CASE технологии

Лекция 2

Язык UML: предыстория

- середина 1970-х – конец 1980-х годов
  - Появление и расцвет объектно-ориентированного проектирования (ООП)
  - «Война методов» проектирования
- середина 1990-х годов
  - Некоторые методы ООП «закрепились»:
    - Метод Гради Буча (Grady Booch): Booch'91, Booch Lite (позже – Booch'93)
    - Метод Джеймса Румбаха (James Rumbaugh): Object Modeling Technique – OMT (позже – OMT-2)
    - Метод Айвара Джекобсона (Ivar Jacobson): Object-Oriented Software Engineering – OOSE

Язык UML: предыстория

- **метод OOSE** содержал средства **представления вариантов использования**, которые имеют существенное значение на этапе анализа требований в процессе проектирования бизнес-приложений.
- **метод OMT-2** наиболее подходил для **анализа процессов обработки данных** в информационных системах.
- **метод Booch'93** нашел наибольшее применение на **этапах проектирования** и разработки различных программных систем.

Язык UML: предыстория

- октябрь 1994:
  - Гради Буч и Джеймс Румбах (оба - из Rational Software Corporation) начали работу по унификации методов Booch и OMT.
- Цель:
  - полная унификация обоих методов
- Результат:
  - Unified Method, v.0.8 (октябрь 1995)

Язык UML: предыстория

- Октябрь 1995:
  - к работе над UM присоединяется Джекобсон (Objectivity AB)
- Цель:
  - добавление и унификация UM+OOSE

Требования к новому языку моделирования

- Позволять моделировать не только программное обеспечение, но и более широкие классы систем и бизнес-приложений, с использованием объектно-ориентированных понятий.
- Явным образом обеспечивать взаимосвязь между базовыми понятиями для моделей концептуального и физического уровней.
- Обеспечивать масштабируемость моделей, что является важной особенностью сложных многоцелевых систем.
- Быть понятен аналитикам и программистам, а также должен поддерживаться специальными инструментальными средствами, реализованными на различных компьютерных платформах.

UML и OMG

- Поддержка разработки языка UML становится одной из целей консорциума OMG (Object Management Group).
  - Консорциум OMG был образован еще в 1989 году с целью разработки предложений по стандартизации объектных и компонентных технологий CORBA
- Язык UML приобрел статус второго стратегического направления в работе OMG.
- Июнь1995: OMG организует совещание по методологиям ООАП
  - Результат: признанение целесообразности поиска индустриальных стандартов в области языков моделирования

Язык UML v1.0

- Строгое определение версии 1.0 языка UML было выполнено консорциумом разработчиков ПО, в который вошли:
  - Rational Software, Digital Equipment Corp., HP, i-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI и Unisys.
- В январе 1997 года был опубликован документ с описанием языка UML 1.0, как начальный вариант ответа на запрос предложений RTP.
- Эта версия языка моделирования была достаточно хорошо определена, обеспечивала требуемую выразительность и мощность и предполагала решение широкого класса задач.

Использование UML

- Статус языка UML определен как открытый для всех предложений по его доработке и совершенствованию.
- На основе технологии UML Microsoft, Rational Software и другие поставщики средств разработки программных систем разработали единую информационную модель, которая получила название UML Information Model.
- Предполагается, что эта модель даст возможность различным программам, поддерживающим идеологию UML, обмениваться между собой компонентами и описаниями.
- Все это позволит создать стандартный интерфейс между средствами разработки приложений и средствами визуального моделирования.

Использование UML

- Существуют средства визуального программирования на основе UML, обеспечивающие интеграцию, включая прямую и обратную генерацию кода программ, с наиболее распространенными языками и средами программирования:
  - MS Visual C++, Java, Object Pascal/Delphi, Power Builder, MS Visual Basic, Forte, Ada, Smalltalk.
- Поскольку при разработке языка UML были приняты во внимание многие передовые идеи и методы, можно ожидать, что на очередные версии языка UML также окажут влияние и другие перспективные технологии и концепции.
- Кроме того, на основе языка UML могут быть определены многие новые перспективные методы. Язык UML может быть расширен без переопределения его ядра.

Назначение языка UML (1)

Предоставить в распоряжение пользователей легко воспринимаемый и выразительный язык визуального моделирования, специально предназначенный для разработки и документирования моделей сложных систем самого различного целевого назначения

Назначение языка UML (2)

Снабдить исходные понятия языка UML возможностью расширения и специализации для более точного представления моделей систем в конкретной предметной области

Назначение языка UML (3)

- Описание языка UML должно поддерживать такую спецификацию моделей, которая не зависит от конкретных языков программирования и инструментальных средств проектирования программных систем.

Назначение языка UML (4)

- Описание языка UML должно включать в себя семантический базис для понимания общих особенностей ООАП

Назначение языка UML (5)

- Поощрять развитие рынка объектных инструментальных средств.

Назначение языка UML (6)

- Способствовать распространению объектных технологий и соответствующих понятий ООАП

Назначение языка UML (7)

- Интегрировать в себя новейшие и наилучшие достижения практики ООАП

Общая структура языка UML

- язык UML состоит из двух взаимодействующих частей:
  - Семантика языка UML - некоторая метамодель, которая определяет абстрактный синтаксис и семантику понятий объектного моделирования на языке UML.
  - Нотация языка UML - графическая нотация для визуального представления семантики языка UML.

Пакеты в языке UML

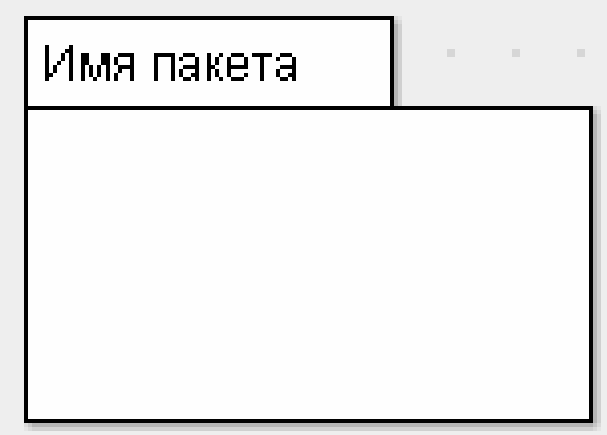
- **Пакет** – основной способ организации элементов модели в языке UML.
- Каждый пакет владеет всеми своими элементами, т. е. теми элементами, которые включены в него.
- Про соответствующие элементы пакета говорят, что они принадлежат пакету или входят в него.
- Каждый элемент может принадлежать только одному пакету.
- Одни пакеты могут быть вложены в другие пакеты. В этом случае первые называются подпакетами, поскольку все элементы подпакета будут принадлежать более общему пакету.
- Тем самым для элементов модели задается отношение вложенности пакетов, которое представляет собой иерархию.

Пакеты

- При рассмотрении отношения «пакет-подпакет» наиболее естественно ассоциировать его с более общим отношением «множество-подмножество
- Действительно, поскольку пакет можно рассматривать в качестве частного случая множества, такая интерпретация помогает нам использовать и графические средства для представления соответствующих отношений между пакетами.

Пакеты

- Для графического изображения пакетов на диаграммах применяется специальный графический символ – большой прямоугольник с небольшим прямоугольником, присоединенным к левой части верхней стороны первого

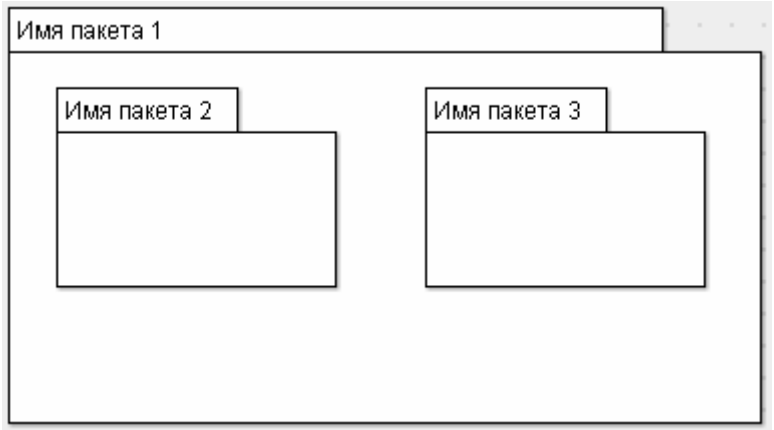


Пакеты

- Имя пакета – строка или несколько строк текста, содержащая любое число букв, цифр и некоторых специальных знаков:
- Например:
  - Графический интерфейс, форма ввода данных, …
- Перед именем пакета может помещаться строка текста, содержащая некоторое ключевое слово – стереотип:
- Например: *facade*, *framework*, *stub* и *topLevel*

Пакеты

- Пакеты допускают вложение других пакетов, других типов диаграмм UML



Структура языка UML и пакеты

- Основа представления UML на метамодельном уровне - описание трех его логических блоков или пакетов: Основные элементы, Элементы поведения и Общие механизмы

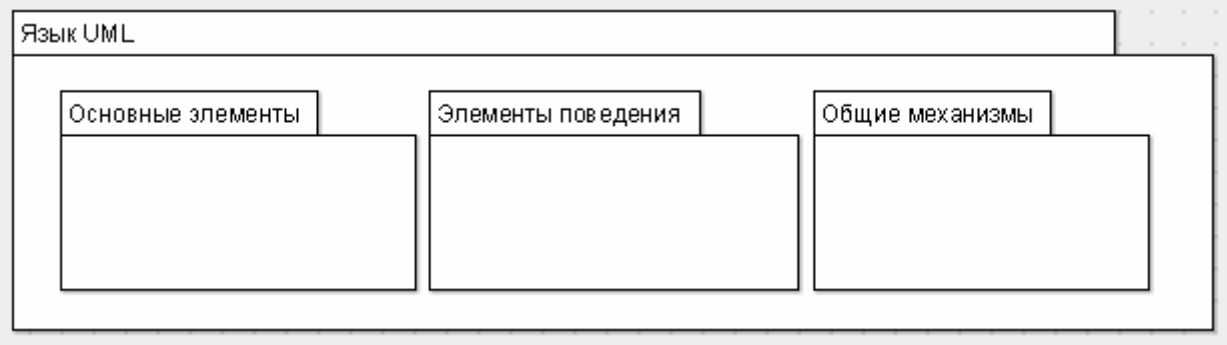


Диаграмма вариантов использования (use case diagram)

- Визуальное моделирование в UML можно представить как некоторый процесс поуровневого спуска от наиболее обшей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы.
- Для достижения этих целей вначале строится модель в форме так называемой диаграммы вариантов использования (**use case diagram**), которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования.
- **Диаграмма вариантов использования** является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Диаграмма вариантов использования

Разработка диаграммы вариантов использования преследует цели:

- Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.
- Сформулировать общие требования к функциональному поведению проектируемой системы.
- Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.
- Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования

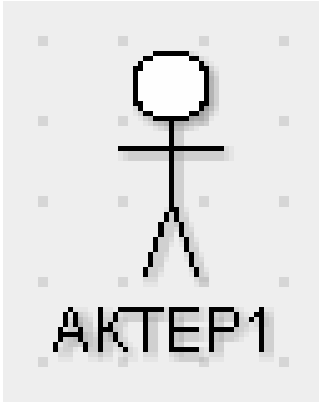
- Суть данной диаграммы:
- система представляется в виде множества сущностей-**актеров**, взаимодействующих с системой с помощью так называемых **вариантов использования**.
- **Эктором** (**actor**) или **действующим лицом** называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.
- **Вариант использования** (**use case**) служит для описания сервисов, которые система предоставляет эктору. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с эктором.
- При этом ничего не говорится о том, каким образом будет реализовано взаимодействие экторов с системой.

Эктор

- Эктор - любая внешняя по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач.
- Эктор может рассматриваться как некая отдельная **роль** относительно конкретного варианта использования.

Эктор

- Стандартным графическим обозначением эктора на диаграммах является фигурка «человечка», под которой записывается конкретное имя эктора



Эктор

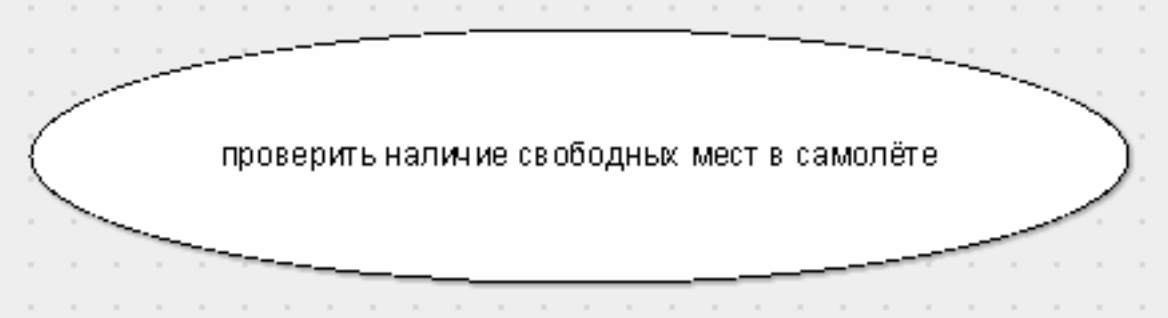
- Имена экторов должны записываться заглавными буквами
- Имена абстрактных экторов должны записываться *курсивом*
- Имя эктора должно быть достаточно информативным с точки зрения семантики:
  - клиент банка, водитель автомобиля, мобильный телефон,…

Вариант использования

- **Вариант использования (use case)** определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим эктором

Вариант использования

- Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое название или имя в форме глагола с пояснительными словам

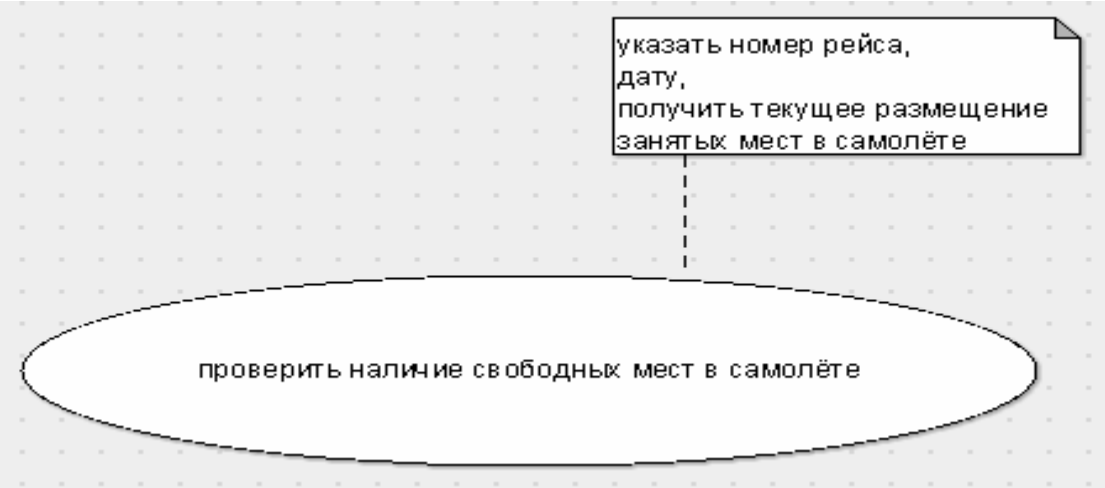


Вариант использования

- Каждый **вариант использования соответствует отдельному сервису**, который предоставляет моделируемую сущность или систему по запросу пользователя (актера), т. е. определяет способ применения этой сущности.
- **Сервис**, который инициализируется по запросу пользователя, представляет собой **законченную последовательность действий**.
- Это означает, что после того как система закончит обработку запроса пользователя, она должна возвратиться в **исходное состояние**, в котором готова к выполнению следующих запросов

Вариант использования

- Диаграмма вариантов может дополняться пояснительным текстом - **сценарием**, который раскрывает смысл или семантику составляющих ее компонентов

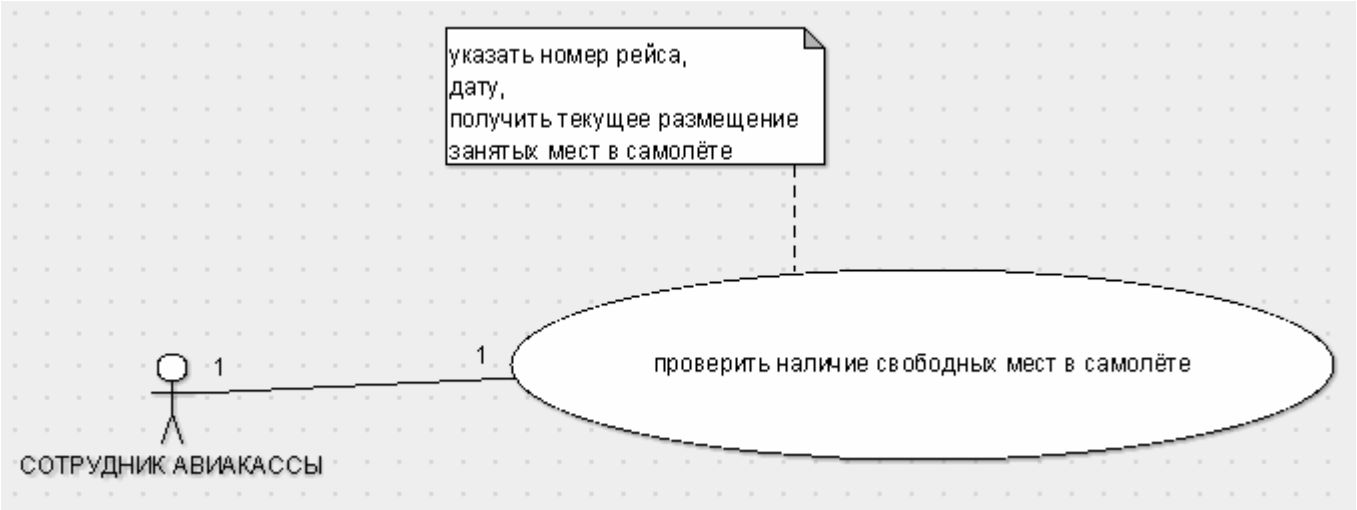


Отношения на диаграмме вариантов использования

- Существуют такие виды отношений:
  - ассоциация (association relationship)
  - расширения (extend relationship)
  - обобщения (generalization relationship)
  - включения (include relationship)

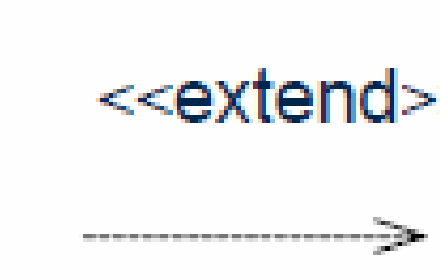
Отношение ассоциации

- Участие эктора в варианте использования
- На диаграмме: ————————

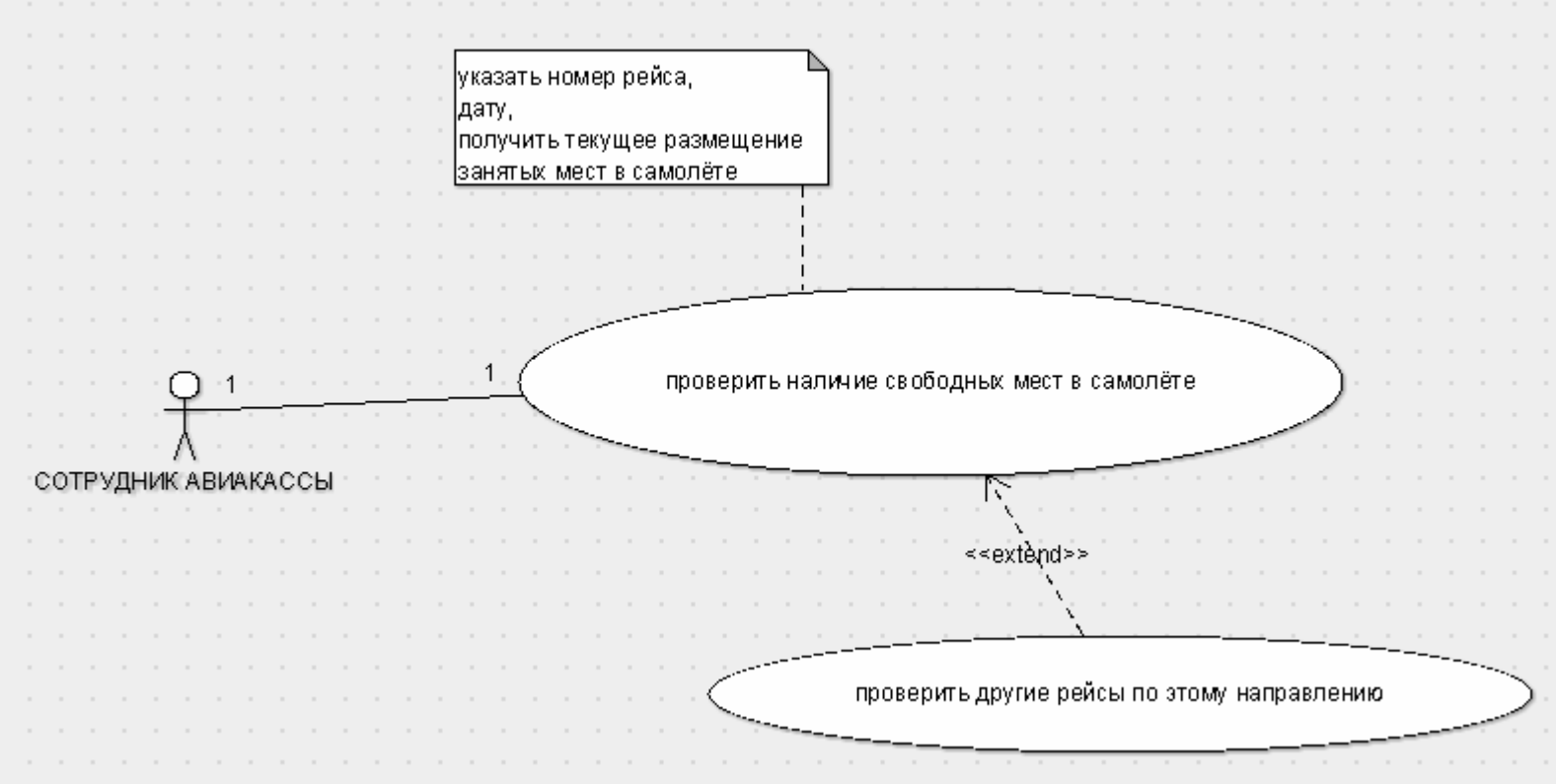


Отношение расширения

- Отношение от расширенного варианта использования к базовому варианту использования
- Показывает, как поведение внутри расширенного варианта использования может быть добавлено в поведение базового варианта использования
- На диаграмме:



Пример отношения расширения

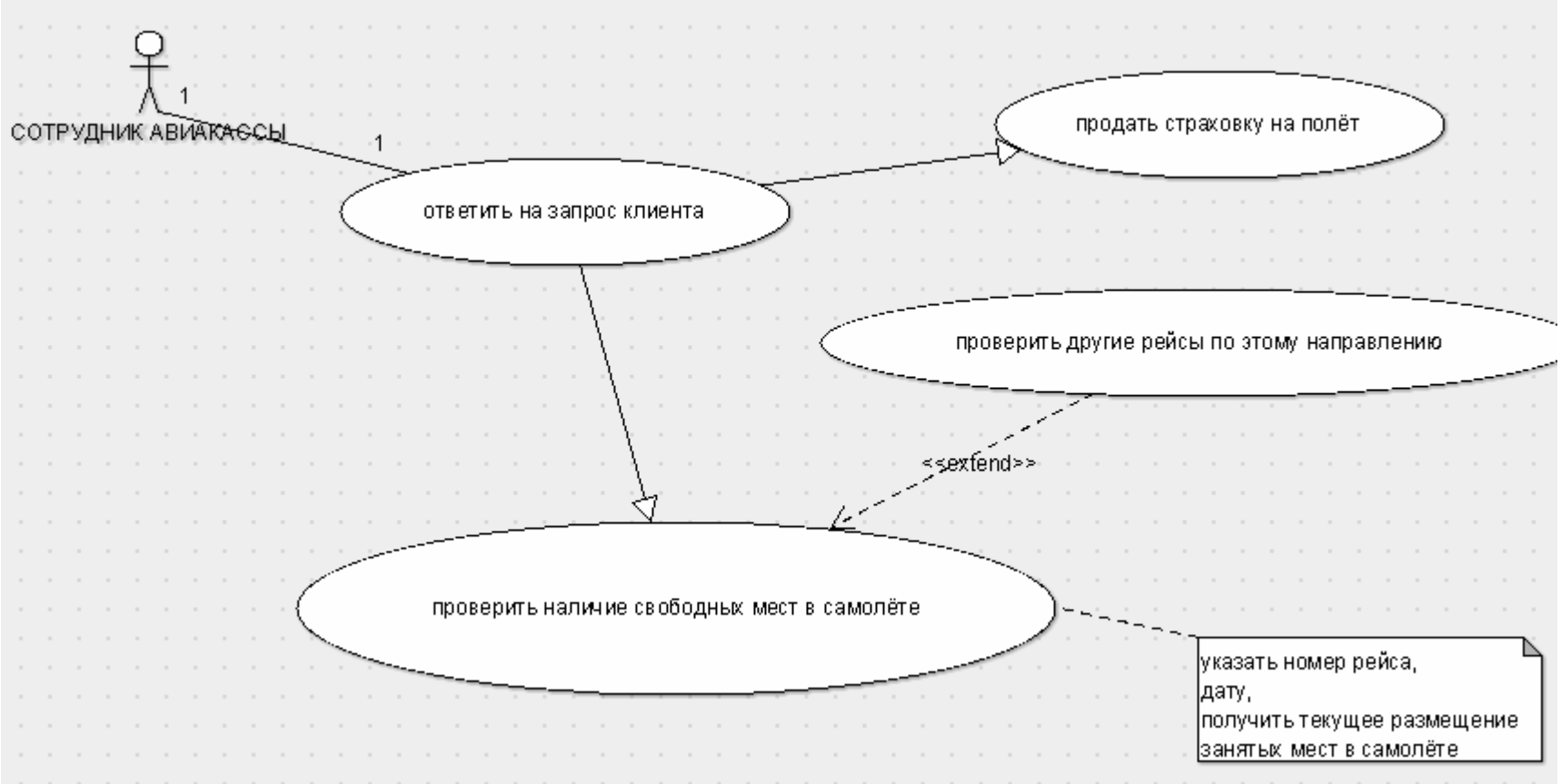


Отношение обобщения

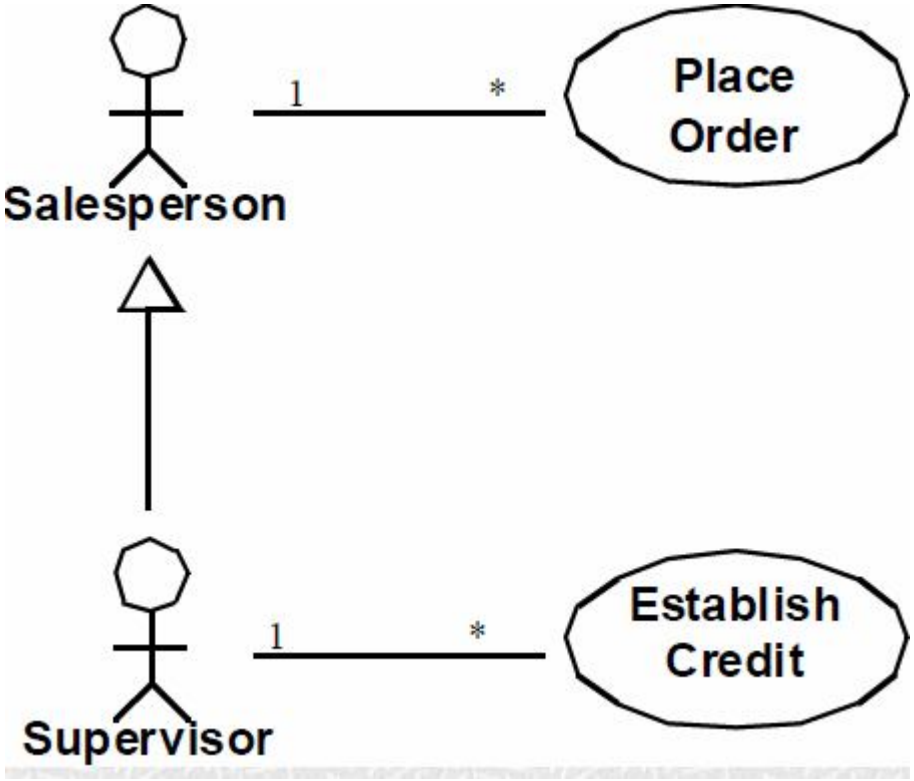
- Отношение обобщения – таксономическое отношение между более общим вариантом использования и более частным

- На диаграмме 

Пример отношения обобщения



Отношение обобщения между актёрами



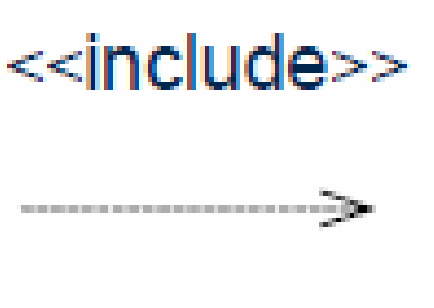
Отношение включения

- Отношение включения указывает, что некоторое заданное поведение для одного варианта использования включается **в качестве составного компонента** в последовательность поведения другого варианта использования.
- Данное отношение является направленным бинарным отношением в том смысле, что пара экземпляров вариантов использования всегда упорядочена в отношении включения.

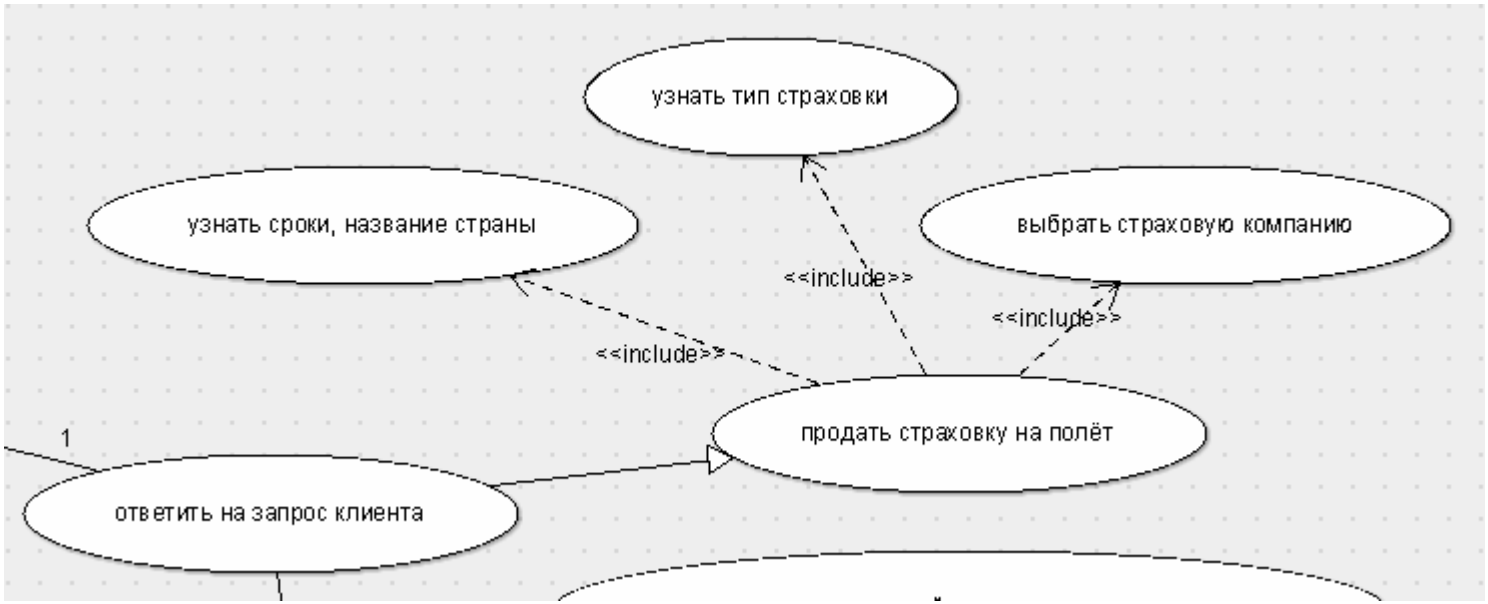
Отношение включения

- Отношение включения, направленное от варианта использования А к варианту использования В, указывает, что каждый экземпляр варианта А включает в себя функциональные свойства, заданные для варианта В.

- На диаграмме

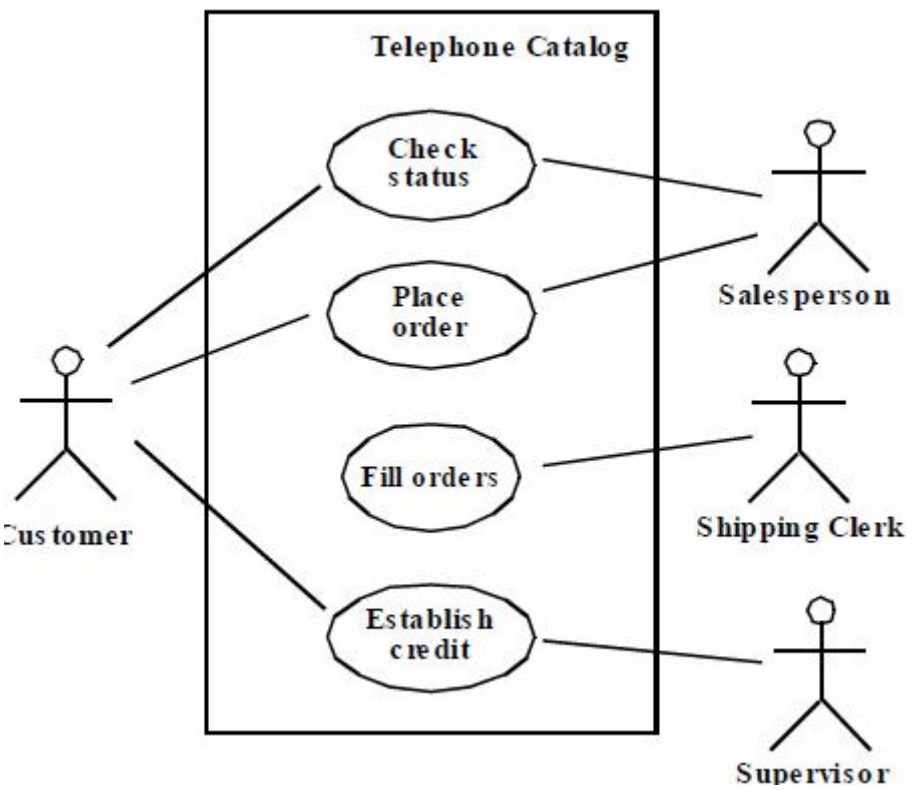


Пример отношения включения



Понятие границы системы

- Представляет **границу** между физической системой и экторами, взаимодейству-ющими с этой физической системой.



Резюме: что узнали на лекции?

- Познакомились с идеологией UML

- Познакомились с диаграммами вариантов использования