Курс лекций по дисциплине «Программная инженерия»

Лекция 4. Архитектура ПО

Ст. преподаватель кафедры ИС
Важдаев А.Н.

Обсуждение

- Задачи backend сложные, ресурсоемкие, выполняются пакетно. Они отделены от графического интерфейса продукта (frontend), который также непросто устроен.
- Frontend – это пользовательский интерфейс: сложный, параметризуемый, с рядом встроенных пользовательских сервисов (в частности, браузер информации), а также настраиваемый.
- Обе эти подсистемы взаимодействуют друг с другом через хорошо определенный и детально описанный программный интерфейс: алгоритмы backend разбиты на методы, которые frontend может вызывать по особым правилам, с параметрами, выстраивая в цепочку для достижения своих задач.

Определение

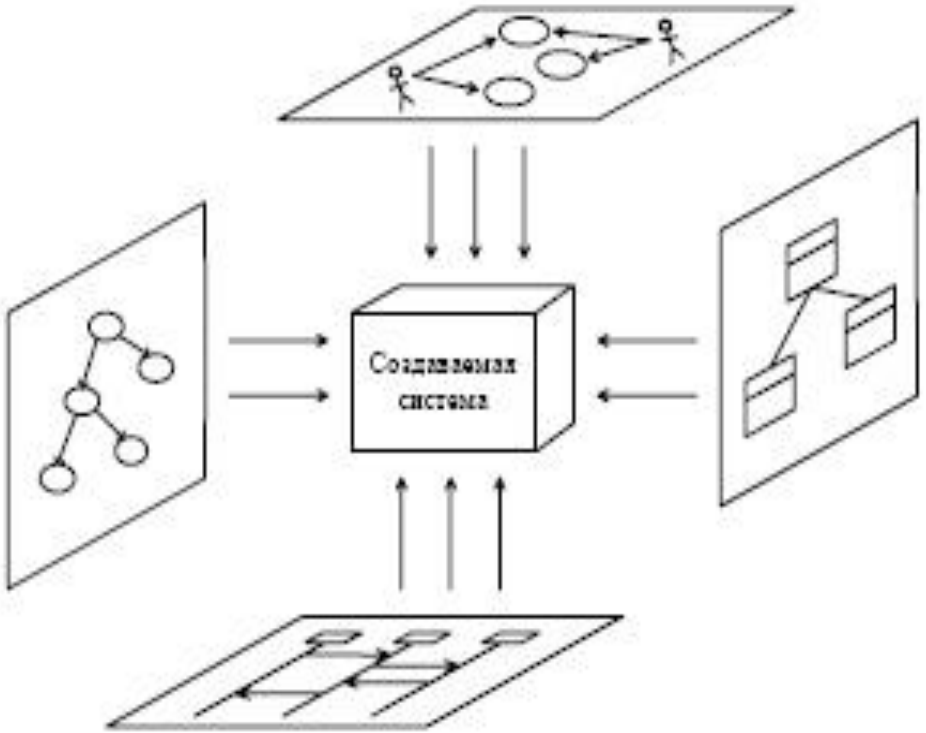
- **Архитектура ПО** - внутренняя структура продукта (компоненты и их связи), основы пользовательского интерфейса продукта, а также квинтэссенцию знаний и решений, являющихся инструментом разработки и управления проектом.
- То есть архитектура – это сквозная концепция или набор таковых для преодоления энтропии и хаоса, стремящихся "проглотить" разработку в виду сложности, нематериальности, согласовываемости и изменчивости ПО.

Множественность точек зрения

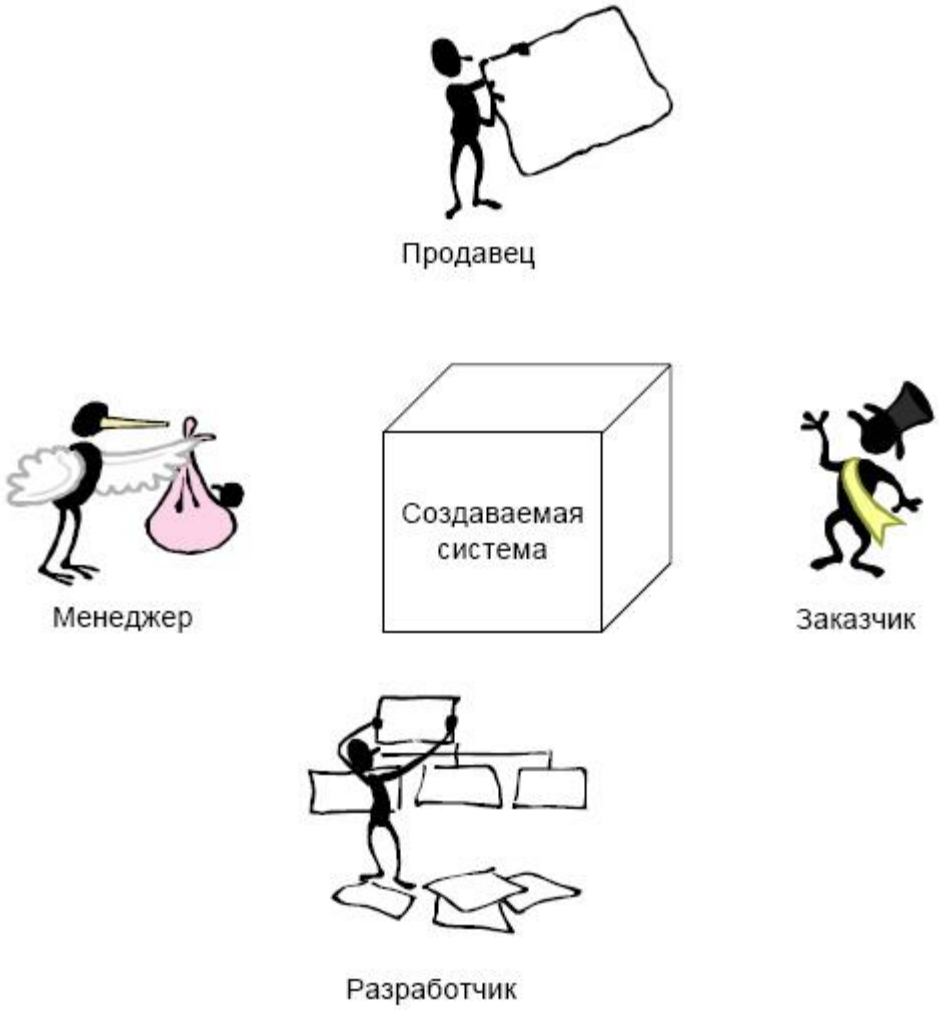
- При разработке архитектуры ПО важным оказывается совмещение множества точек зрения. ПО оказывается настолько сложным, что его архитектуру не построить как единую модель – множество отдельных аспектов должны быть представлены в архитектуре, их связи сложны и плохо выразимы в явном виде.
- **Причина множественности точек зрения при разработке ПО**. Умение рассматривать предмет с разных точек зрения является важнейшей философией успешной практики при работе с большими объемами разнородной и сложной информации.

Множественность точек зрения

Разработчик



Множественность точек зрения



Множественность точек зрения

- **Точка зрения** (viewpoint) — это определенный взгляд на систему, который осуществляется *для выполнения какой-то определенной задачи кем-либо из участников* проекта. Точку зрения нужно ясно осознавать при создании визуальных моделей, например, варианты использования. Важно понимать, что она может быть в каждом конкретном случае своя.
- Важнейшими характеристиками точки зрения моделирования является **цель** (зачем создается модель) и **целевая аудитория** (то есть, для кого она предназначается).

Язык UML

- Часто понятие архитектуры сильно сужают, понимая под ним лишь описание основных, важных аспектов ПО, создаваемых, например, архитектором при разработке дизайна системы. Для этих целей используется язык моделирования UML (Unified Modeling Language).
- "Скелетом" UML является диаграммная структура. Каждый вид диаграмм является типом моделей, реализующим определенную точку зрения на программную систему. Виды диаграмм не являются строго обязательными в UML – их можно перемешивать, создавать свои собственные виды диаграмм.

Язык UML

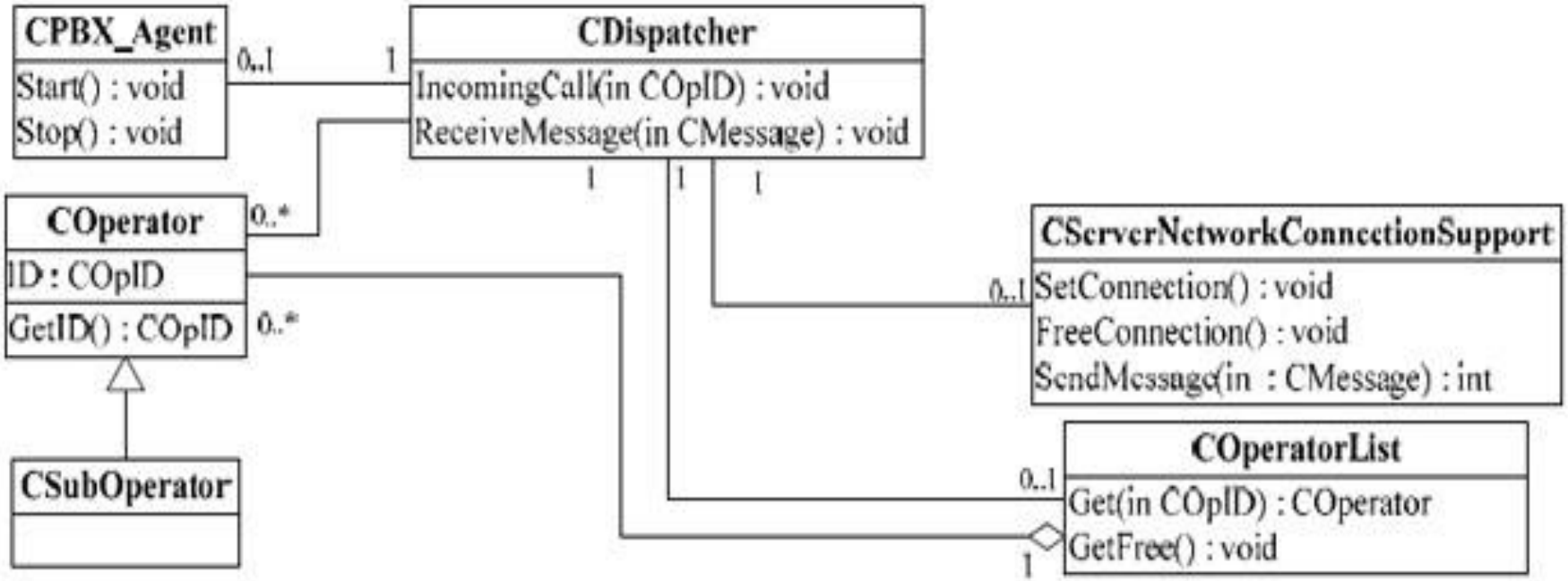
- Структурные диаграммы:
  - диаграммы классов (class diagrams);
  - диаграммы компонент (component diagrams);
  - диаграммы объектов (object diagrams);
  - диаграммы композитных структур (composite structure diagrams);
  - диаграммы развертывания (deployment diagrams);
  - диаграммы пакетов (package diagrams).

Язык UML

- Поведенческие диаграммы:
  - диаграммы активностей (activity diagrams);
  - диаграммы случаев использования (use case diagrams);
  - диаграммы конечных автоматов (state machine diagram);
  - диаграммы взаимодействий (interaction diagram):
    - диаграммы последовательностей (sequence diagram);
    - диаграммы схем взаимодействия (interaction overview diagram);
    - диаграммы коммуникаций (communication diagrams);
- временные диаграммы (timing diagrams)

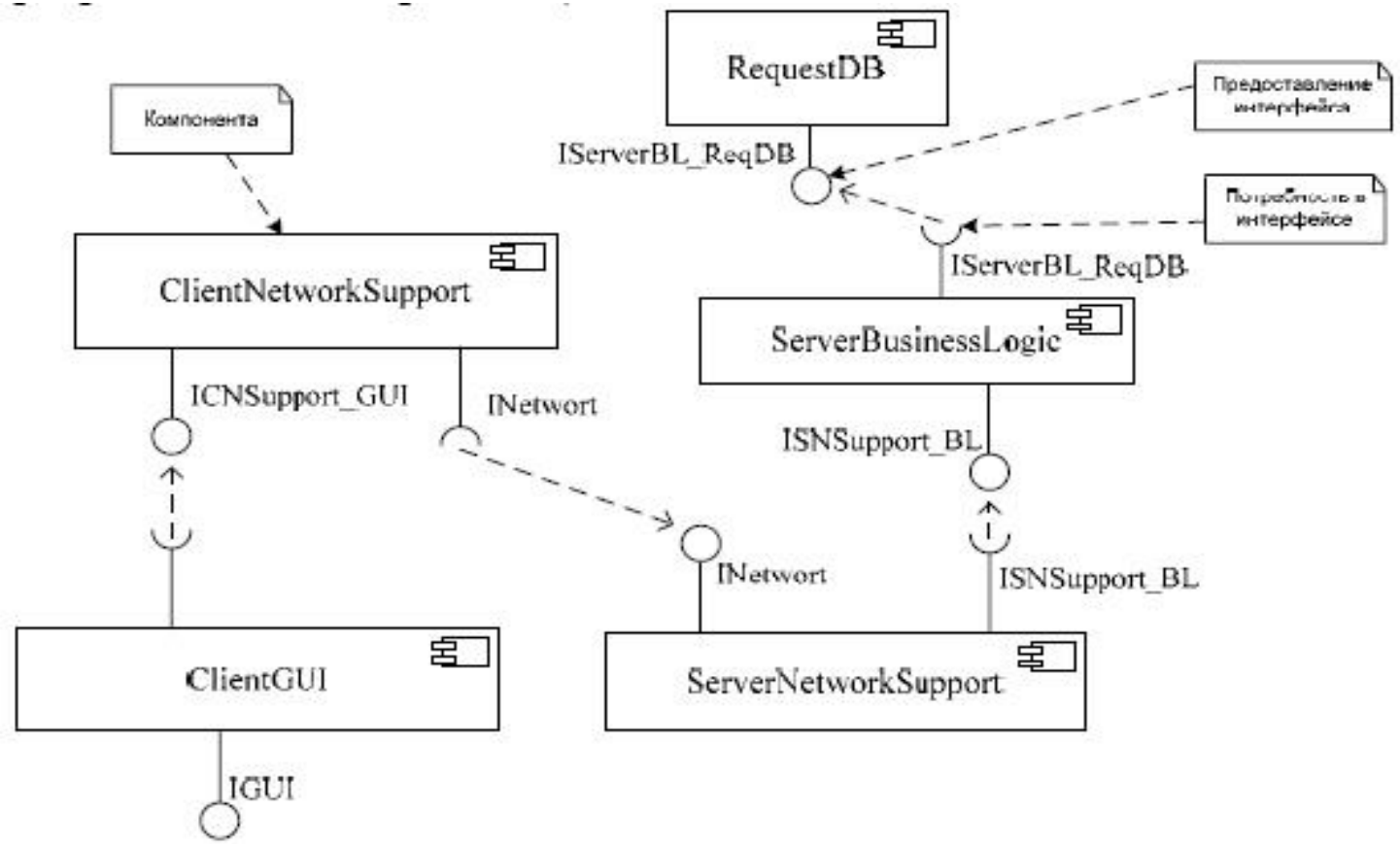
Язык UML

- Пример диаграмм классов



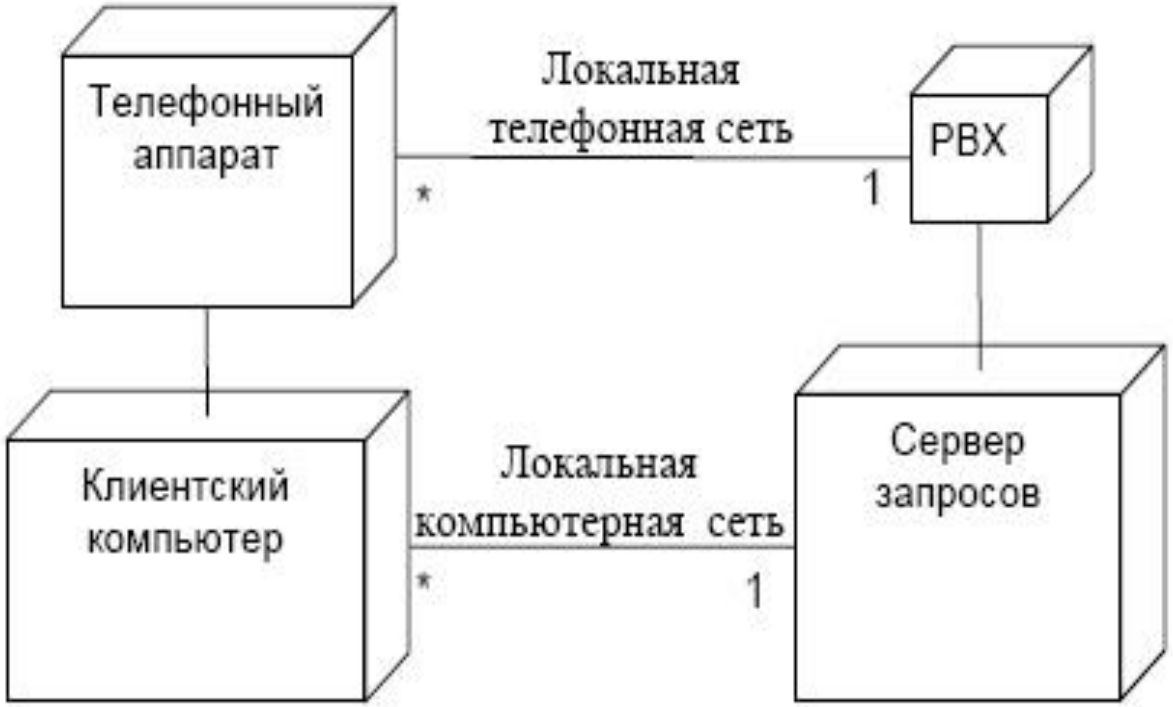
Язык UML

- Пример диаграмм компонент



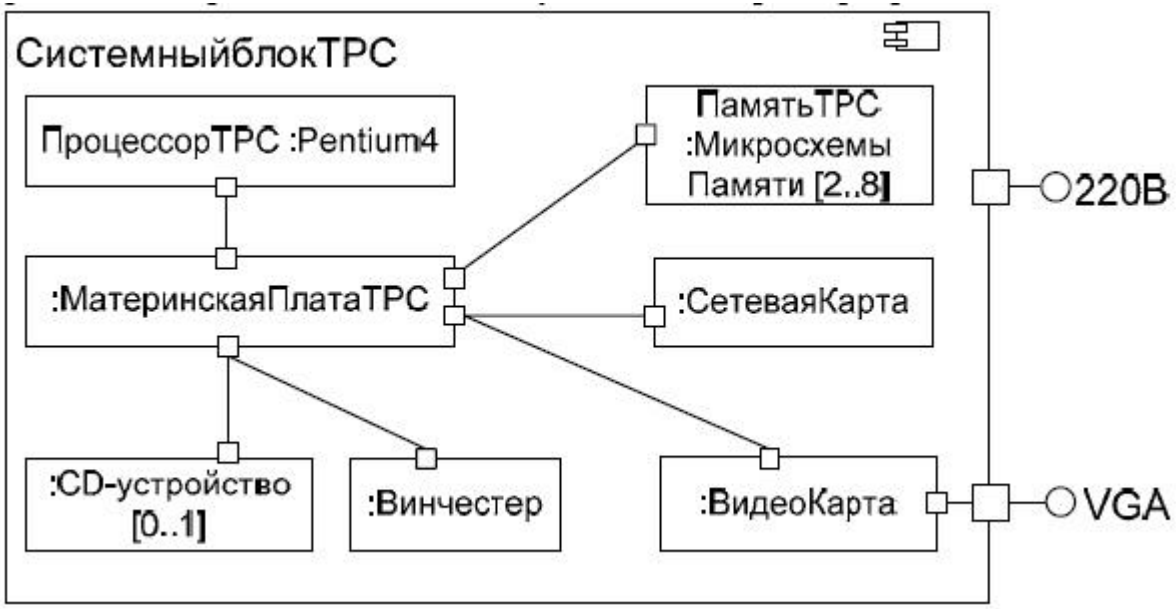
Язык UML

- Пример диаграмм размещений



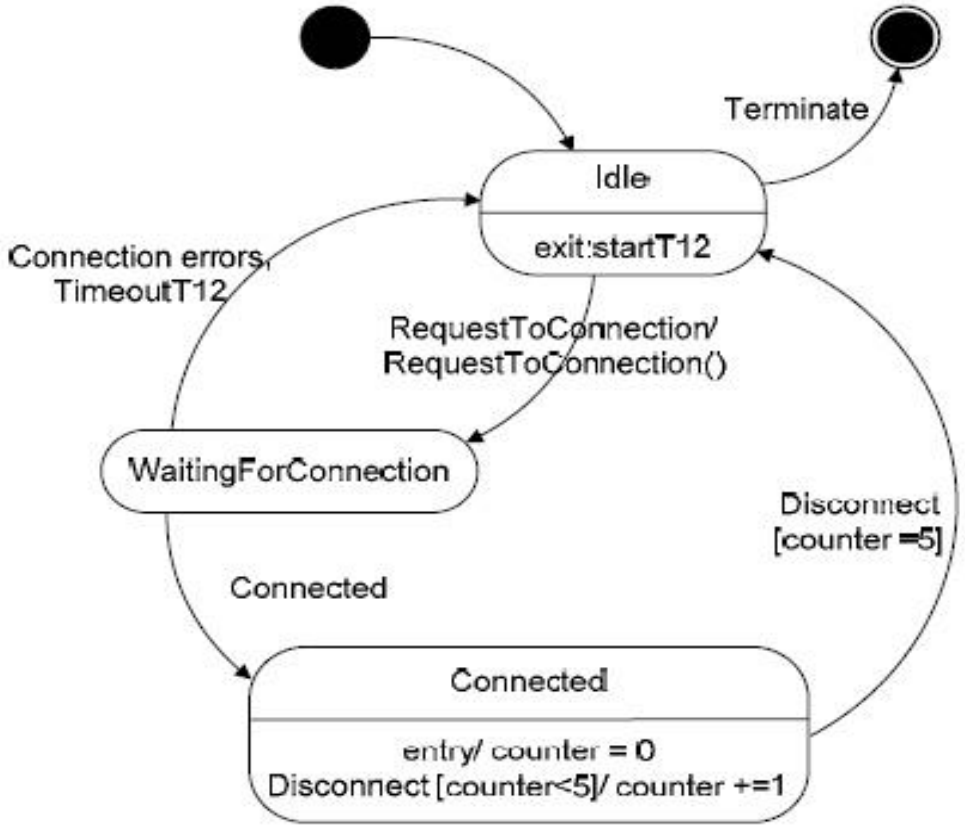
Язык UML

- Пример диаграмм композитных структур



Язык UML

- Пример диаграмм конечных автоматов



Язык UML

- Пример диаграмм последовательностей