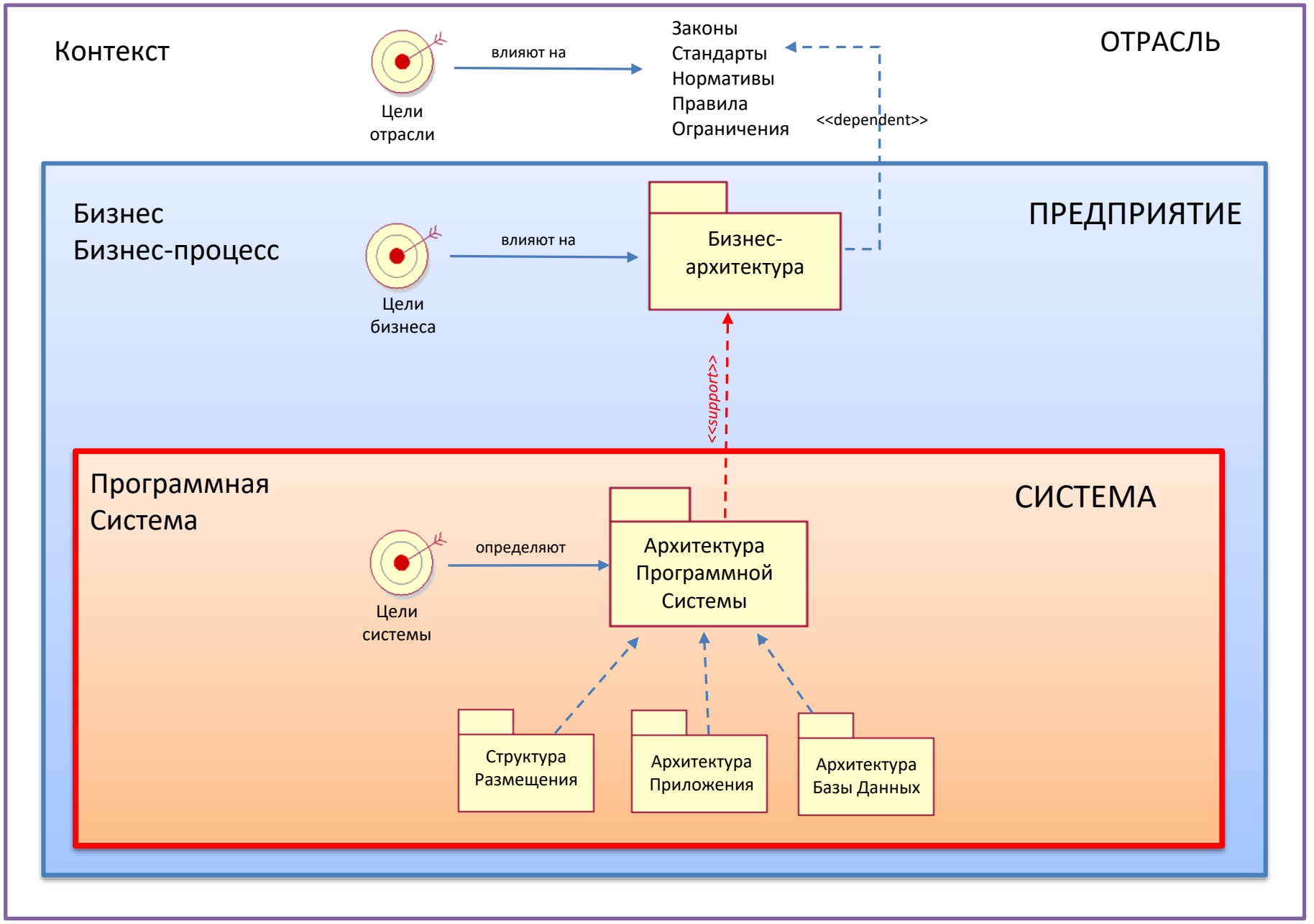
Что нужно знать и понимать, приступая к разработке любой программной системы?

Уровни: Контекст – Бизнес – Система



АРХИТЕКТУРА:

совокупность упорядоченных (по разным принципам) структур (иерархий) взаимосвязанных и взаимодействующих элементов

АРХИТЕКТУРА определяет ПОВЕДЕНИЕ

ПОВЕДЕНИЕ: отклик на запрос (воздействие)

ПОДХОД/Approach:

структура проекта соответственно с применяемой методологией

МОДЕЛЬ/Model:

упрощённое (абстрактное) представление структур и/или поведения в текстовом и/или графическом виде **Абстрагирование = упрощение**: процесс выявления основных свойств элемента

ПРЕДСТАВЛЕНИЕ (ВИД) МОДЕЛИ/View:

точка зрения на архитектуру, которая отражает её определённый аспект

НОТАЦИЯ (ЯЗЫК):

совокупность графических символов, используемых в графической модели, и правил их использования.

ПРОЕКТИРОВАНИЕ:

разработка и документирование архитектурных решений с помощью графических моделей

ИНСТРУМЕНТ:

программное средство (графический редактор или CASE-средство), позволяющее строить модели в определённой нотации

ИНСТРУМЕНТАРИЙ:

набор инструментов для реализации определённой методологии

CASE-средство:

(Computer Aided Software/System Engineering)

программное средство компьютерного проектирования, объединяющее графический редактор, средство поддержки структуры проекта и генератор программного кода

Проектирование бизнеса

Определение: бизнес-модель

Бизнес-модель -- механизмы и логика действий предприятия, направленные на получение прибыли. В рамках одной компании может использоваться несколько бизнес-моделей.

Бизнес-стратегия -- поведение компании для достижения её интересов и целей. Цели могут быть весьма обширными и не ограничиваться рамками конкретной бизнес-модели.

Бизнес-архитектура -- структура бизнеса, которая состоит из представленных выше элементов и дополненная другими жизненно важными частями: организационной иерархией, процессами и пр.

Определение: бизнес-модель

Бизнес-модель служит для описания основных принципов создания, развития и успешной работы организации [1]

Бизнес-модель отражает бизнес-архитектуру

[1] А. Остервальдер, И. Пинье «Построение бизнес-моделей»

Визуализация бизнес-модели

ПС
1 **Потребительские сегменты (ПС)**
Организация может обслуживать один или несколько потребительских сегментов.

ЦП
2 **Ценностные предложения (ЦП)**
Организация старается решить проблемы и удовлетворить потребности клиентов, делая им определенные ценностные предложения.

КС
3 **Каналы сбыта (КС)**
Ценностные предложения поступают к потребителю через информационные, дистрибьюторские и торговые сети.

ВК
4 **Взаимоотношения с клиентами (ВК)**
Взаимоотношения устанавливаются и поддерживаются всеми службами по работе с клиентами.

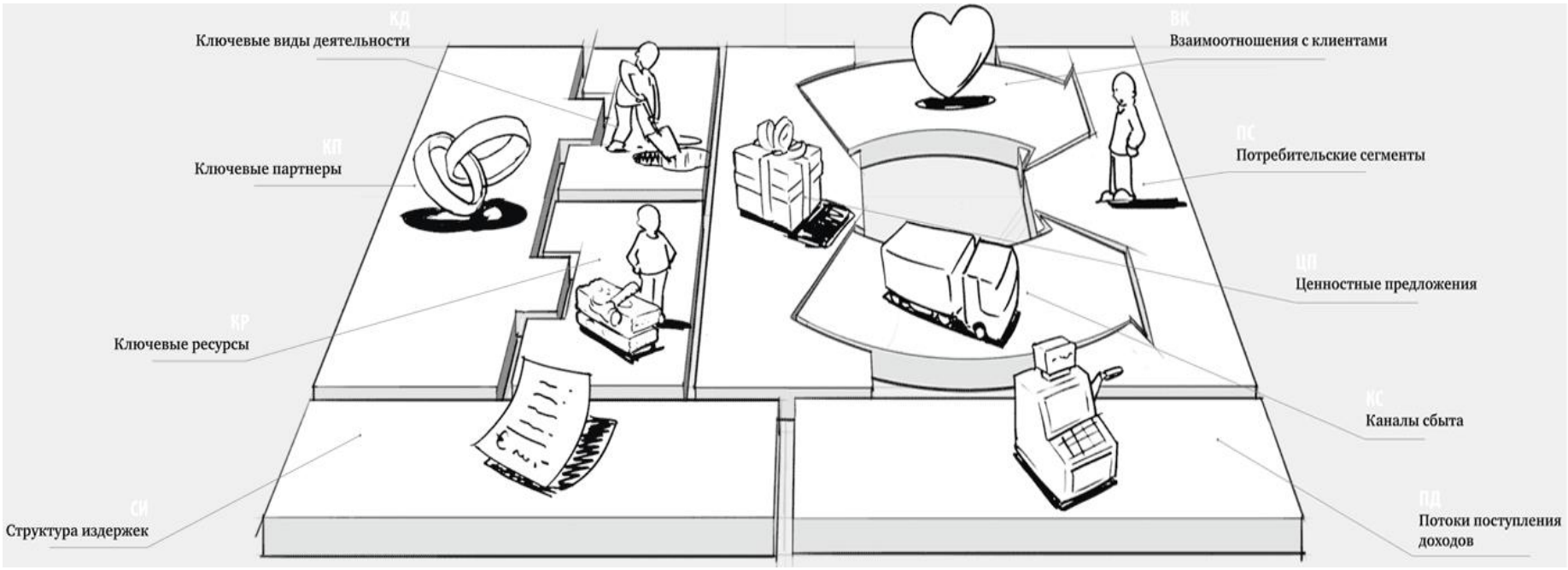
ПД
5 **Потоки поступления доходов (ПД)**
Потоки поступления дохода возникают благодаря успеху ценностных предложений, сделанных клиентам.

КР
6 **Ключевые ресурсы (КР)**
Ключевые ресурсы — средства, необходимые для предложения и доставки ранее описанных элементов…

КД
7 **Ключевые виды деятельности (КД)**
…посредством определенных ключевых видов деятельности.

КП
8 **Ключевые партнеры (КП)**
Какие-то виды деятельности передаются на аутсорсинг, а ресурсы — привлекаются со стороны.

СИ
9 **Структура издержек (СИ)**
Структура издержек — результат взаимодействия всех элементов бизнес-модели.



[1] А. Остервальдер, И. Пинье «Построение бизнес-моделей»

Разработкой бизнес-архитектуры должен заниматься Business или Enterprise Architect (бизнес-архитектор),

Business Analyst (бизнес-аналитик) отвечает за разработку бизнес-процессов

Проектирование программной системы

С чего начать разработку ПО?

Разработка ПО, *как правило*, **начинается с ТРЕБОВАНИЙ** (*функциональные и нефункциональные*)

Функциональные требования определяют архитектуру системы

Самыми общими (абстрактными) функциональными требованиями являются **ЦЕЛИ**

Цели бывают разные:

- Цели отрасли
- Цели бизнеса
- Цели программной системы

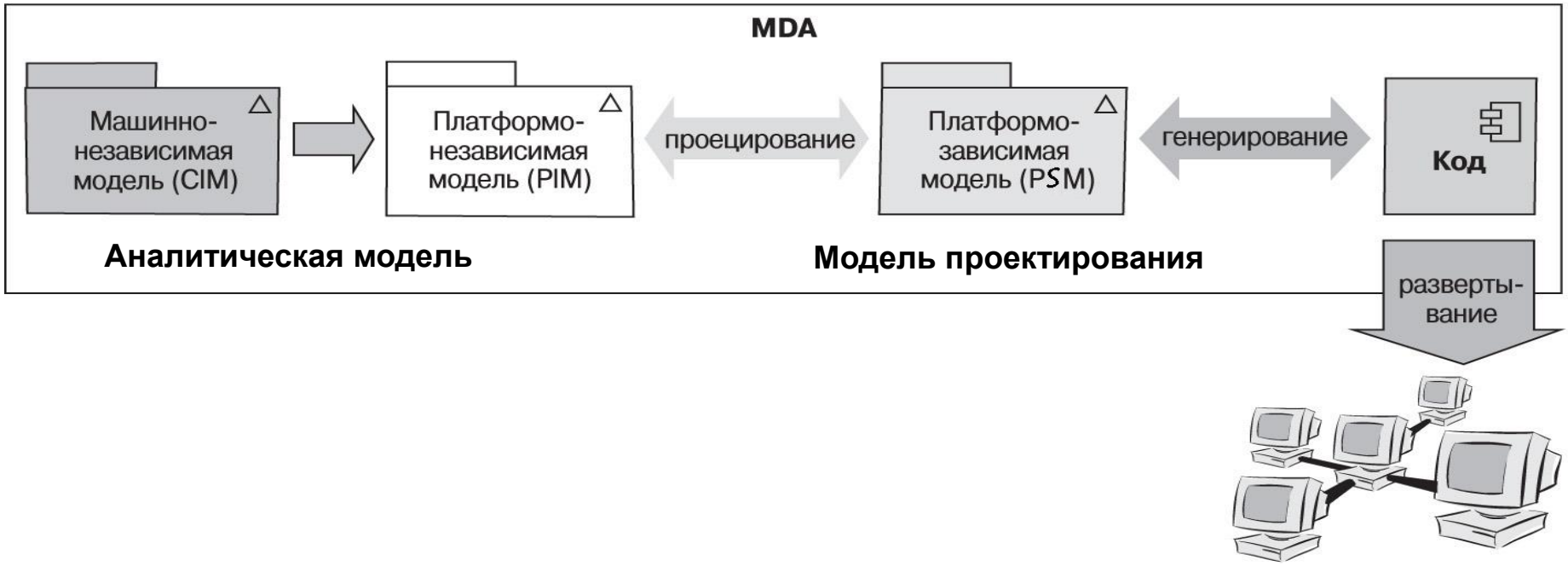
Пример. Оказание платных медицинских услуг
Цели: Контекст – Бизнес – Система

Цель	Пример	Уровень абстрактности	Необходимость	Артефакт
 Общие цели (цели отрасли)	Улучшение медицинского обслуживания населения	самый высокий	необязательны	текст Vision
 Цели Бизнеса	Получение прибыли от оказания платных медицинских услуг населению	высокий	желательны	текст Vision
 Цели ПС	• Учет платных мед.услуг, оказываемых поликлиникой населению; • Учёт врачей поликлиники, оказывающих плат.мед.услуги; • Учёт пациентов, пользующихся плаными мед.услугами; • Расчёт стоимости платных мед.услуг оказанных пациентам	**SMART**	**обязательны**	**текст SRS**

Назначение и основные элементы языка UML

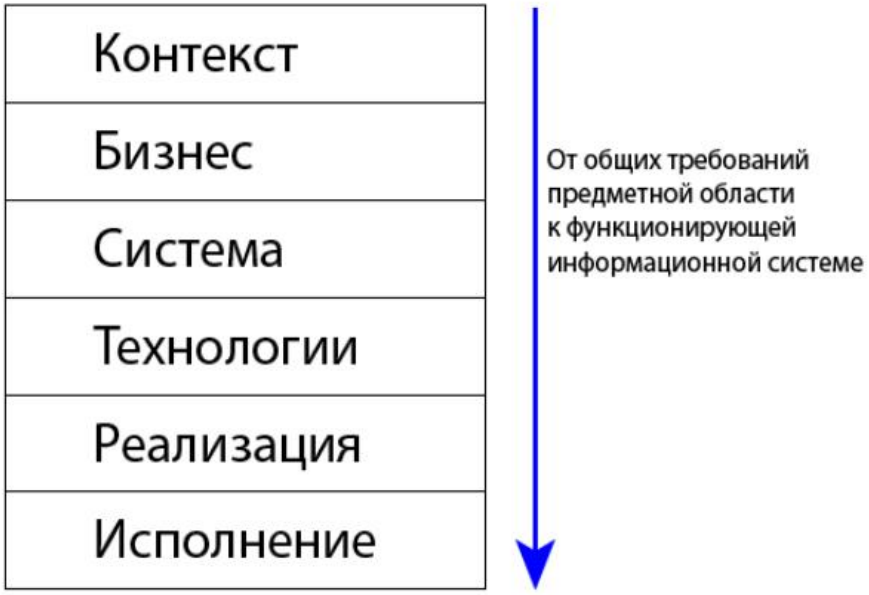
Принцип MDA (Model Driven Architecture)

MDA дает видение того, как разрабатывать программное обеспечение на основе моделей. Суть этого видения заключается в том, что модели управляют созданием исполняемой программной архитектуры. В настоящее время уже встречается подобный подход к разработке программного обеспечения, но MDA позволяет точно определить степень автоматизации данного процесса, чего до сих пор удавалось достичь довольно редко. В MDA создание программного обеспечения происходит в результате ряда трансформаций модели при поддержке инструмента моделирования MDA. Абстрактная машиннонезависимая модель (computer-independent model, CIM) используется как основа для платформонезависимой модели (platform-independent model, PIM). PIM трансформируется в платформозависимую модель (platform-specific model, PSM), которая преобразуется в код.

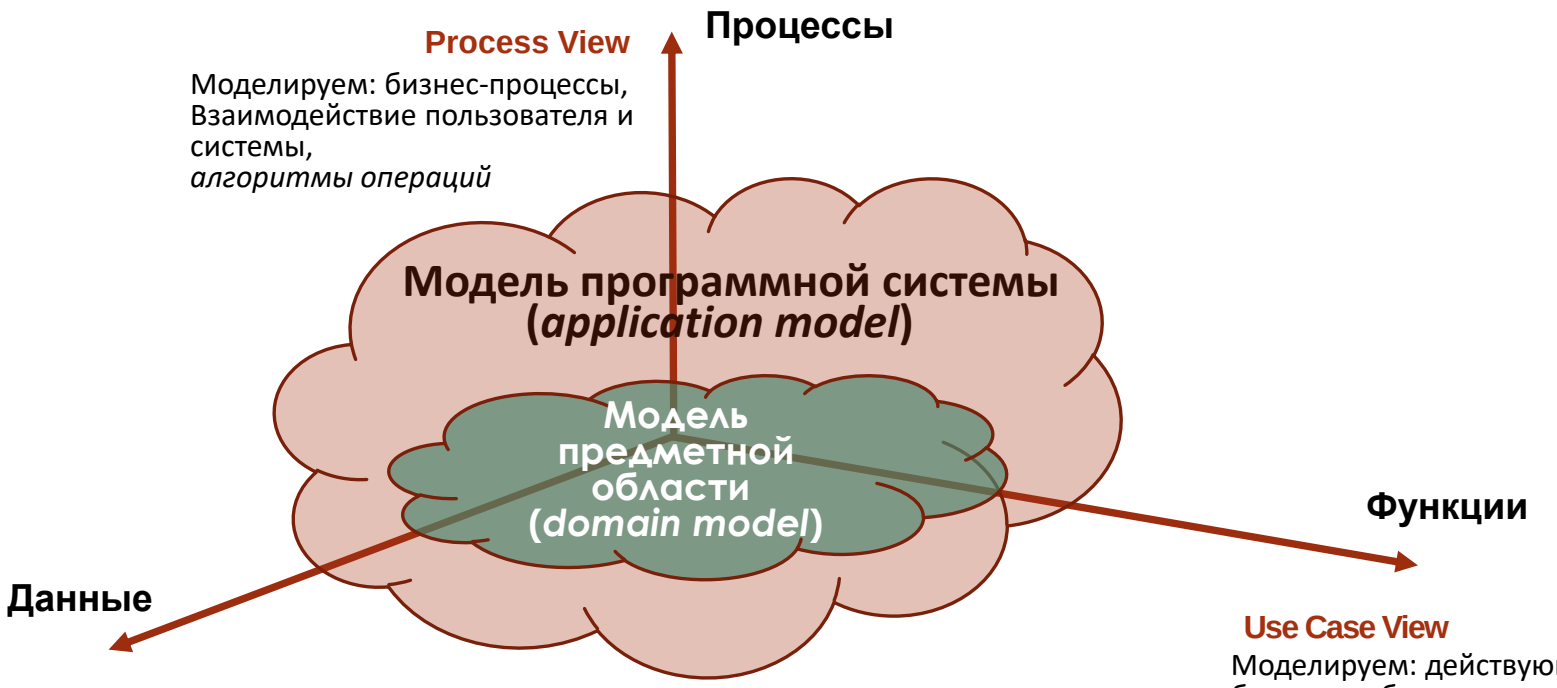


Фреймворк Дж. Захмана

Люди/организации достигают поставленных **целей**, исполняя возложенные на них **функциональные обязанности**, в определённом **месте**, в определённое **время** и оперируя конкретными **данными**



Фреймворк Дж. Захмана vs UML



+ Deployment View

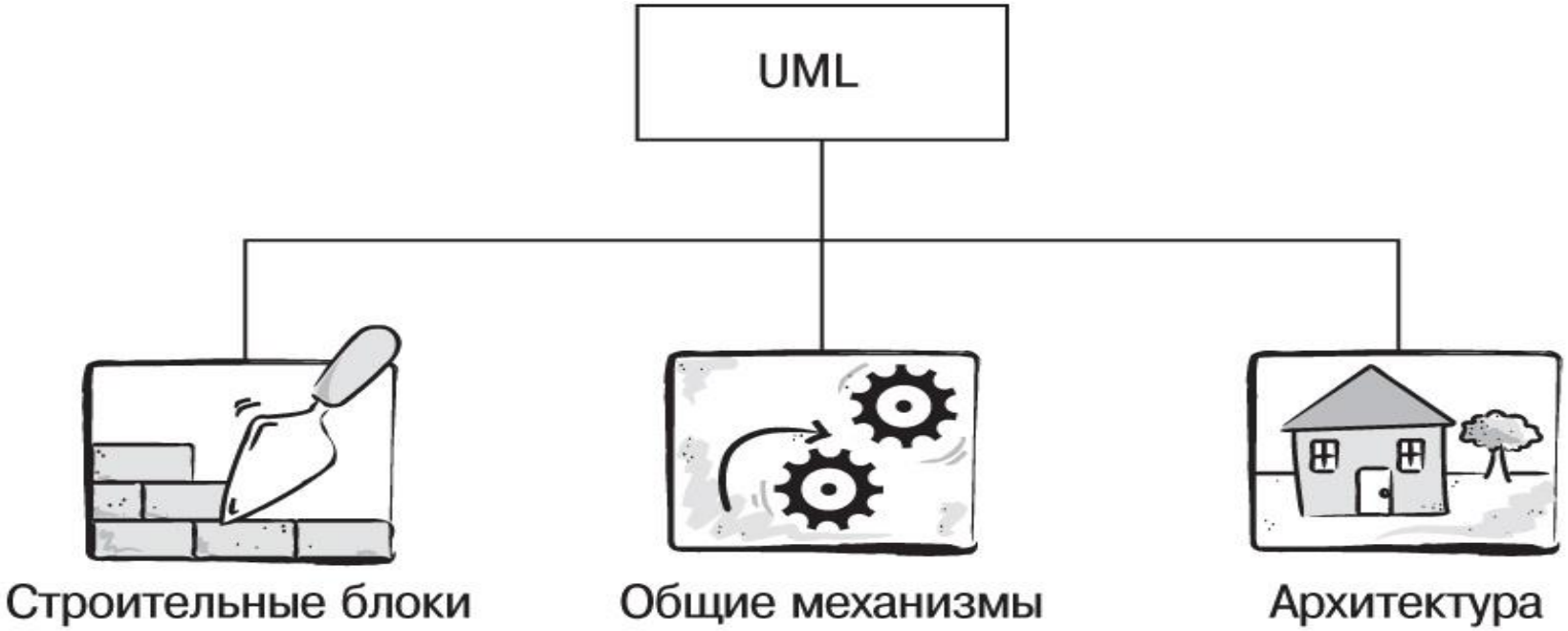
+ Component View

+ Timing

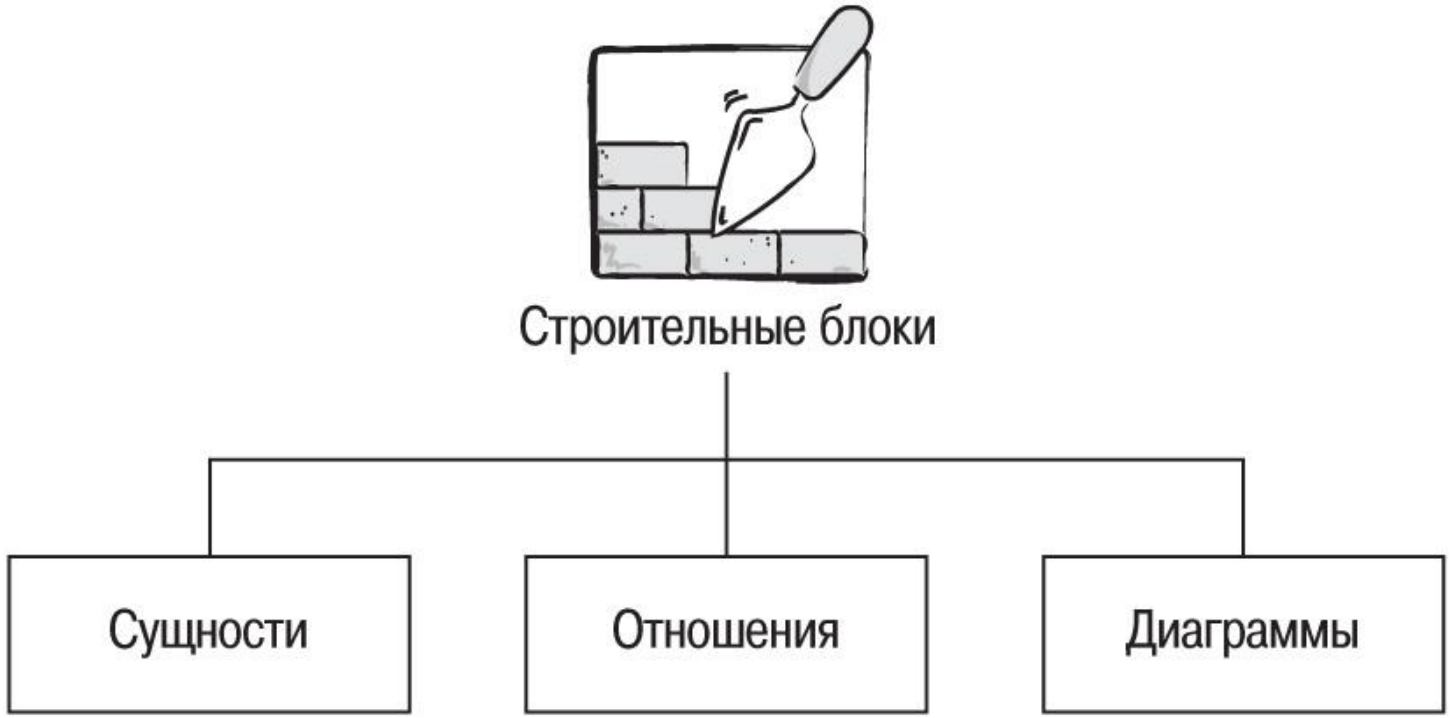
Goals

Назначение и основные элементы языка UML

Структура языка UML



- строительные блоки – основные элементы, отношения и диаграммы UML- модели;
- общие механизмы – общие UML-пути достижения определенных целей;
- архитектура – UML-представление архитектуры системы.



Все UML-сущности можно разделить на:

- **структурные сущности – существительные UML-модели, такие как класс, интерфейс, кооперация, прецедент, активный класс, компонент, узел;**
- **поведенческие сущности – глаголы UML-модели, такие как взаимодействия, деятельности, автоматы;**
- **группирующая сущность – пакет, используемый для группировки семантически связанных элементов модели в образующие единое целое модули;**
- **аннотационная сущность – примечание, которое может быть добавлено к модели для записи специальной информации, очень похожее на стикер.**

UML- 1,5 и 2

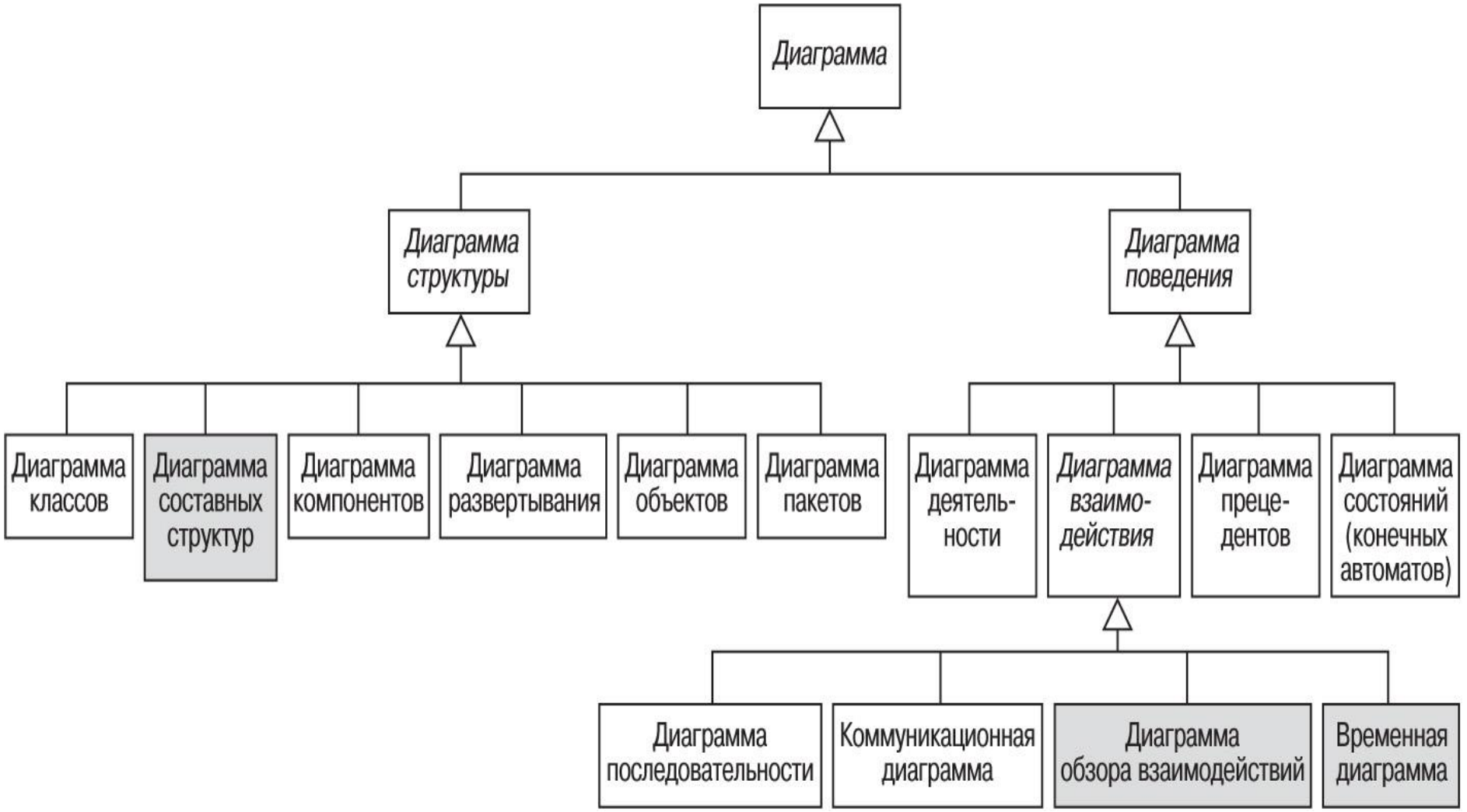
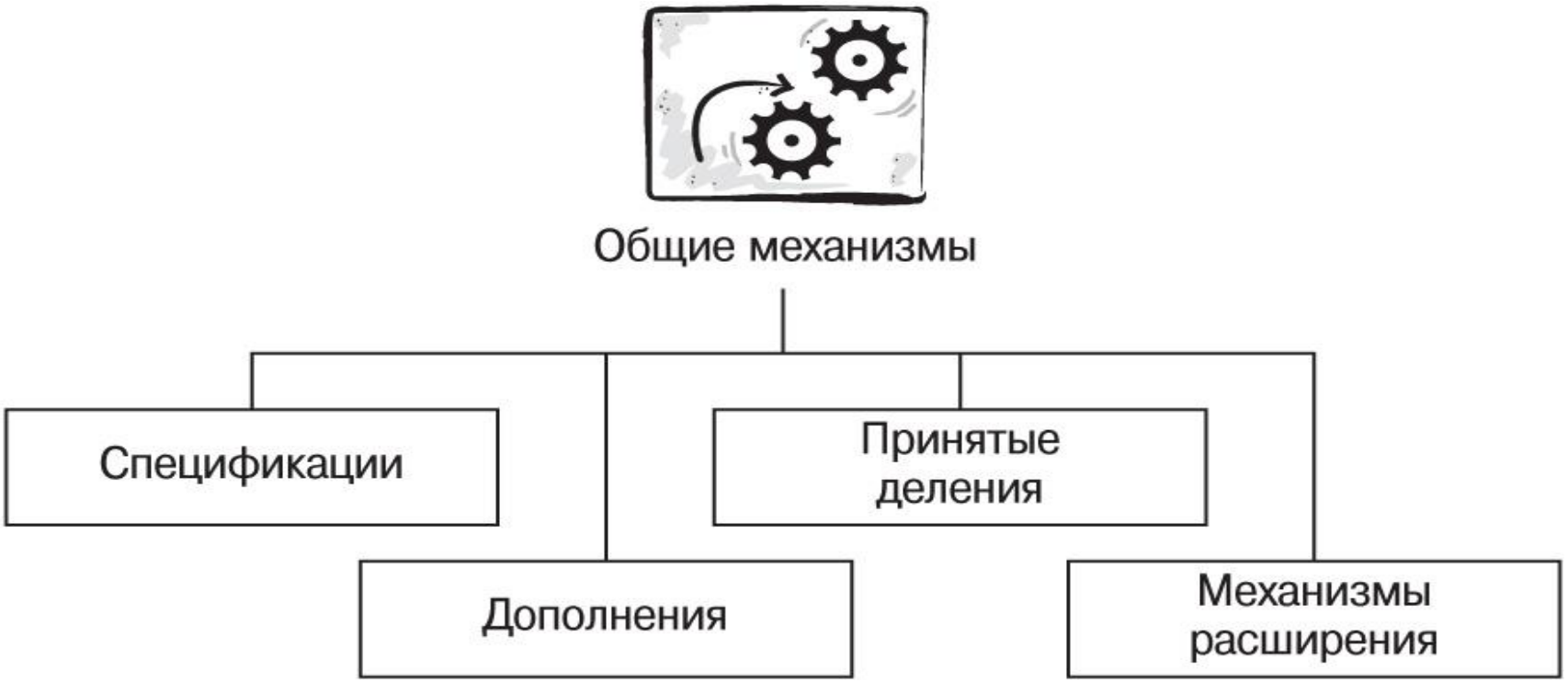


Диаграмма
Диаграмма структуры
Диаграмма поведения
Диаграмма классов
Диаграмма составных структур
Диаграмма компонентов
Диаграмма развертывания
Диаграмма объектов
Диаграмма пакетов
Диаграмма деятель-ности
Диаграмма взаимо-действия
Диаграмма преце-дентов
Диаграмма состояний (конечных автоматов)
Диаграмма последовательности
Коммуникационная диаграмма
Диаграмма обзора взаимодействий
Временная диаграмма

Общие механизмы UML



Спецификации – это текстовые описания семантики элемента

Различные детали базовой модели могут быть раскрыты в виде *дополнений*, расширяющих это минимальное представление

Принятые деления описывают конкретные способы представления мира. В UML существует два принятых деления: классификатор/экземпляр и интерфейс/реализация.

Назначение и основные элементы языка UML

Механизмы расширения

К механизмам расширения относятся:

- *«**стереотипы**» (stereotype)* – новый вид элемента модели, который обладает той же структурой, что и существующий элемент, но имеет дополнительные ограничения, другую интерпретацию и пиктограмму
- *«**именованные (помеченные) значения**» (tagged value*) – текстовая информация, прикрепляемая к любым типам элементов модели, представляющее пару «имя – значение». Имя в именованном значении называют тегом (tag).

Именованные значения на *диаграммах* изображаются в форме строки текста, заключенного в фигурные скобки. При этом используется следующий формат записи: {тег = значение}. Теги встречаются в нотации языка UML, но их определение не является строгим, поэтому они могут быть указаны самим разработчиком.

Моделирование на UML

Объектно-ориентированный подход

Принципы объектно-ориентированного подхода (ООП)

Объектно-ориентированная технология основывается на **объектной модели** (object model), которая, согласно Г.Бучу, базируется на четырех главных принципах (главных в том смысле, что без любого из них модель не будет объектно-ориентированной): *абстрагирование, инкапсуляция, модульность, иерархичность, полиморфизм.*

Абстракция *(abstraction)* – выделяет **существенные характеристики** некоторого объекта, которые отличают от всех других объектов и четко определяют его концептуальные границы для наблюдателя.
Абстрагирование – процесс выявления абстракций.
Абстракция определяет границу представления соответствующего элемента *модели* и применяется для определения фундаментальных понятий ООП, таких как *класс* и *объект*.

Инкапсуляция *(encapsulation)* – процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение.
Инкапсуляция характеризует сокрытие отдельных деталей внутреннего устройства классов от внешних по отношению к нему объектов или пользователей.
Инкапсуляция определяет четкие границы между различными абстракциями.
Абстракция и инкапсуляция дополняют друг друга: абстрагирование направлено на наблюдаемое поведение объекта, а инкапсуляция занимается внутренним устройством.

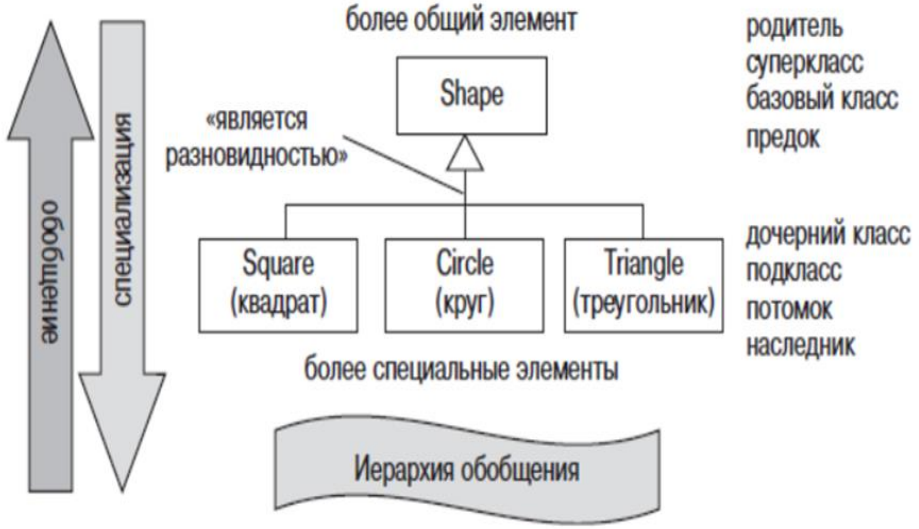
Модульность *(modularity)* – это свойство системы, которая была разложена на внутренние связанные, но слабо связанные между собой модули.

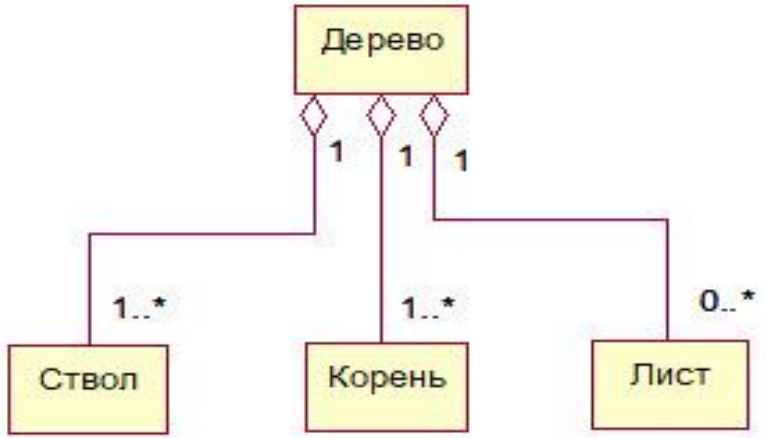
Принципы ООП

Иерархия *(hierarchy)* – упорядочение абстракций, расположение их по уровням.
Основными видами иерархических структур применительно к сложным системам являются:

структура классов «is-a». Отношение «обобщение / специализация» («наследование»)

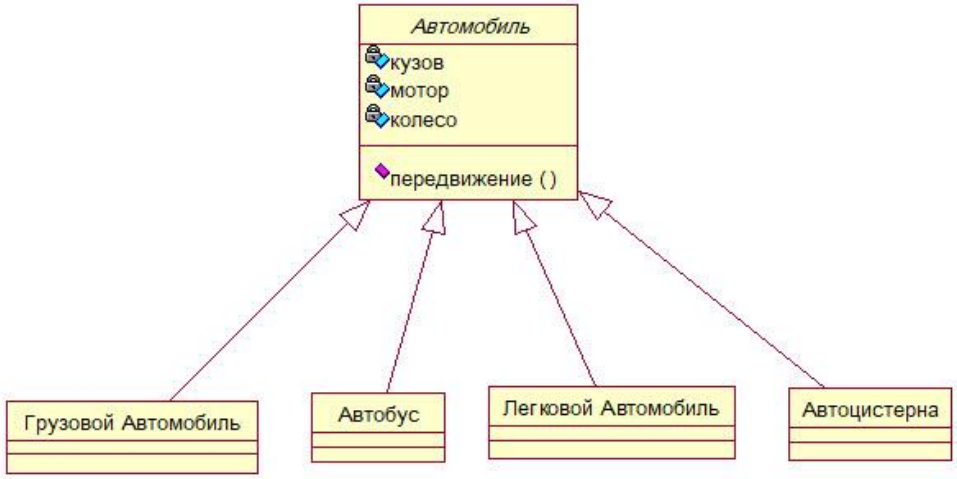
структура объектов (иерархия «part of» -- часть от или отношение агрегации)



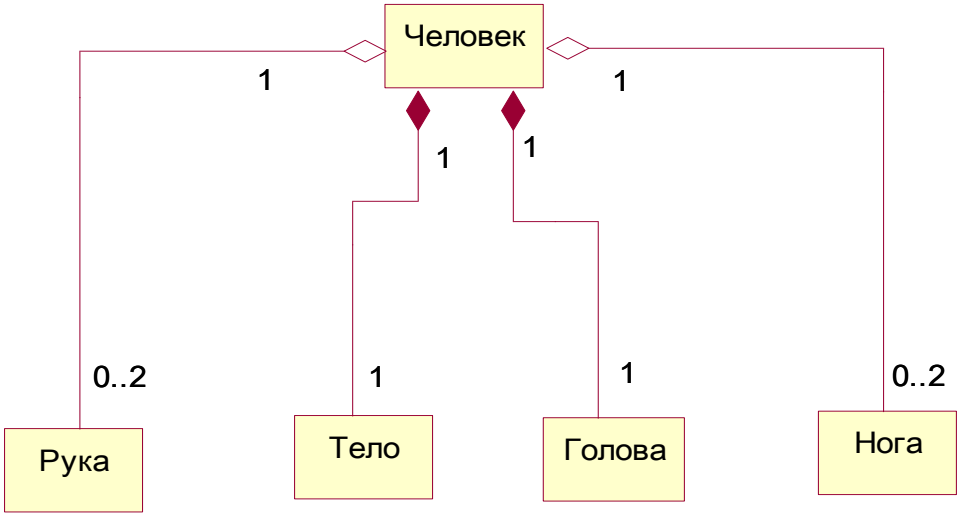


Полиморфизм *(polymorphism)* – наличие множества форм или реализаций конкретной функциональности
Под ***полиморфизмом*** (греч. Poly - много, morfos - форма) понимается свойство *объектов* принимать различные внешние формы в зависимости от обстоятельств.
С точки зрения ООА проявлением полиморфизма является то, что одну и ту же функциональность программной системы можно реализовать несколькими разными вариантами ее архитектуры.
Применительно к ООП *полиморфизм* означает, что действия, выполняемые одноименными методами, могут различаться в зависимости от того, к какому из *классов* относится тот или иной метод.

Примеры иерархий



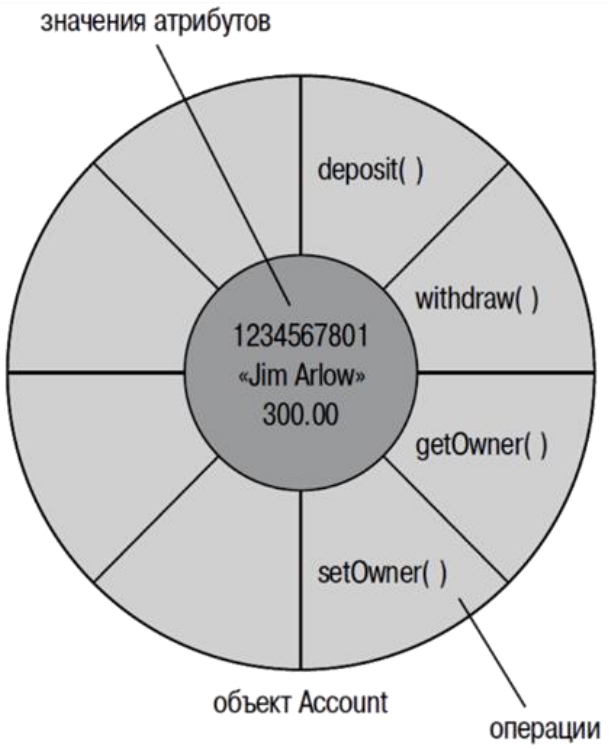
Пример: иерархия «общее – частное» (is a…)



Пример: иерархия «целое – часть» (part of…)

Классы и объекты

- ***Архитектура*** *(architecture)* **–** логическая (классы и их отношения) и физическая структуры системы (объекты, связи, сообщения), сформированная всеми стратегическими и тактическими проектными решениями. [Г. Буч, 2]

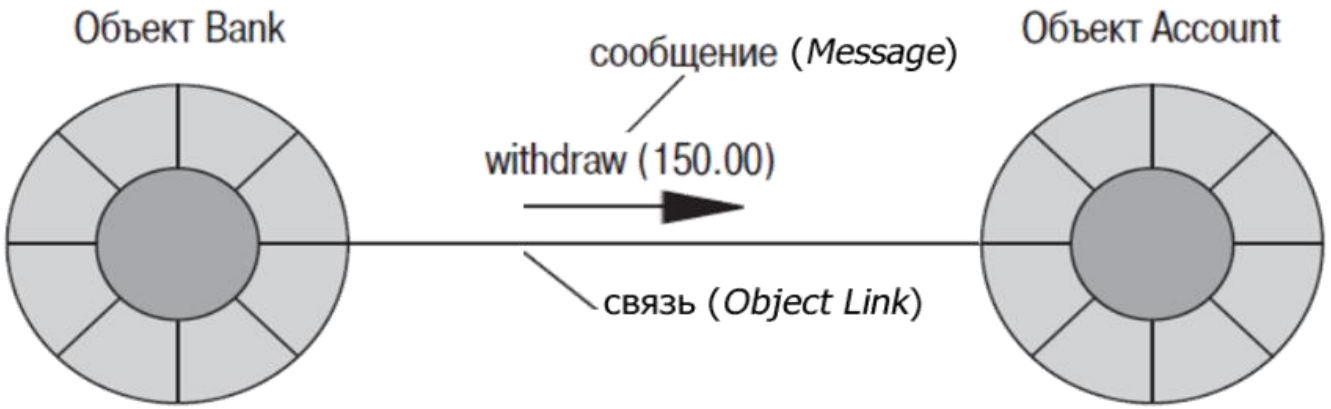


- *Объектно-ориентированная программная система – это совокупность взаимосвязанных взаимодействующих объектов, каждый из которых является экземпляром класса.*
- Объект объединяет (***инкапсулирует***) данные и функциональность в единый блок, это *сущность с явно выраженными границами, обладающая индивидуальностью, которая характеризуется состоянием и поведением;*
- *Класс* (class) – это *абстракция* совокупности реальных *объектов*, которые имеют общий набор свойств и обладают одинаковым поведением.
- *Классы взаимосвязаны между собой и являются членами иерархии наследования.*

Классы и объекты

- *В значениях атрибутов сохраняются данные (существенные характеристики), ими определяется состояние объекта.*
- *Единственный путь добраться до данных объекта – вызвать одну из предоставляемых им функций. Эти функции называются операциями (operations).*
- ***Функциональное поведение программной системы реализуется благодаря взаимодействию объектов, при котором объект «клиент» посылает сообщение объекту «сервер» с целью вызова соответствующего метода последнего.***

Модель взаимодействия объектов «Клиент – Сервер»

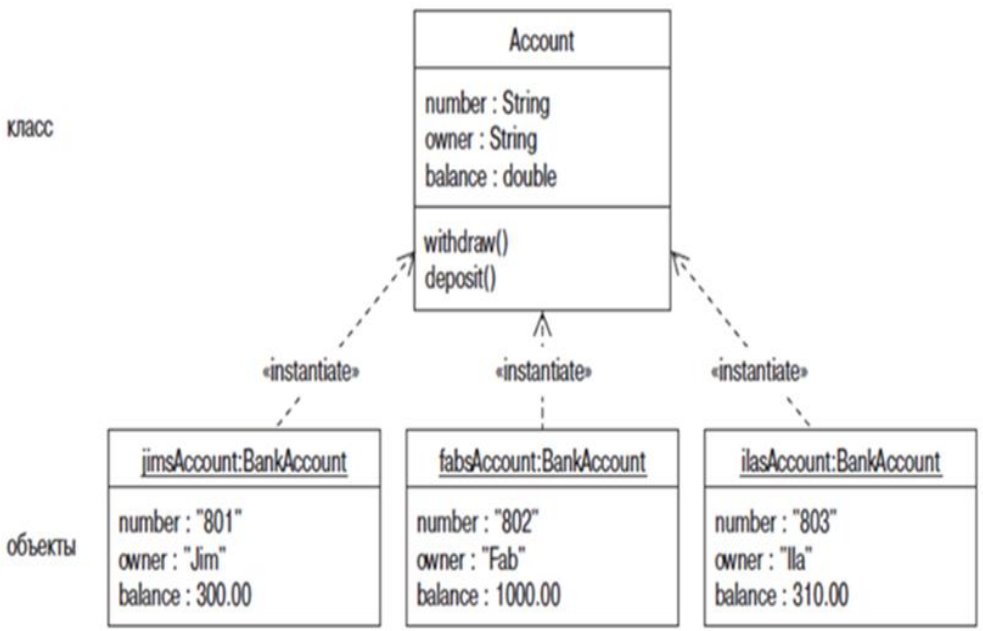


Объект Bank посылает сообщение «withdraw 150.00» объекту Account

В ответ объект Account вызывает свою операцию withdraw, которая уменьшает баланс счета на 150.00

Классы и объекты

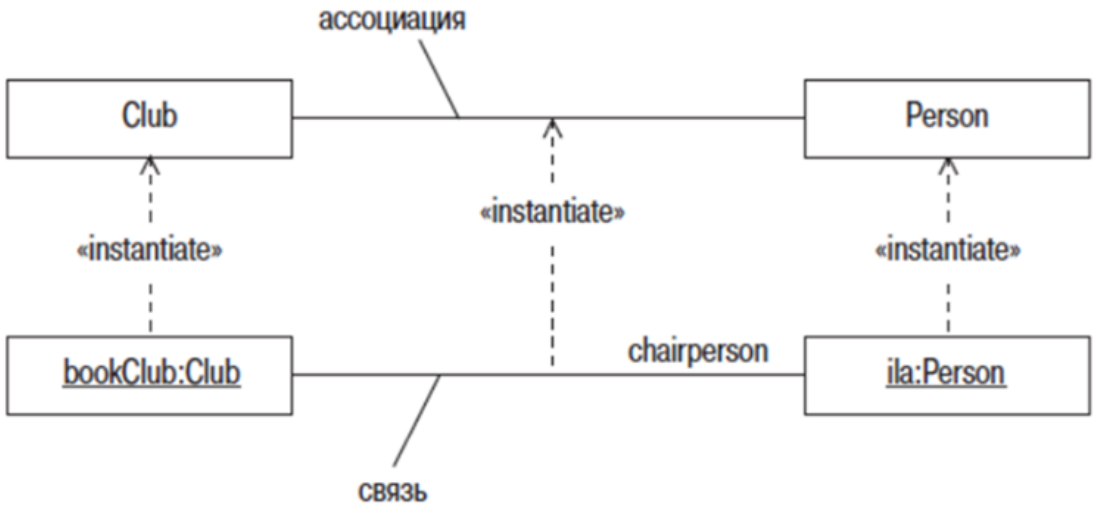
- Каждый объект имеет уникальный идентификатор.
- Поведение объекта – это то, «что он может сделать для нас», т. е. его методы.
- Объекты формируют поведение системы путем обмена сообщениями по связям. Это – кооперация.
- Каждый объект – это экземпляр только одного класса.
- Класс описывает свойства группы однотипных объектов.
- Первоначально, классы создаются на основе сущностей (объектов) предметной области.



- При создании экземпляра класса создается новый объект. Класс используется как шаблон.
- Экземплярами класса могут быть объекты, другие классы, класс может не иметь экземпляров (абстрактный класс).
- Объект – это экземпляр класса, связь между объектами – экземпляр отношений классов, каждое сообщение это вызов метода, метод – это реализация операции класса.

Классы и объекты

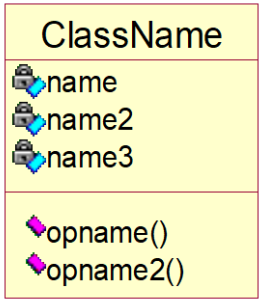
- Имя класса отражает его назначение.
- Класс является четкой абстракцией, моделирующей один конкретный элемент предметной области.
- Обязанности класса описывают связный набор операций. Обязанность – это контракт или обязательство класса по отношению к его клиентам. По существу, обязанность – это сервис, который класс предлагает другим классам. Обязанности сущностей, в том числе классов, в UML могут быть разграничены с помощью ***стереотипов***.
- *Объектно-ориентированный анализ – это метод анализа, исследующий требования к системе с точки зрения классов и объектов, относящихся к словарю предметной области.*
- Сущности со стереотипом ***business entity*** *моделируют группу однотипных объектов предметной области*. Они являются основными кандидатами в классы программной системы.



Моделирование классов в UML

простое отображение класса без стереотипов, атрибутов и операций

Модель проектирования (ООР)



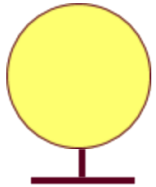
основное отображение класса в виде типового элемента (класс проектирования)

Модель бизнес-анализа (BA)

business entity (стереотип) моделирует группу однотипных объектов предметной области

Модели системного анализа (ООА)

entity (стереотип) – класс программной системы, отображение некой сущности предметной области. Основное назначение – хранение данных



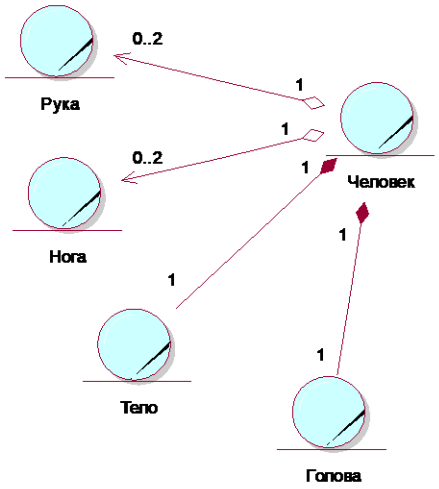
bounary (стереотип) – класс программной системы, представляющий интерфейс между внешним пользователем (actor) и программной системой

control (стереотип) – класс программной системы, отображающий элементы управления

Отношения классов

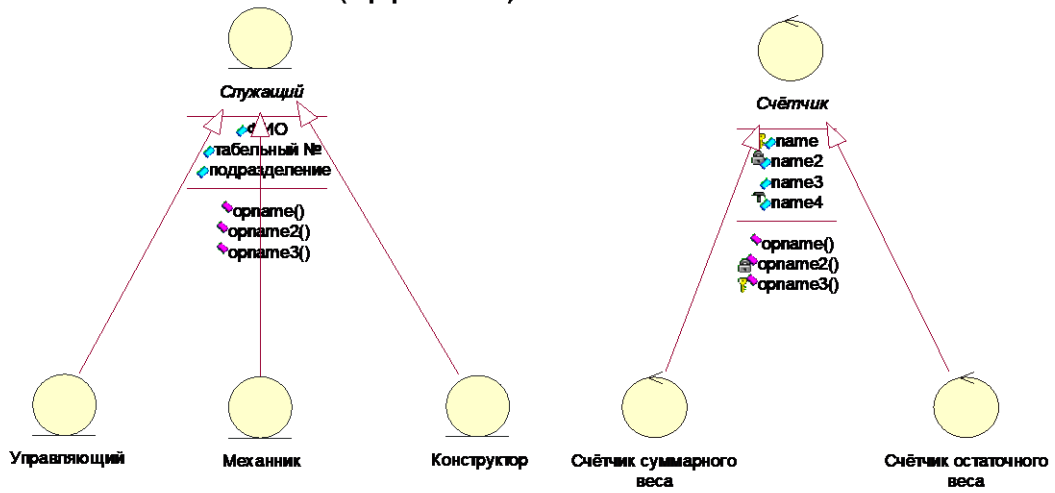
Агрегация и композиция одно- и двунаправленные

(иерархия "part of...")



Обобщение (Generalization)

(иерархия "is a...")



Управление экспортом:

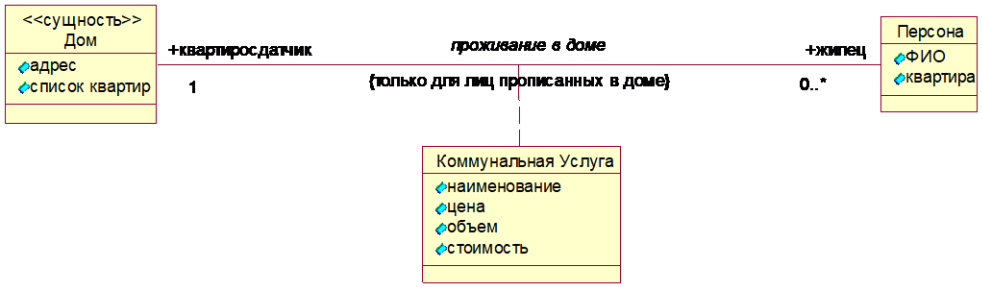
Public (Общий) – доступ к атрибуту/операции осуществляется из всех остальных классов;

Private (Закрытый) – доступ к атрибуту/операции невозможен ни из какого другого класса;

Protected (Защищённый) – доступ к атрибуту/операции возможен только из самого класса и его потомков;

Package or Implementation (Пакетный) – доступ осуществляется только из классов того же пакета.

Ассоциация одно- и двунаправленная



Используется если:
1. Операции класса A необходим параметр класса B.
2. Операция класса A возвращает значение класса B.
3. Операция класса A где-то в своей реализации использует объект класса B, но не в качестве атрибута.