольшого ромба, внутри которого нет текста



Слияние и разделение

для представления параллельных процессов используется специальный символ для *разделения* и *слияния* параллельных вычислений или потоков управления.

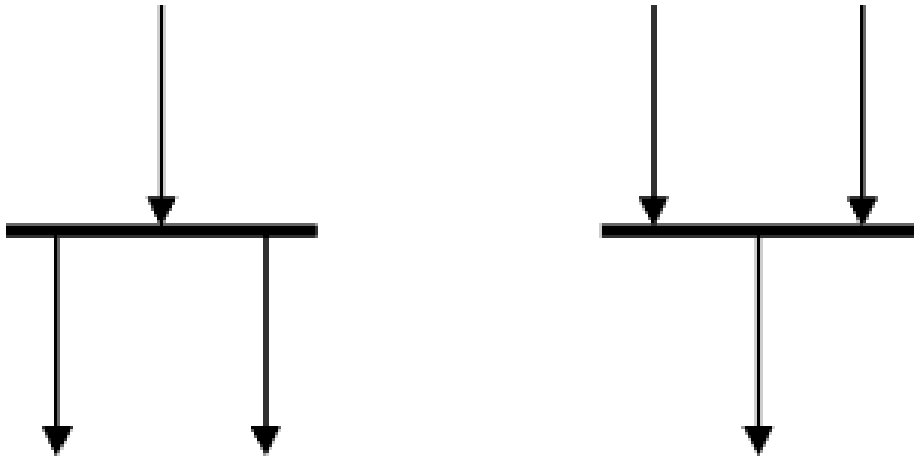


Диаграмма деятельности (пример)

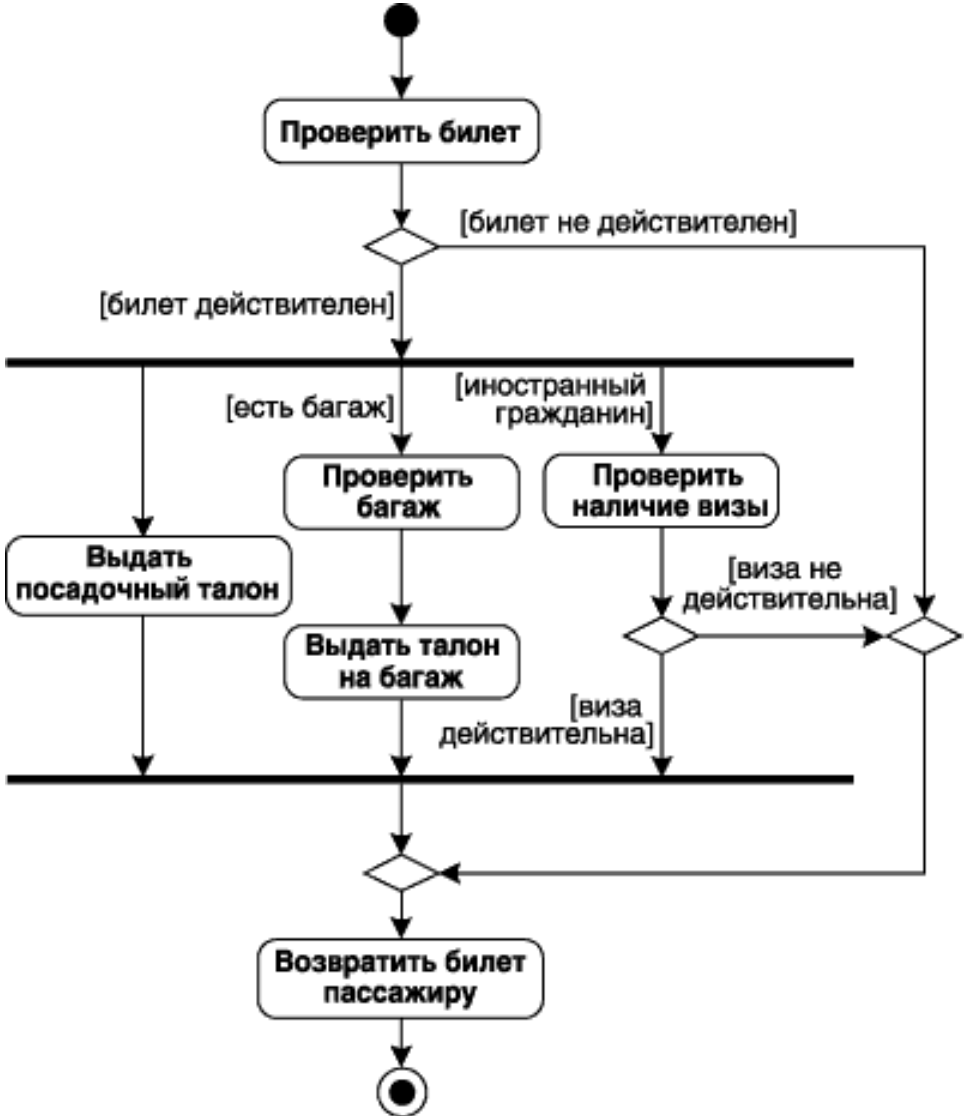


Диаграмма деятельности

- Реализация варианта использования диаграммой деятельности является компромиссным способом ведения разработки — в сущности, это проектирование сверху вниз в терминах и обозначениях UML.
- Эту диаграмму можно показать заказчику, чтобы проверить, действительно ли проектируемая нами логика работы системы соответствует тому бизнес-процессу, который существует в реальности.

Диаграмма деятельности

- Применение диаграмм деятельности для реализации вариантов использования не слишком приближает к появлению целевого артефакта — программного кода, однако может привести к более глубокому пониманию существа задачи и даже открыть неожиданные возможности улучшения приложения, которые было трудно усмотреть в первоначальной постановке задачи.

Выводы

- Составление диаграмм использования — это первый шаг моделирования.
- Основное назначение диаграммы использования — показать, что делает система во внешнем мире.
- Диаграмма использования не зависит от программной реализации системы и поэтому не обязана соответствовать структуре классов, модулей и компонентов системы.
- Идентификация действующих лиц и вариантов использования — ключ к дальнейшему проектированию.
- В зависимости от выбранной парадигмы проектирования и программирования применяются различные способы реализации вариантов использования.