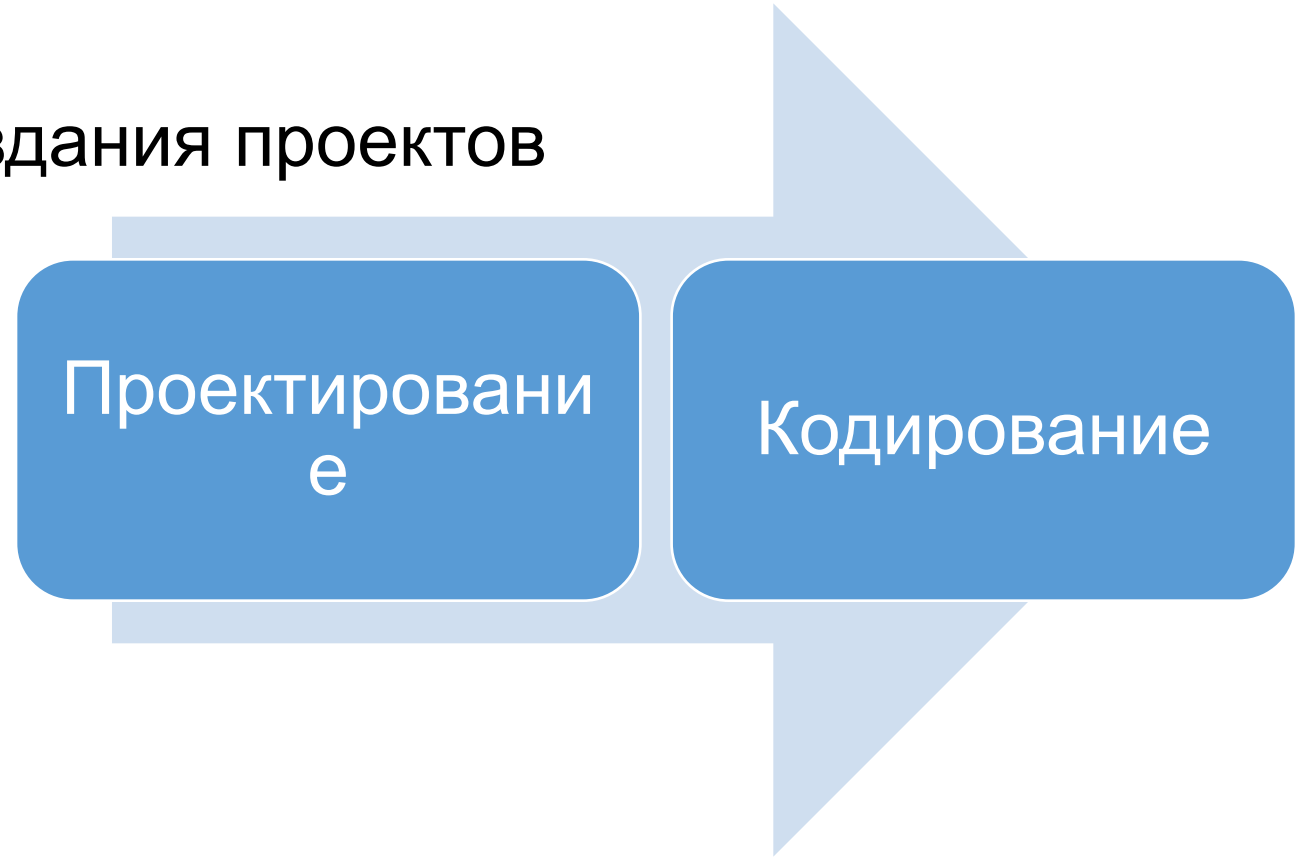
UML

Принцип создания проектов



Проектирование

В ходе проектирования архитектором создается проектная документация, включающая:

- текстовые описания
- диаграммы
- модели будущей программы

Для этого используется графический язык для визуализации, описания параметров, конструирования и документирования различных систем – **UML**

UML(Unified Modeling Language)

- Цель UML – проектирование, документирование, визуальное описание основных компонентов проекта
- Диаграмма – визуальное описание процесса, классов, взаимодействия
- На каждый процесс – своя диаграмма
- Хранение важной информации в эл. виде
- Не язык программирования (но можно генерировать код)
- Дополняет ТЗ
- Видение процесса «сверху»

Кто может использовать UML

Заказчик – общие задачи и цели проекта

Аналитик – подходы, правильность работы системы и её частей, «слои» приложения

Разработчик/Архитектор – дизайн кода, архитектура классов, объектов и взаимодействий

Тестировщик – проверка функционала на всех уровнях

Менеджер проекта – общая картина по проекту

Плюсы

- **Универсальность – единая технология**
- Автоматическая генерация кода на основе UML-диаграмм
- Широкое применение – ИТ, бизнес и др.
- Поддержка ООП
- **Большое количество типов диаграмм**
- Удобные инструменты, плагины для многих IDE
- **Разбор основных моментов проекта без изучения кода**
- Миграция диаграмм инструментальными средствами

Минусы

- Изучение UML
- Для начинающих – путаница в количестве диаграмм
- Знание ООП
- Детализация/поверхностное описание
- **Учебные материалы сложны и запутаны**

Типы диаграмм

Структурные

Structure diagrams

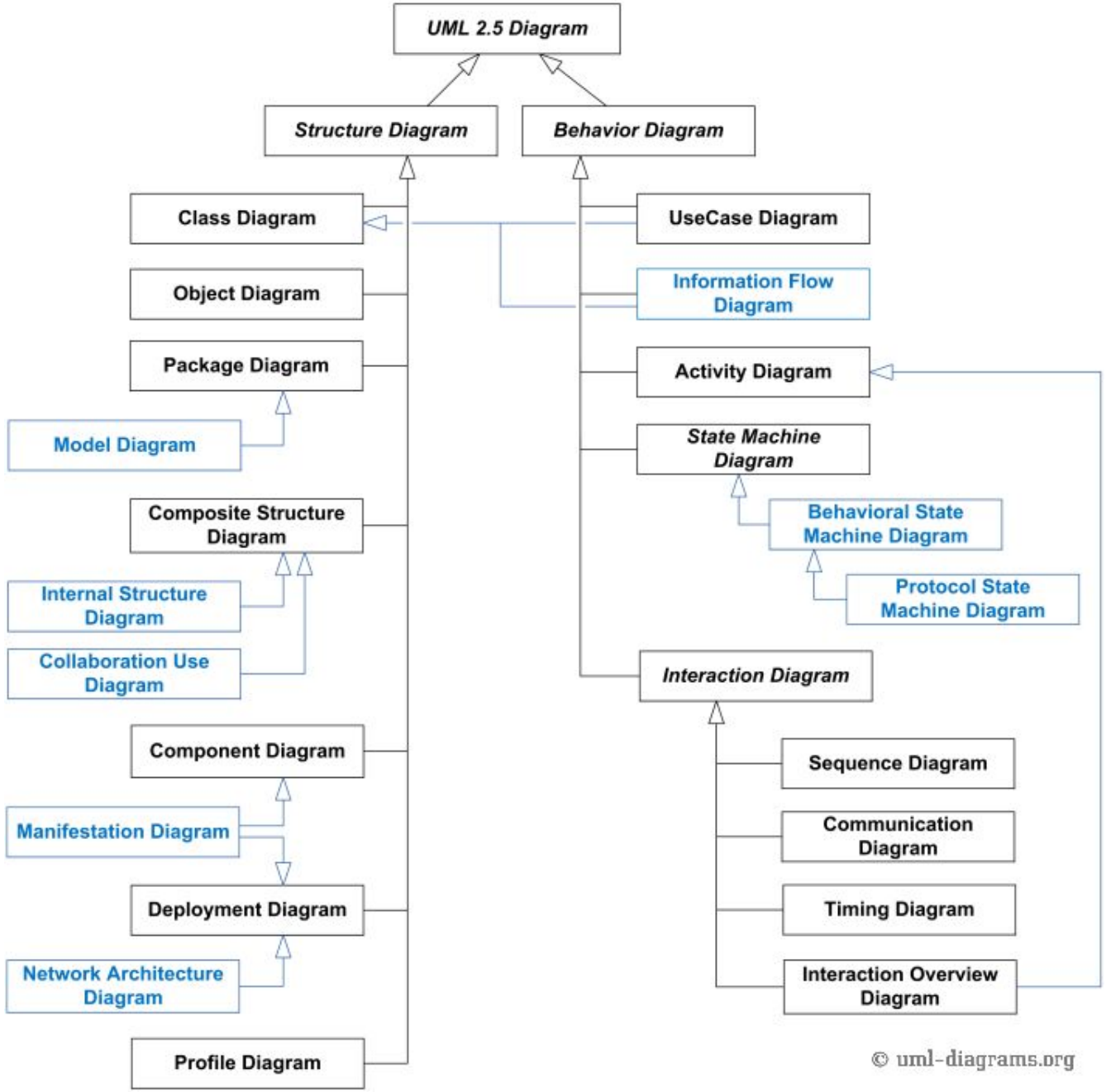
- ✔ Общая картина взаимодействия
- ✔ Как устроено, кто с кем связан

Поведенческие

Behavior diagrams

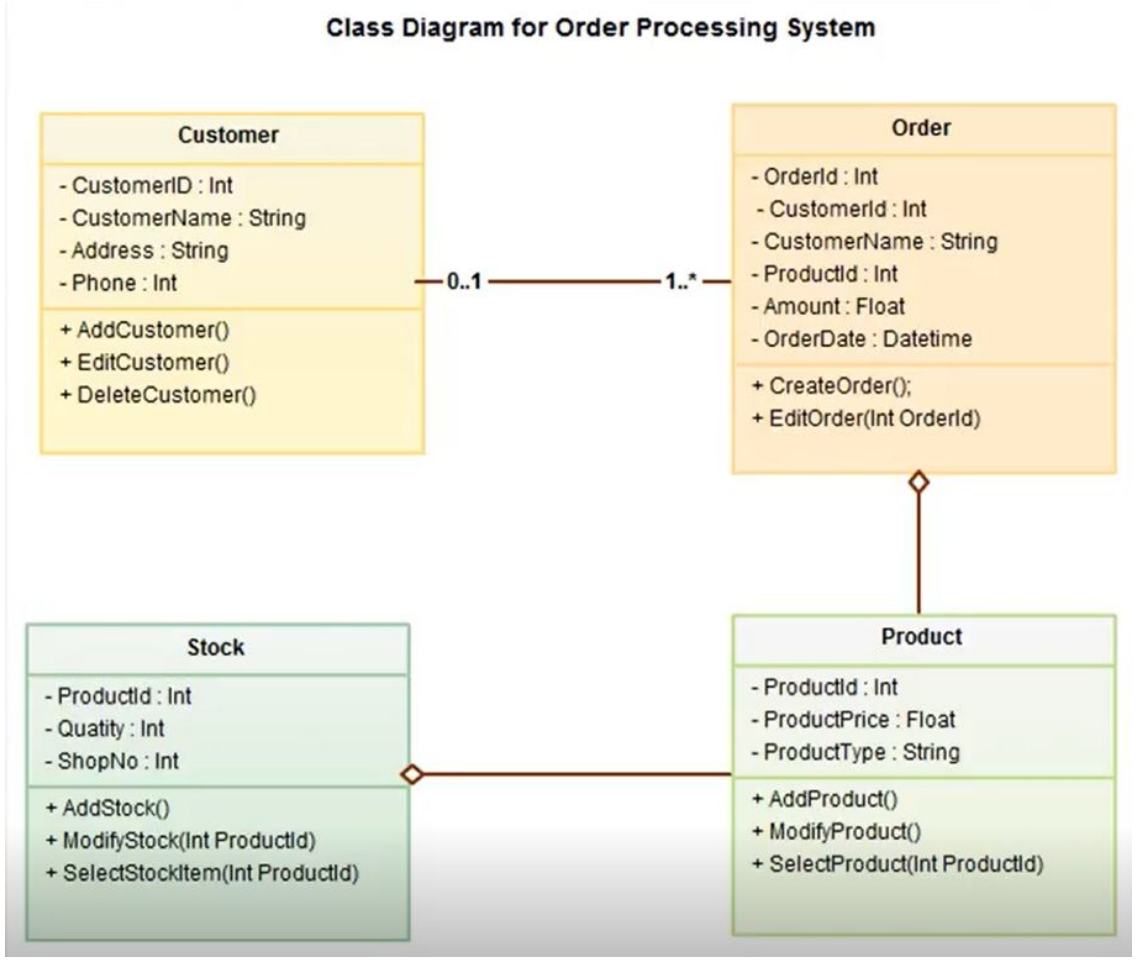
- ✔ Динамическое поведение, изменение состояния во времени
- ✔ Как работает, последовательность процессов

Типы диаграмм



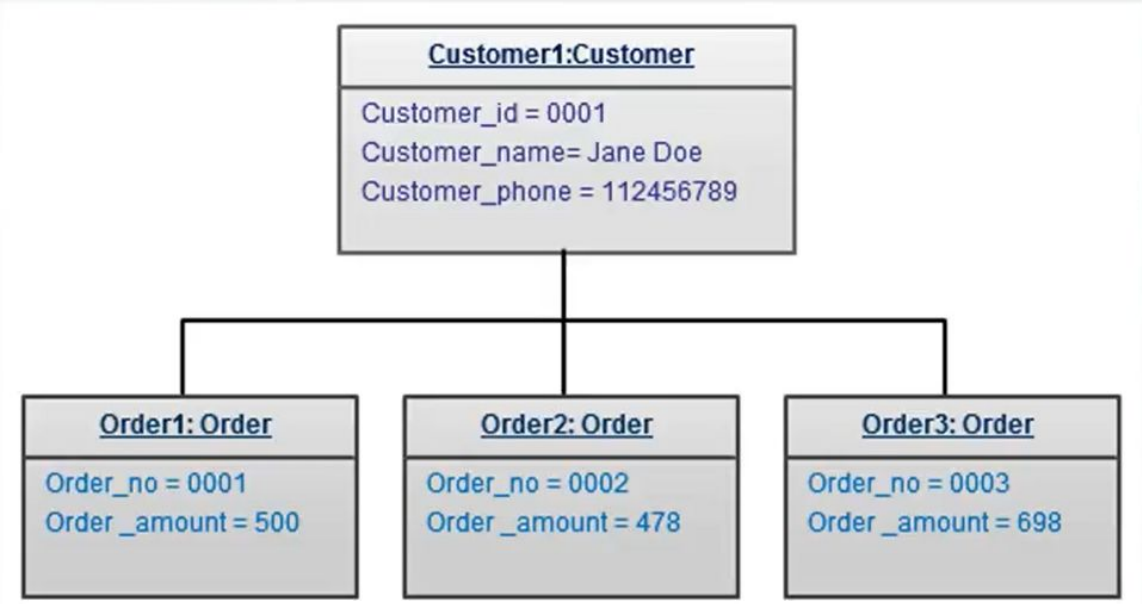
Class diagram

- ✔ Описание классов, интерфейсов, связей, методов
- ✔ Структура в стиле ООП
- ✔ Позволяет понять работу кода без изучения самого кода
- ✔ Автогенерация кода на основе диаграмм



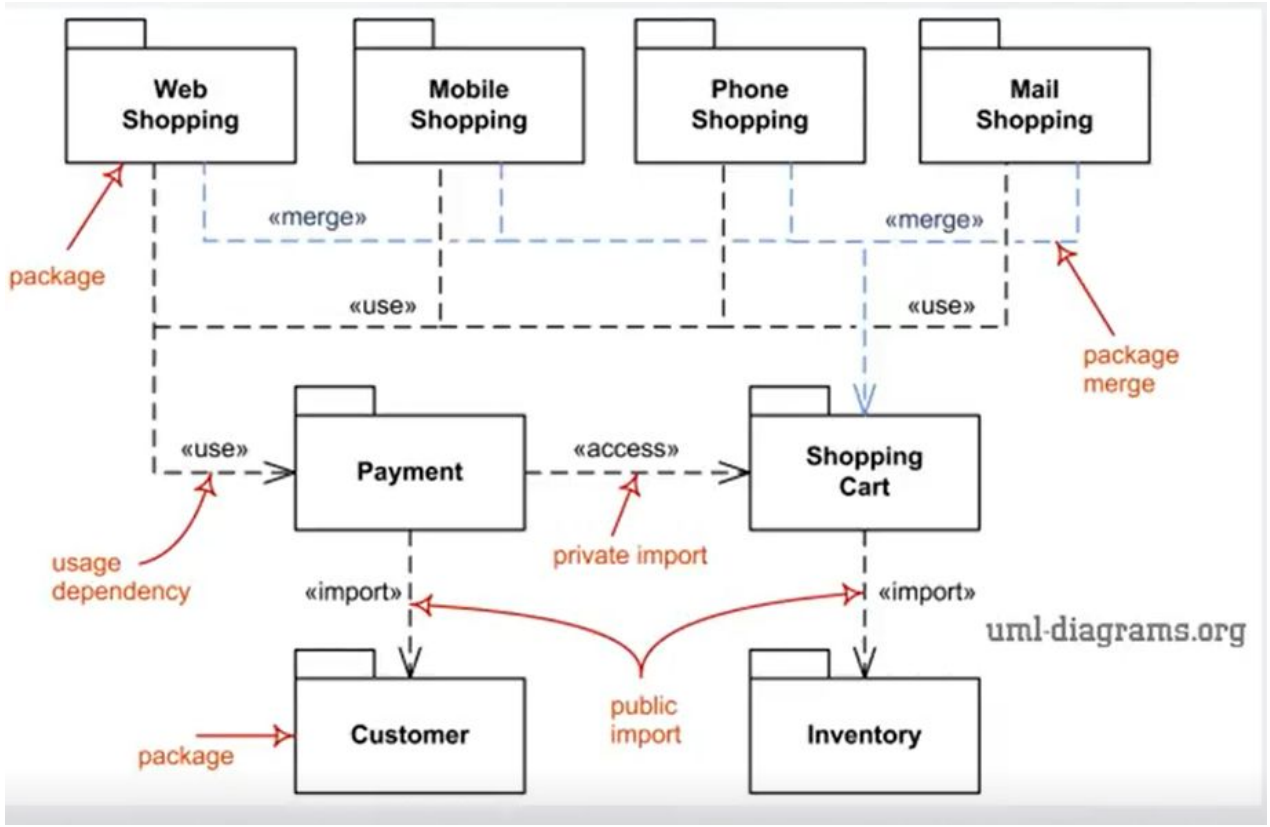
Object diagram

- ✔ Состояние экземпляров классов с конкретными значениями полей в определенный момент времени
- ✔ Похож на диаграмму классов, но в режиме runtime (во время работы программы)



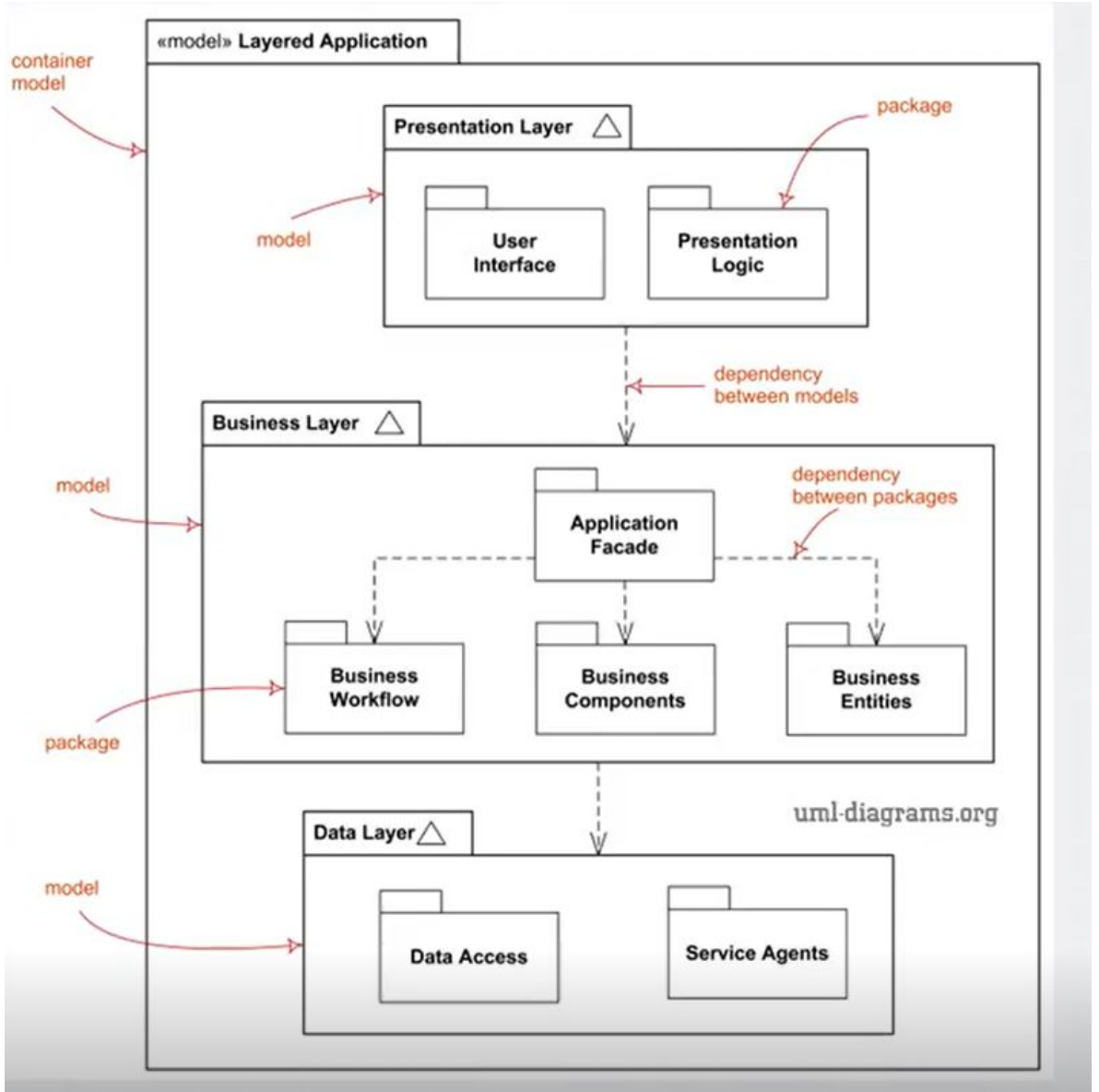
Package diagram

- ✔ Показывает вложенность и связи между пакетами
- ✔ Более высокий уровень, чем классы



Model diagram

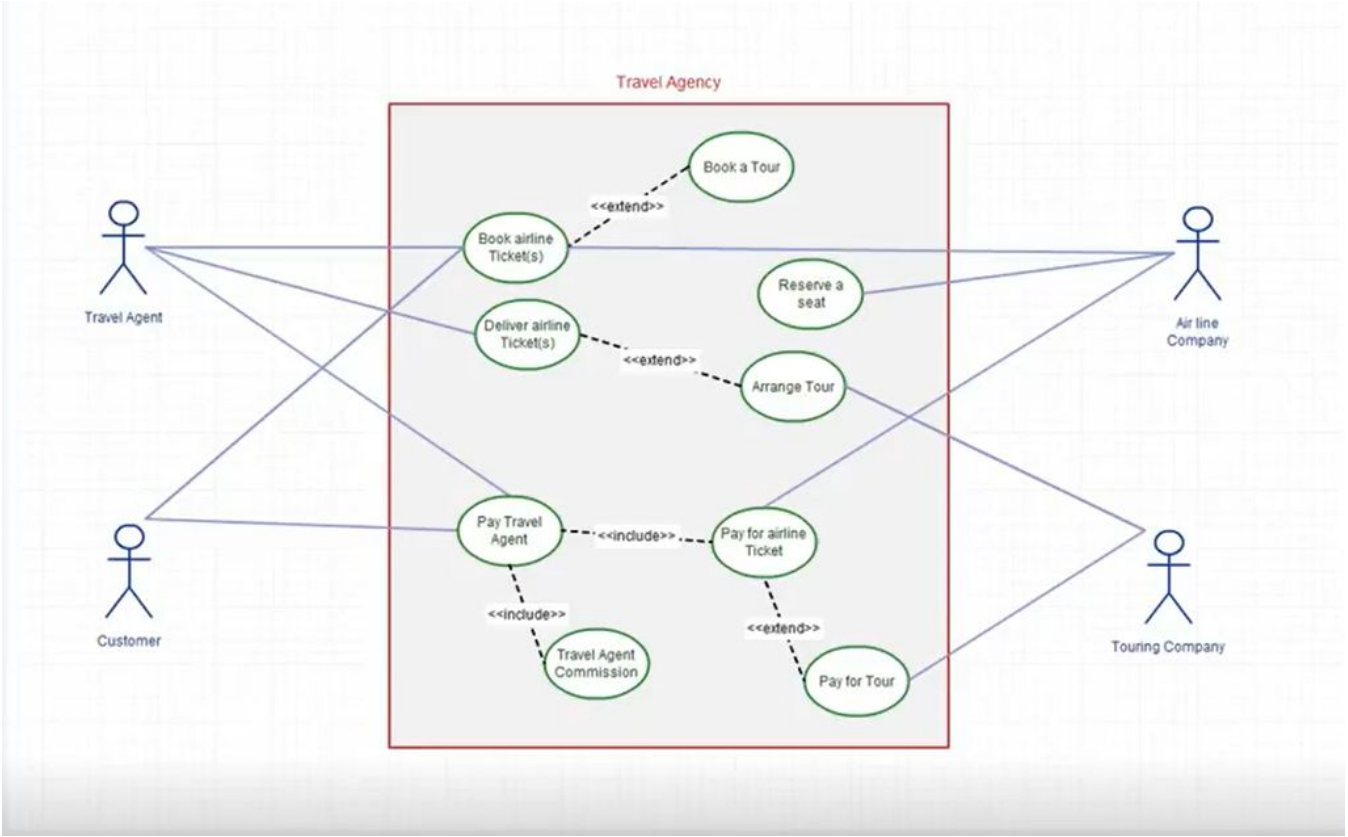
- ✔ Описание «слоев» проекта
- ✔ Используется для многоуровневых приложений
- ✔ Часто используется в ТЗ для общего описания частей проекта



Use Case Diagram

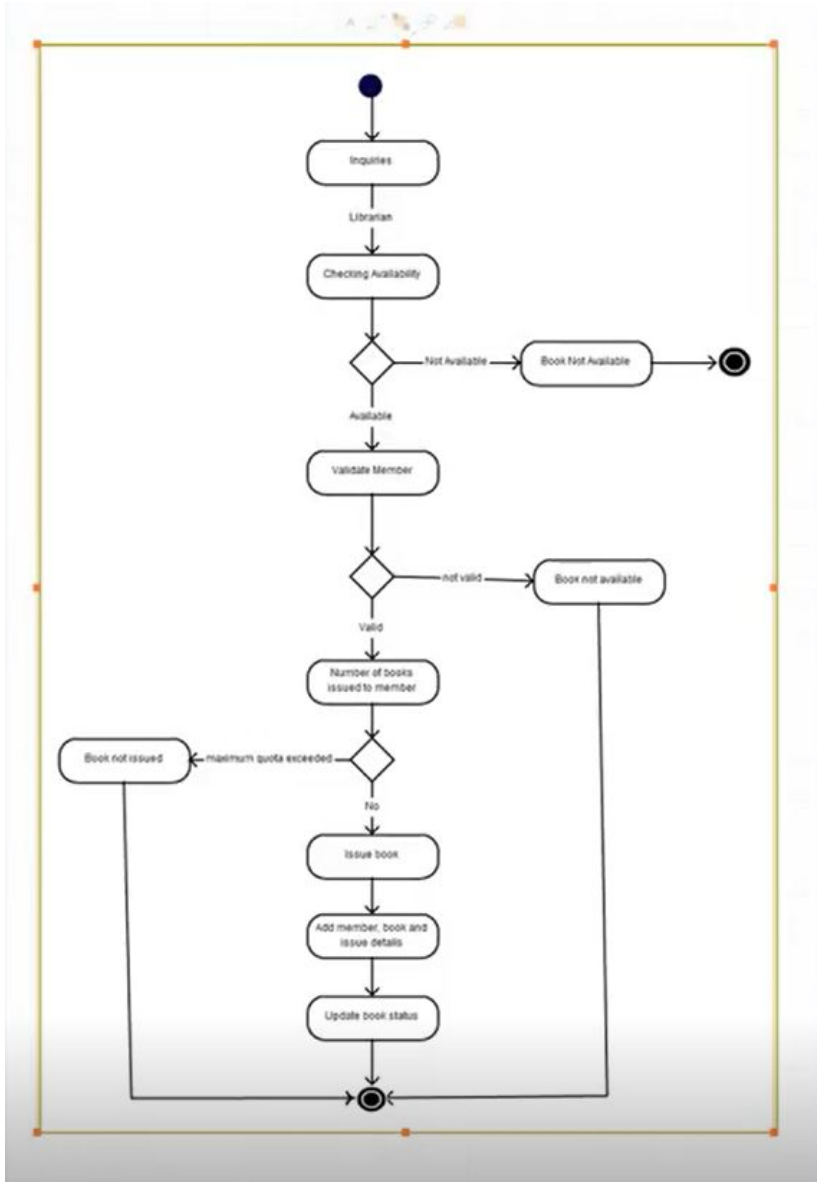
Диаграмма прецедентов/вариантов использования

- ✔ Описание возможных сценариев работы с системой с точки зрения пользователя
- ✔ Возможные пути использования системы
- ✔ Описание всех участников системы (актеры)



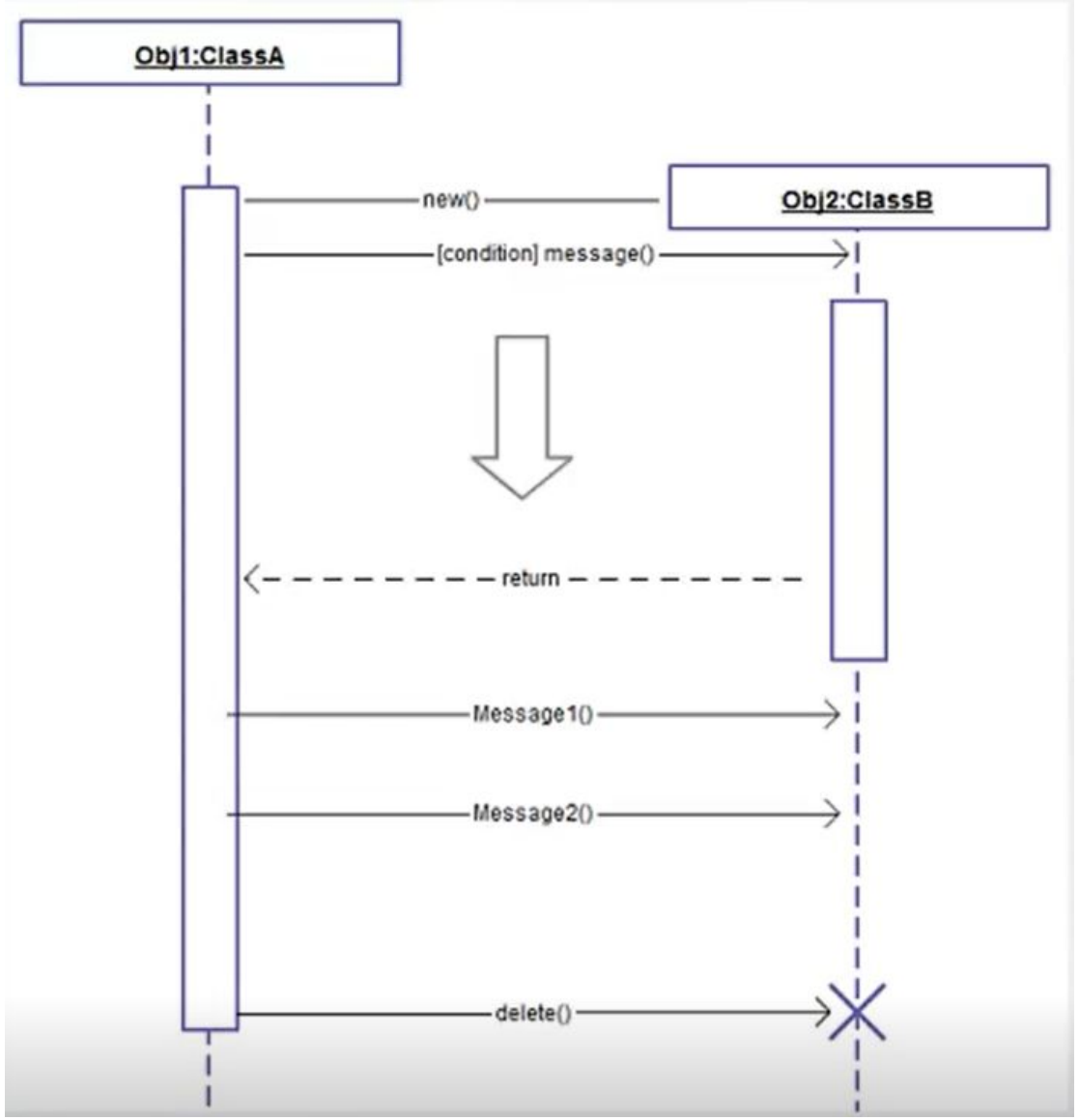
Activity Diagram

- ✔ Описание возможных бизнес-процессов приложения
- ✔ Взаимодействие «потоков», пошаговое представление действия
- ✔ Более низкий уровень, чем UseCase
- ✔ Может быть конкретным описанием блоков UseCase диаграмм



Sequence diagram

- ✔ Последовательность взаимодействия объектов для определенного бизнес-процесса
- ✔ Как объекты друг друга вызывают и какие данные передают
- ✔ Показывают объекты в действии
- ✔ Показывают время жизни объектов (создание, удаление)



Deployment diagram

- ✔ Описание архитектуры, топологии системы (ОС, БД, сервера и пр.)
- ✔ Информация для администраторов

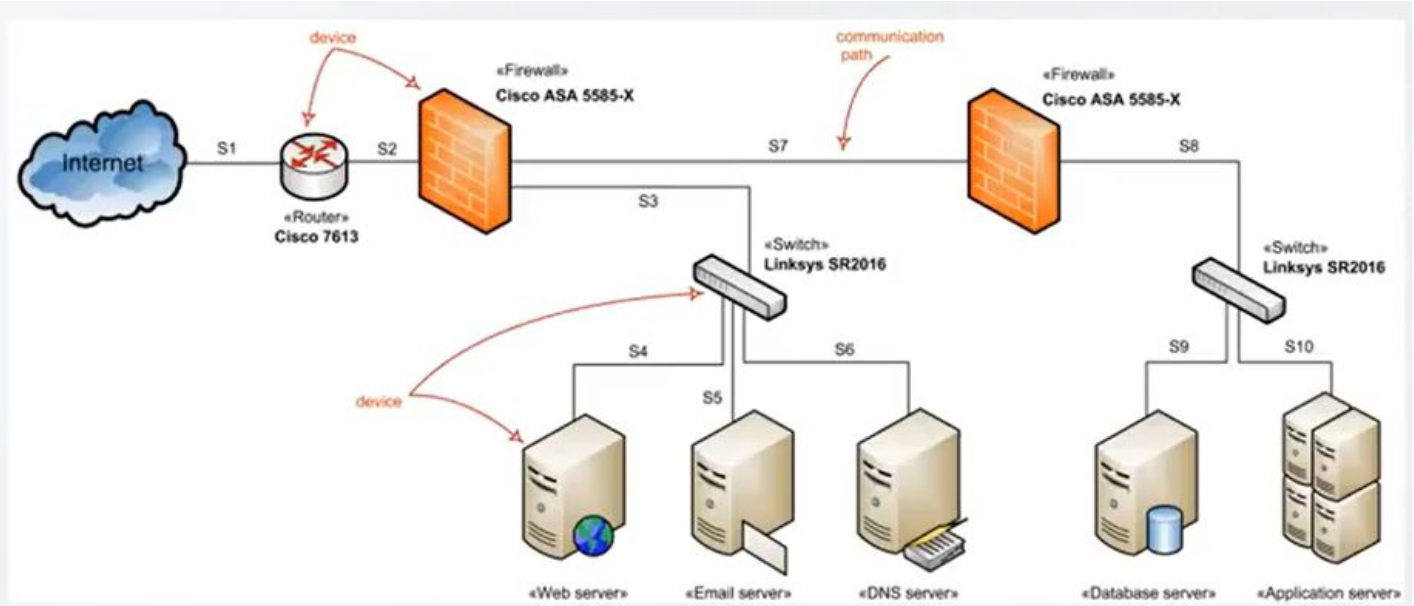
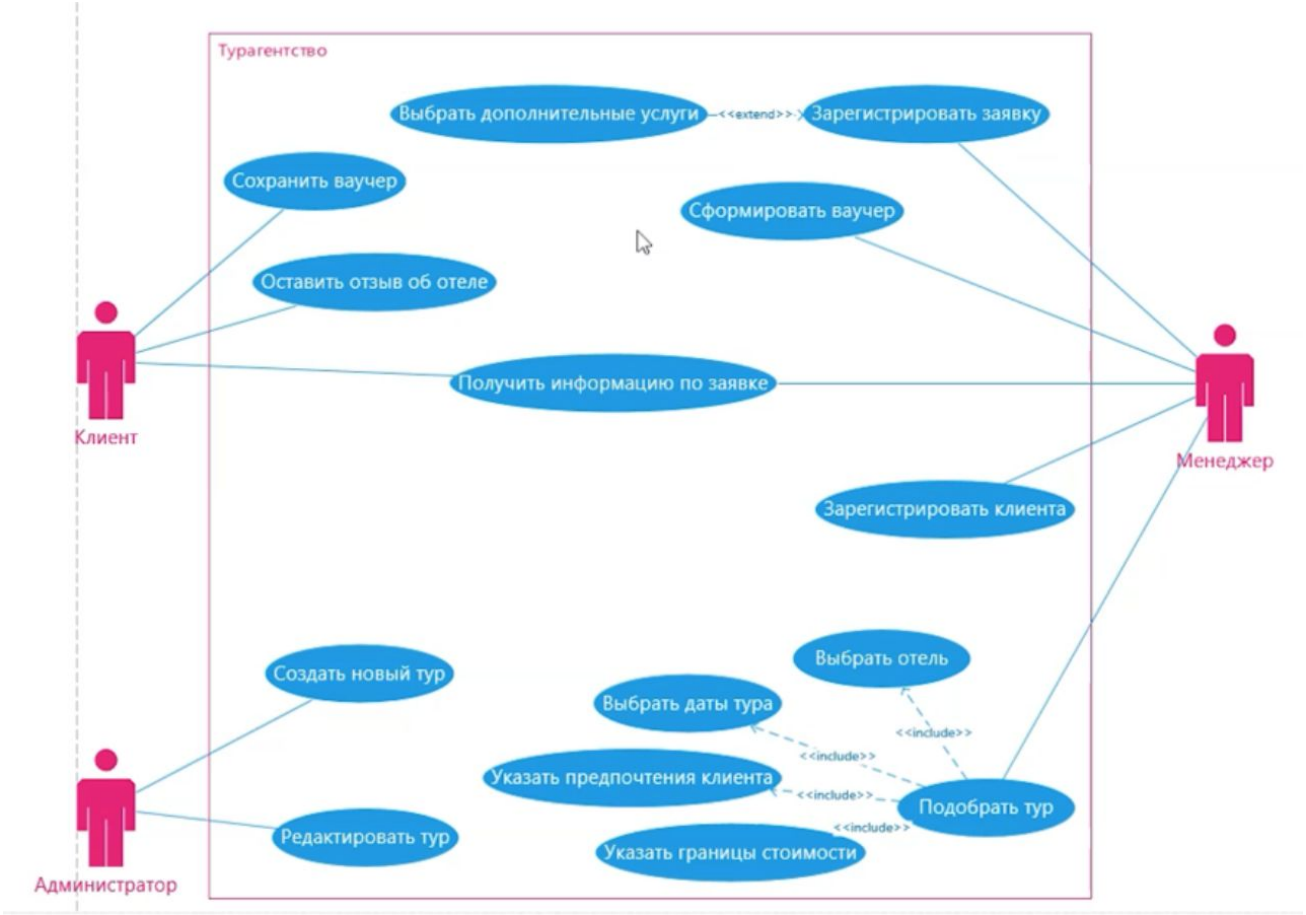


Диаграмма вариантов использования (Use Case Diagram)

- Диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне
- Диаграмма вариантов использования достаточно проста, что позволяет использовать ее для согласования технического задания с заказчиком

Пример Use Case Diagram



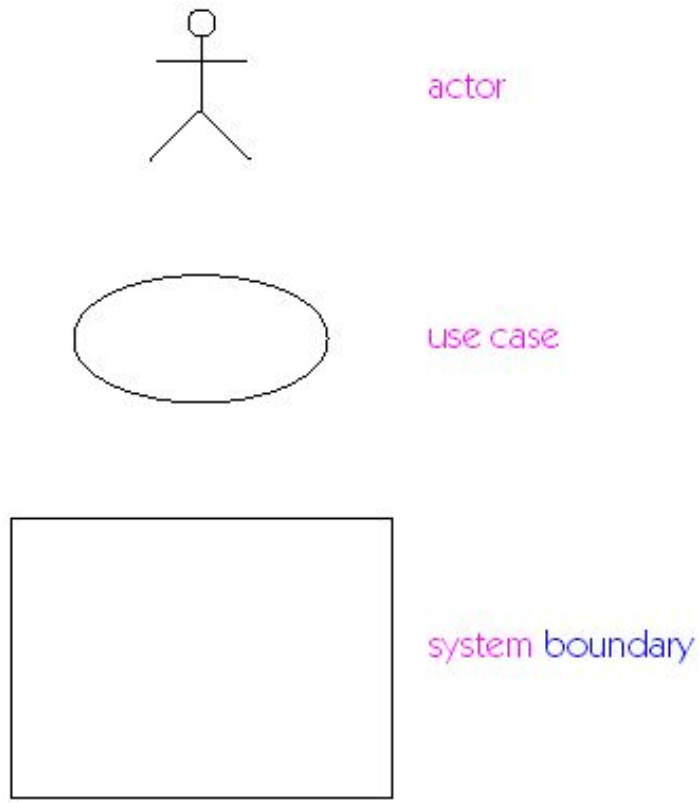
Назначение диаграммы вариантов использования

- Определить общие границы функциональности проектируемой системы в контексте моделируемой предметной области.
- Специфицировать требования к функциональному поведению проектируемой системы в форме вариантов использования.
- 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

Прецеденты

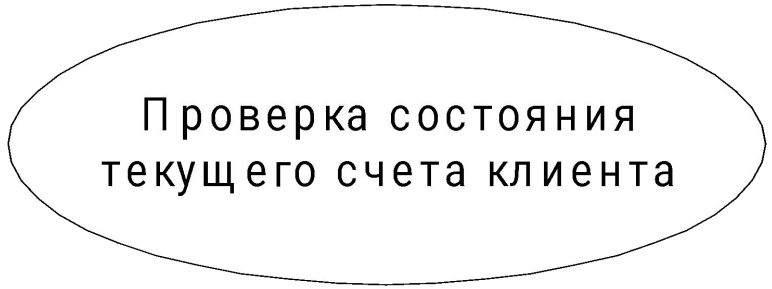
- ✔ UseCase (случай использования, прецедент) – набор сценариев, путей, которые нужно выполнить для достижения целей приложения (с точки зрения пользователей)
- ✔ Описывает участников (actor), возможные сценарии (успешные и неудачные), которые выполняются для решения задач приложения
- ✔ Может описывать основные сценарии, которые составляют «ядро» приложения (дополнительные сценарии могут добавляться по ходу разработки)
- ✔ Ориентация на пользователей

Основные обозначения на диаграмме вариантов использования



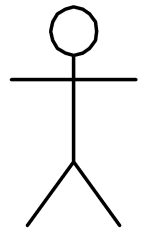
Вариант использования (use case)

- Представляет собой общую спецификацию совокупности выполняемых системой действий с целью предоставления некоторого наблюдаемого результата, который имеет значение для одного или нескольких актеров
- Отвечает на вопрос «Что должна выполнять система?», не отвечая на вопрос «Как она должна выполнять это?»
- Имена – отглагольное существительное или глагол в неопределенной форме



Актер (actor)

- Любая внешняя по отношению к проектируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач
- Примеры актеров: кассир, клиент банка, банковский служащий, президент, продавец магазина, менеджер отдела продаж, пассажир авиарейса, водитель автомобиля, администратор гостиницы, сотовый телефон

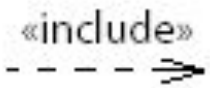
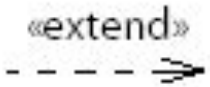


Клиент банка

Вопросы для идентификации актеров в системе

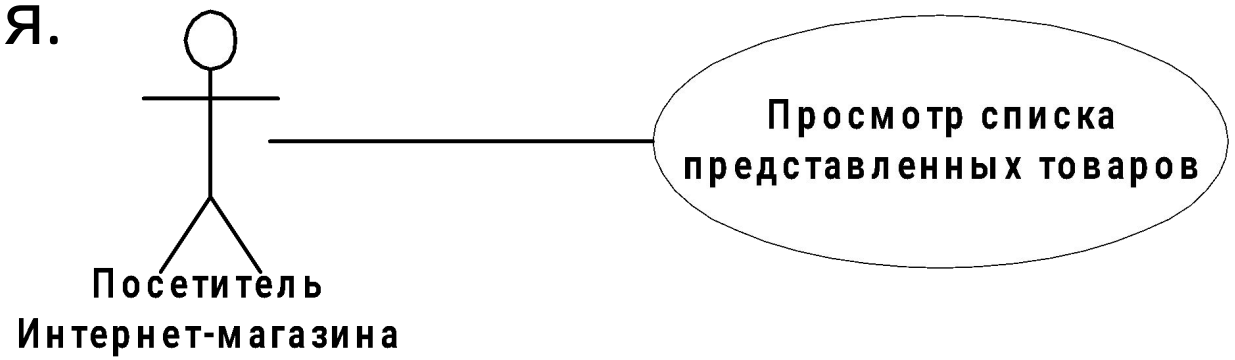
- Какие организации или лица будут использовать систему
- Кто будет получать пользу от использования системы
- Кто будет использовать информацию от системы
- Будет ли использовать система внешние ресурсы
- Может ли один пользователь играть несколько ролей при взаимодействии с системой
- Могут ли различные пользователи играть одну роль при взаимодействии с системой

Отношения на диаграмме вариантов использования

Relationship	*Function*	*Notation*
association	The communication path between an actor and a use case that it participates in	
include	The insertion of additional behavior into a base use case that explicitly describes the insertion	
extend	The insertion of additional behavior into a base use case that does not know about it	
use case generalization	A relationship between a general use case and a more specific use case that inherits and adds features to it	

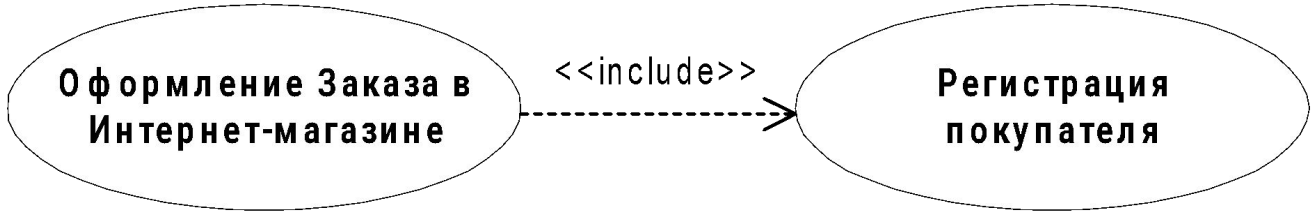
Отношение ассоциации

- Ассоциация (association) является одним из фундаментальных понятий в языке UML 2.x и может использоваться на различных канонических диаграммах при построении визуальных моделей
- Применительно к диаграммам вариантов использования отношение ассоциации может служить только для обозначения взаимодействия актера с вариантом использования.



Отношение включения

- Отношение зависимости (dependency) определяется как форма взаимосвязи между двумя элементами модели, предназначенная для спецификации того обстоятельства, что изменение одного элемента модели приводит к изменению некоторого другого элемента
- Отношение включения (include) специфицирует тот факт, что некоторый вариант использования содержит поведение, определенное в другом варианте использования

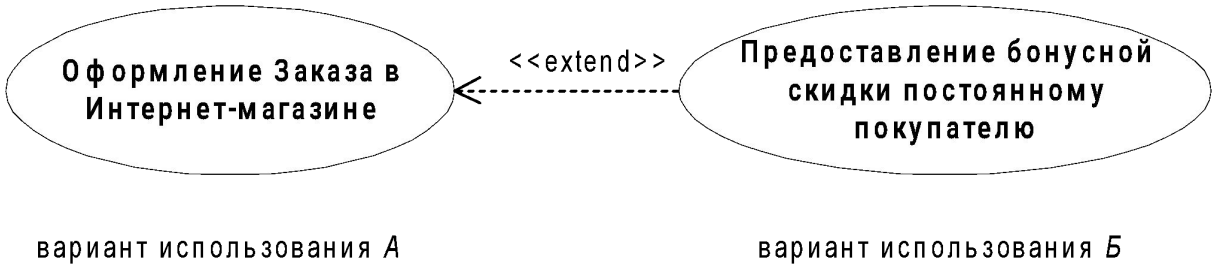


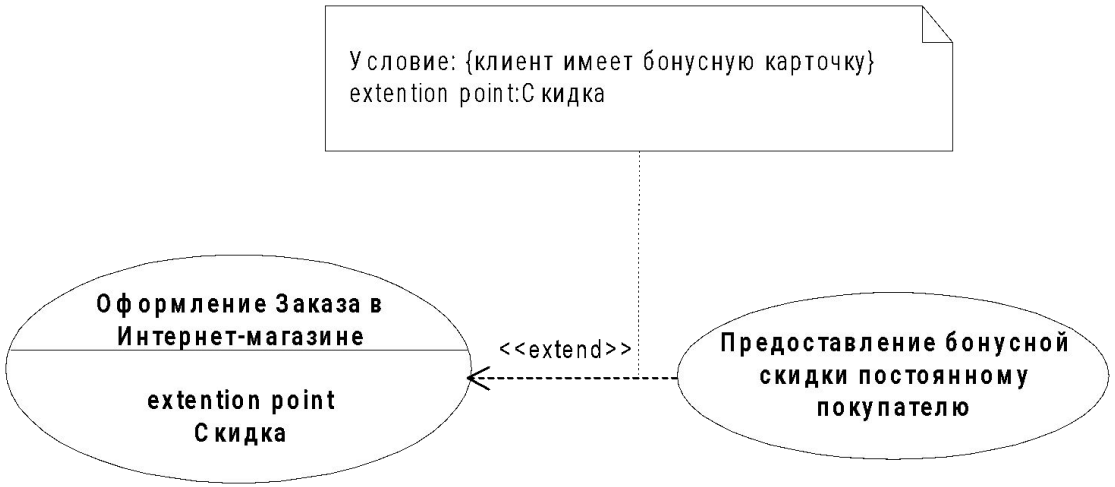
вариант использования *А*

вариант использования *Б*

Отношение расширения

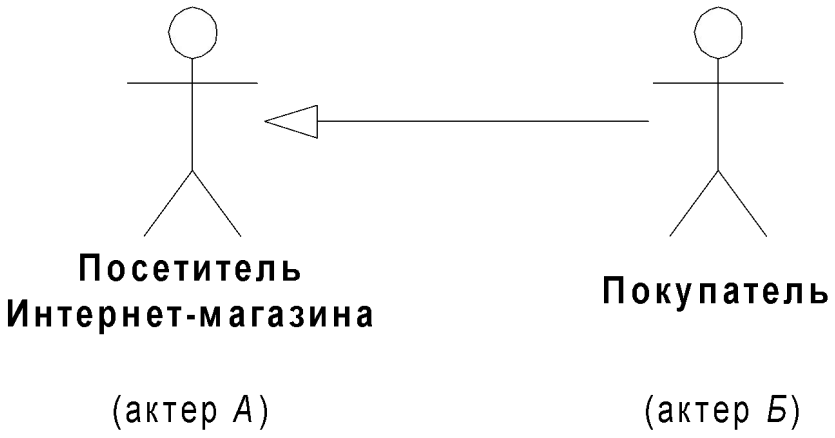
- Отношение расширения (extend) определяет взаимосвязь одного варианта использования с некоторым другим вариантом использования, функциональность или поведение которого задействуется первым не всегда, а только при выполнении некоторых дополнительных условий.





Отношение обобщения

- Отношение обобщения (generalization relationship) предназначено для спецификации того факта, что один элемент модели является специальным или частным случаем другого элемента модели



ПРАКТИКА

- Создать Use Case диаграмму для системы онлайн-экзаменов.
- Описание системы:
  - Экзаменатор перед запуском экзамена должен сперва подготовить банк вопросов (в частности, вопросы по Java и вопросы по C#)
  - Когда экзаменационные вопросы готовы, он может запустить экзамен, а также впоследствии отменить его в любой момент
  - Студент может сдать экзамен, только получив разрешения на старт работы от экзаменатора
  - Сдача экзамена включает в себя загрузку выполненной работы в систему
  - Студент может в любое время посмотреть результаты своих прошедших экзаменов