Tuesday, April 19, 2022 11:02 AM

Лекция 11

Архитектурные стили

— Это набор типовых архит. решений с определенным именованием.

Задаются:

① контекст применения

② ограничения

③ полезные (и нет) свойства системы.

1 "Античные" традиционные стили

- основная пр-ма main и подпрограммы
  процедурное ПО
- ОО ПО

2 Слоистая архитектура

- слои
- вирт. машина
- клиент-сервер
- модель-предстал.-контроллер (MVC) и разновидности

3 Архитектура потоков данных (data flow)

- послед-ная пакетная
- каналы и фильтра (Pipe-and-filter, Pipeline)

4 Общая память

- Blackboard
- Rule-based

5 Интерпретатор

- Система интерпр-в
- Моб. приложения

6 Неявный вызов

- подписчики и трансляторы
- Event-based

7 Сетевые равновые

[illegible]

8 Сервис-ориент. системы

- микросервисы, REST

— Пир. арх-ра (Peer-to-Peer)

$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{Стили, которые мы рассмотрим}}$

! Часто проект содержит эл-ты разных арх. стилей.

1 Античные стили

① Main и подпрограммы

* Обеспечивает min декомпозицию сложности
* Модули — это процедуры
* Взаимодействие — вызовы
* Связность модулей сильная

Декомп.:	функц-ная
Компоненты:	пр-мы и подпр-мы
Коннекторы:	вызовы ф-ций и процедур
Эл-ты данных	значения, передаваемые при вызове
Топология:	статическая, иерархическая, граф

Ограничения: нет

Обеспечивает: модульность — замена процедур с разными реализациями, но едиными инт-сом.

Применение: небольшие пр-мы, и небольшие

Прим... скрипты, у ...

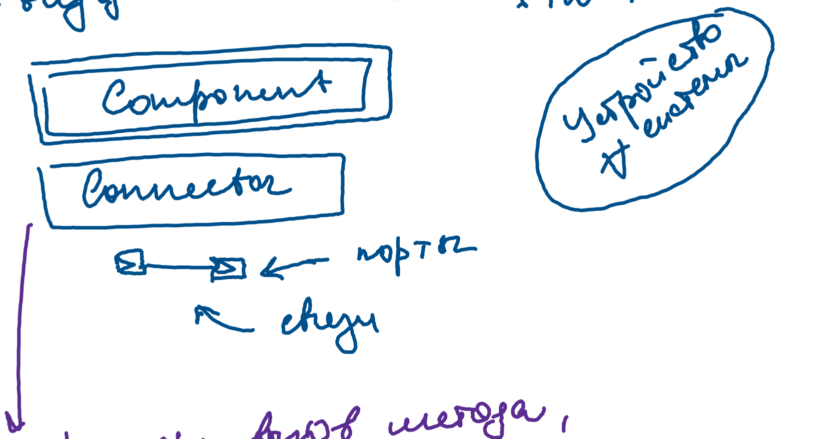
Опасности: плохо масштаб-ся, сложные стр-ры / данных

Трудно прогноз-ть затраты

Технологии: C, Pascal, Basic

Язык xADL

- визуальное представление в ПО Arch Studio



Пример: вызов метода, не класс

② ОО ПО

- Эл-ты декомп: объекты + концепции модулей / компонентов
- Взаим-вие – вызовы и отправка сообщений
- Объекты отвечают за своё внутр. сост.
- Устройство объектов скрыто

+ декомп. сложности
+ независимо проект-ть объекты
- побочные эффекты при вызове методов

– объекты сильно связаны

Декомп: объектная

Комп-ты: объекты (экз-ры классов)

Коннекторы: вызовы, сообщения

Эл-ты данных: аргументы методов

Топология: динамич-кая, иерарх-кая, наследование

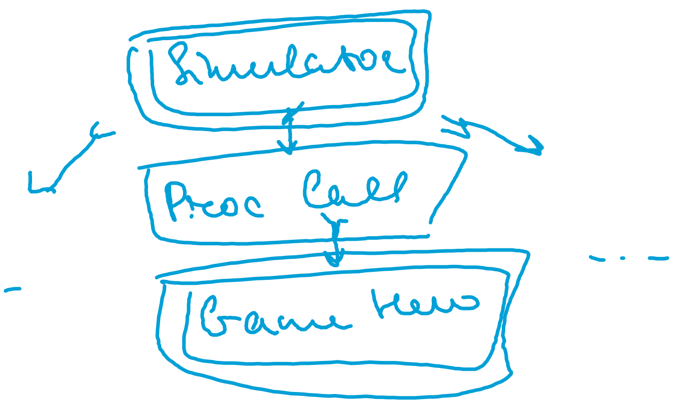
Ограничения: обычно общая память, трудность с многопоточ-ностью

Обеспечивает: целост-ть операций над данными, скрытыми от некорр. изменения. Возможность абстрагирова-ния.

Тип. применение: прикл. пр-мы.

Опасности: плохо подд-ся паралл-зм, нужная производит-ть легко усложняется

Tech: C++, Java, C#, Objective C, Kotlin, Swift и т.д.



2 Слоевая архитектура

Каждый layer имеет API, который

уровнем выше

доступен от...

- сервер (для выше)
- клиент (для ниже)

Коннекторы — это протоколы взаим-я между уровнями

+ доп уровень абстракции
+ лучше адапт. к изменениям
+ лучше обеспеч. повторное использование
+ лучше декомп. сложность
+ обесп. основу для проекта библиотек и каркасов
+ упрощает управление изменениями -ту

- не всегда применим
- низкая произв-ть
- иногда сложная структура
- лишний код для сокрытие
- опред-ние корректного уровня абстракции — трудно.

1 Слои

Layer 1

↓

⋮

Layer n

2 Виртуальная машина

Декомп. - упоряд набор слоев, каждый слой — вирт машина

Компоненты. слои, содерж. в себе компоненты

компоненты

Коннекторы: вызовы методов объектов, процедур, сообщения

Эл-ты данных: параметры вызовов

Топология: линейная, иерархия.

Объекты: логичная и простая структура

Применение: ОС, встраиваемое ПО, вирт. машины

Опасности: нужна производ-ть, негибкость

Tech. JVM, TCP/IP

*Модель OSI — сложная арх протоколы

3 Клиент-сервер

Декомп. сервер имеет набор сервисов. клиенты отр. запросы

Топология. двухуровневая (многоур.)

Ограничения: сервер пассивный, нет взаим-я клиентов с клиентами.

Комп: клиенты, сервер

Коннекторы: удаленные вызовы, сетевое взаим-вие

Эл-ты данных: сетевые данные, параметры вызовов

Применение: инф. системы, системы массового обслуживания, ... ПО контейнер

Обеспеч. : централизации данных и вычислений

Опасности : произв. зависит от аппаратной среды, необходимо нагрузочные тестирование.

4. MVC

* контроллер домен для тонкий

$$\begin{tikzcd}
 & \text{model} \arrow[dl, "\text{updates}"'] & \\
\text{view} \arrow[dr, "\text{sees}"'] & & \text{controller} \arrow[ul, "\text{manipulates}"'] \\
 & \text{user} \arrow[ur, "\text{uses}"'] &
\end{tikzcd}$$

варианты.

- model-view-presenter
- -adapter
- -viewmodel
- presentation-abstraction-control

3 Архитектура потоков данных

Отдельные пр-мы исп-ся в заданном порядке для послед. обработки данных

коннекторы – каналы между пр-ми

исп-ся в системах обработки данных, фин. системах, компиляторах, банковском ПО, в ML

– нет интерактивности

–легко просядет производительность
из-за одного медленного фильтра

Комп. программы
Коннекторы: Средства передачи данных
Эл-ты данных: наборы данных
Топология: линейная, послед.-ая
Ограничение: нет паралл. обработки
Обеспечивает целостность, простоту арх-ры
Тип. применение: фин. инф. системы, банк системы

1 Pipeline

Декомп. отдельные пр-мы вып-ся независимо

Компоненты: программы (= фильтры)
Коннекторы каналы ОС, потоки данных

Эл-ты данных: лин. потоки данных, часто (unix/linux/DOS) – текстовые

Топология: лин., T-shape

Применение оболочки ОС, ML-конвейеры

Ограничения: нет
Опасности: невозможно обеспечить [illegible]-вие компонентов,

данн
сложно передавать
сложные данные
Обеспечивает: незав-ть и заменяемость
фильтров

4 Shared State

Отдельные модули взаим-ют через
общую память

1 Blackboard

Компоненты. хранилище данных – board
обработчики данных
на доске
+ Состояние системы хранится
на доске

Применение. экспертные системы,
системы ML, системы
научных расчетов
системы AI, IDE,
компиляторы

Декомп: отдельные пр-ма и компоненты

Коннекторы: прямой доступ к памяти,
доступ по ссылке

Топология. звезда

Обеспечивает: гибкость, когда надо
менять стратегию

доступны более эфф. архит-ры,

Опасности: [illegible]
если стратегии решения
заранее известны.
Все завязано на единый
центр

<u>Экспертная система</u>

– применяются в прикладных областях.
(анализ многих аспектов)

* Wolfram Alpha, Alice. .... Siri...