

Проектирование и архитектура ПО

Лекция 1: Об архитектуре

Юрий Литвинов
y.litvinov@spbu.ru

13.02.2025

Организационное

- ▶ Лекционно-практический курс, будем практиковаться по большей части прямо на парах
- ▶ В конце теоретический зачёт
- ▶ Балльная система ECTS
 - ▶ 60% за теорзачёт, 40% за работу на парах
 - ▶ Можно доделывать дома за половину баллов
 - ▶ Прежде всего, проектирование, но будут и задачи на покодить
 - ▶ Будут групповые и индивидуальные задания
- ▶ Материалы и задания будут выкладываться в HwProj
 - ▶ <https://hwproj.ru/courses/40032>
- ▶ Коммуникация в Teams, в канале команды курса

Что будет в курсе

- ▶ Объектно-ориентированное проектирование
- ▶ Моделирование, язык UML и, немного, другие визуальные языки
- ▶ Шаблоны проектирования и антипаттерны
- ▶ Архитектурные стили
- ▶ Предметно-ориентированное проектирование
- ▶ Проектирование распределённых приложений
- ▶ Развёртывание и DevOps
- ▶ Примеры архитектур (если успеем)

Программа и программный продукт



© Ф. Брукс, «Мифический человек»



Размер типичного ПО

| | |
|------------------------------|------------|
| Простая игра для iOS | 10000 LOC |
| Unix v1.0 (1971) | 10000 LOC |
| Quake 3 engine | 310000 LOC |
| Windows 3.1 (1992) | 2.5M LOC |
| Linux kernel 2.6.0 (2003) | 5.2M LOC |
| MySQL | 12.5M LOC |
| Microsoft Office (2001) | 25M LOC |
| Microsoft Office (2013) | 45M LOC |
| Microsoft Visual Studio 2012 | 50M LOC |
| Windows Vista (2007) | 50M LOC |
| Mac OS X 10.4 | 86M LOC |

<http://www.informationisbeautiful.net/visualizations/million-lines-of-code/>

Архитектура

- ▶ Совокупность важнейших решений об организации программной системы
 - ▶ Эволюционирующий свод знаний
 - ▶ Разные точки зрения
 - ▶ Разный уровень детализации
- ▶ Для чего
 - ▶ База для реализации, «фундамент» системы
 - ▶ Инструмент для оценки трудоёмкости и отслеживания прогресса
 - ▶ Средство обеспечения переиспользования
 - ▶ Средство анализа системы ещё до того, как она реализована



© Интернеты

Профессия «Архитектор»

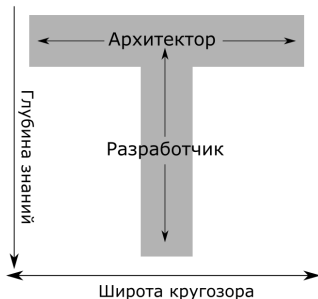
- ▶ Архитектор — специально выделенный человек (или группа людей), отвечающий за:
 - ▶ разработку и описание архитектуры системы
 - ▶ доведение её до всех заинтересованных лиц
 - ▶ контроль реализации архитектуры
 - ▶ поддержание её актуального состояния по ходу разработки и сопровождения

Трудовые функции архитектора

По профстандарту 06.003

- ▶ Выявление и согласование требований к программной системе с точки зрения архитектуры
- ▶ Выбор и моделирование архитектурного решения для реализации программной системы
- ▶ Разработка разделов по архитектуре проектных и эксплуатационных документов программной системы
- ▶ Контроль реализации и испытаний программной системы с точки зрения архитектуры
- ▶ Сопровождение эксплуатации программной системы с точки зрения архитектуры

Архитектор vs разработчик



- ▶ Широта знаний
- ▶ Коммуникационные навыки
- ▶ Часто архитектор играет роль разработчика и наоборот
 - ▶ Архитектор в «башне из слоновой кости» — это плохо

Пример: ПО для осциллографа

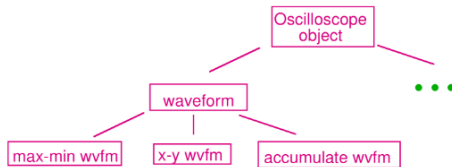
- ▶ Считывать параметры сигнала
- ▶ Оцифровывать и сохранять их
- ▶ Выполнять разные фильтрации и преобразования
- ▶ Отображать результаты на экране
 - ▶ С тач-скрином и встроенным хелпом
- ▶ Возможность настройки под конкретные задачи



© Hantek

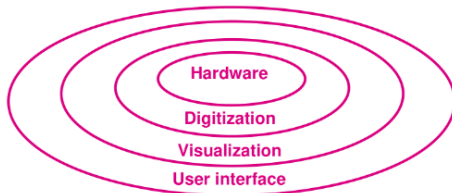
По статье Garlan D., Shaw M. An introduction to software architecture

Вариант 1: объектная модель



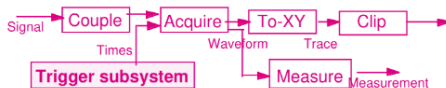
© Garlan D., Shaw M. An introduction to software architecture

Вариант 2: слоистая архитектура



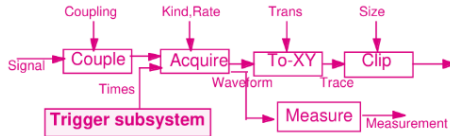
© Garlan D., Shaw M. An introduction to software architecture

Вариант 3: каналы и фильтры



© Garlan D., Shaw M. An introduction to software architecture

Вариант 4: модифицированные каналы и фильтры



© Garlan D., Shaw M. An introduction to software architecture

Выводы

- ▶ Можем делать утверждения о свойствах системы, базируясь на её структурных свойствах
 - ▶ Не написав ни строчки кода и даже не выбрав язык реализации
- ▶ Рассуждения очень субъективны
 - ▶ Многое зависит от интуиции и вкуса архитектора, однако ошибки очень дороги
- ▶ Можно выделить архитектурные стили — «архитектуры архитектур»
- ▶ Можно выделить архитектурные точки зрения и архитектурные виды
- ▶ Разный уровень детализации

Архитектурные виды

Стандарт IEEE 1016-2009

- ▶ Контекст — фиксирует окружение системы
 - ▶ Диаграммы активностей UML, IDEF0 (SADT)
- ▶ Композиция — описывает крупные части системы и их предназначение
 - ▶ Диаграммы компонентов UML, IDEF0
- ▶ Логическая структура — структура системы в терминах классов, интерфейсов и отношений между ними
 - ▶ Диаграммы классов UML, диаграммы объектов UML

Архитектурные виды (2)

- ▶ Зависимости — определяет связи по данным между элементами
 - ▶ Диаграммы компонентов UML, диаграммы пакетов UML
- ▶ Информационная структура — определяет персистентные данные в системе
 - ▶ Диаграммы классов UML, IDEF1x, ER, ORM
- ▶ Использование шаблонов — документирование использования локальных паттернов проектирования
 - ▶ Диаграммы классов UML, диаграммы пакетов UML, диаграммы коллабораций UML

Архитектурные виды (3)

- ▶ Интерфейсы — специфицирует информацию о внешних и внутренних интерфейсах
 - ▶ IDL, диаграммы компонентов UML, макеты пользовательского интерфейса, неформальные описания сценариев использования
- ▶ Структура системы — рекурсивное описание внутренней структуры компонентов системы
 - ▶ Диаграммы композитных структур UML, диаграммы классов UML, диаграммы пакетов UML
- ▶ Взаимодействия — описывает взаимодействие между сущностями
 - ▶ Диаграммы композитных структур UML, диаграммы взаимодействия UML, диаграммы последовательностей UML

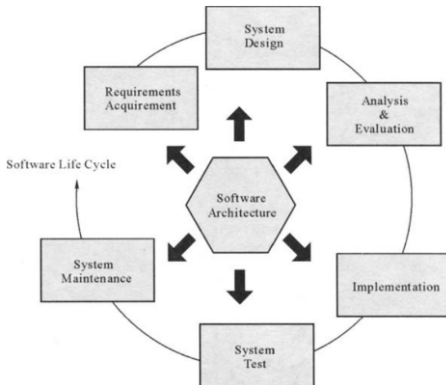
Архитектурные виды (4)

- ▶ Динамика состояний — описание состояний и правил переходов между состояниями
 - ▶ Диаграммы конечных автоматов UML, диаграммы Харела, сети Петри
- ▶ Алгоритмы — описывает в деталях поведение каждой сущности
 - ▶ Диаграммы активностей UML, псевдокод, настоящие языки программирования
- ▶ Ресурсы — описывает использование внешних ресурсов
 - ▶ Диаграммы развёртывания UML, диаграммы классов UML, OCL

Ещё про архитектурные виды

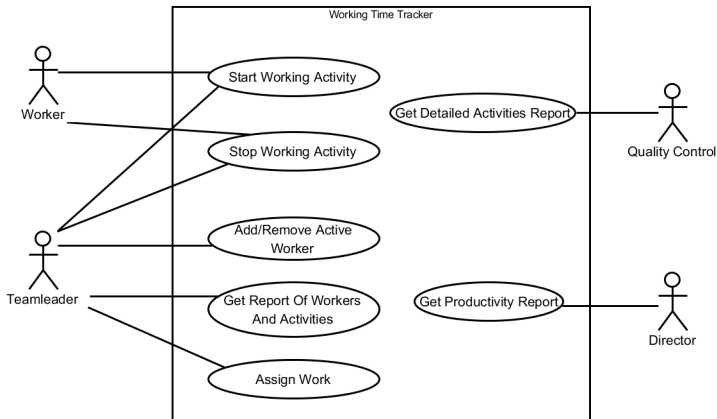
- ▶ Пример — http://robotics.ee.uwa.edu.au/courses/design/examples/example_design.pdf
- ▶ Ни один вид не обязателен
- ▶ Активно используются визуальные языки
 - ▶ В основном как дополнение к тексту
- ▶ Моделирование принципиально важно для архитектуры
 - ▶ Нельзя абстрагироваться от сложности, но можно декомпозировать сложность

Роль архитектуры в жизненном цикле



© Z. Quin, "Software Architecture"

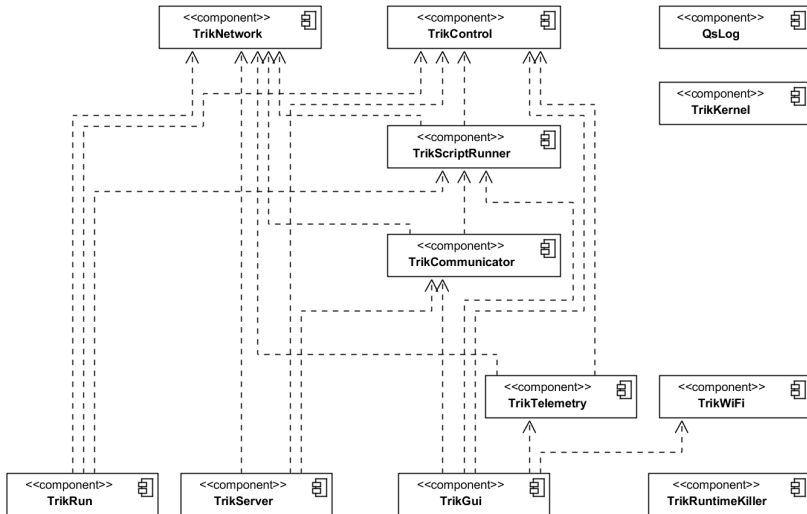
Архитектура и требования



Требования

- ▶ Функциональные — то, что система должна делать
- ▶ Нефункциональные — то, как система должна это делать
 - ▶ Эффективность
 - ▶ Масштабируемость
 - ▶ Удобство использования
 - ▶ Надёжность
 - ▶ Безопасность
 - ▶ Сопровождаемость и расширяемость
 - ▶ ...
- ▶ Ограничения
 - ▶ Технические
 - ▶ Бизнес-ограничения

Архитектура и проектирование



Архитектура и проектирование — задачи

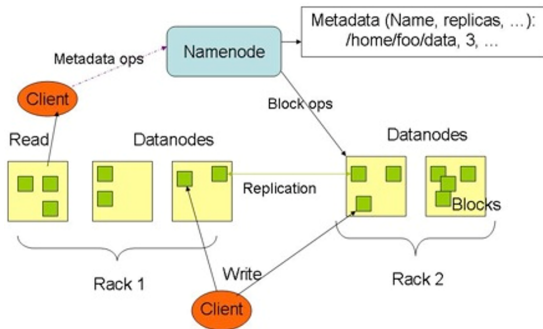
- ▶ Декомпозиция задачи
- ▶ Определение границ компонентов
- ▶ Определение интерфейсов между компонентами
- ▶ Общие для всей системы вопросы
 - ▶ Стратегия обработки ошибок
 - ▶ Стратегия логирования
 - ▶ Стратегия обновлений
 - ▶ Стратегия разделения доступа
 - ▶ Вопросы локализации
 - ▶ ...
- ▶ Анализ и верификация архитектуры

Архитектура и разработка

- ▶ prescriptive architecture — архитектура, как её определил архитектор
- ▶ descriptive architecture — архитектура, как она есть в системе
 - ▶ Архитектура у ПО есть всегда, как вес у камня
- ▶ architectural drift — «сползание» фактической архитектуры
 - ▶ появление в ней важных решений, которых нет в описательной архитектуре
- ▶ architectural erosion — «размывание» архитектуры
 - ▶ отклонения от описательной архитектуры, нарушения ограничений
- ▶ Антипаттерн «Big ball of mud» — результат эрозии

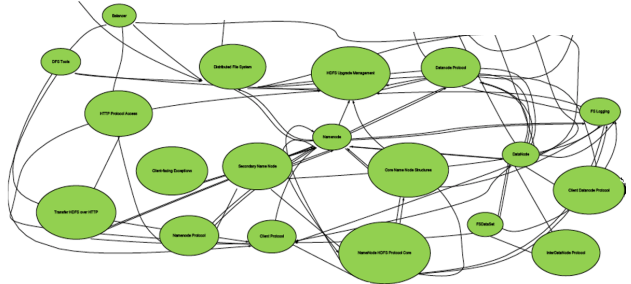
Пример: Hadoop

As-designed



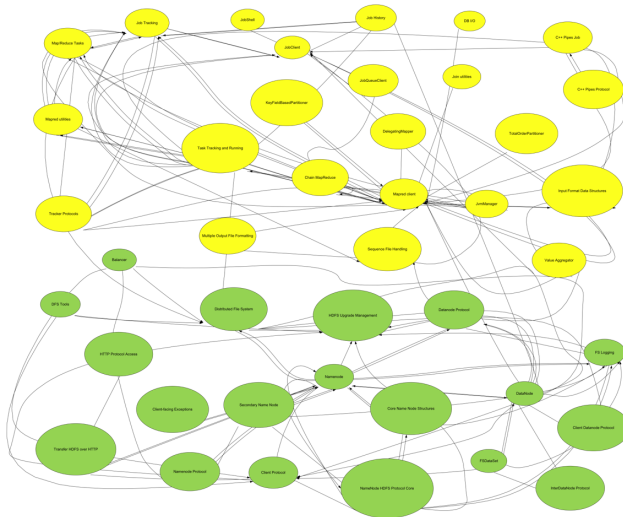
Special thanks to prof. Nenad Medvidovic (USC) for kind permission for using his slides

© Nenad Medvidovic



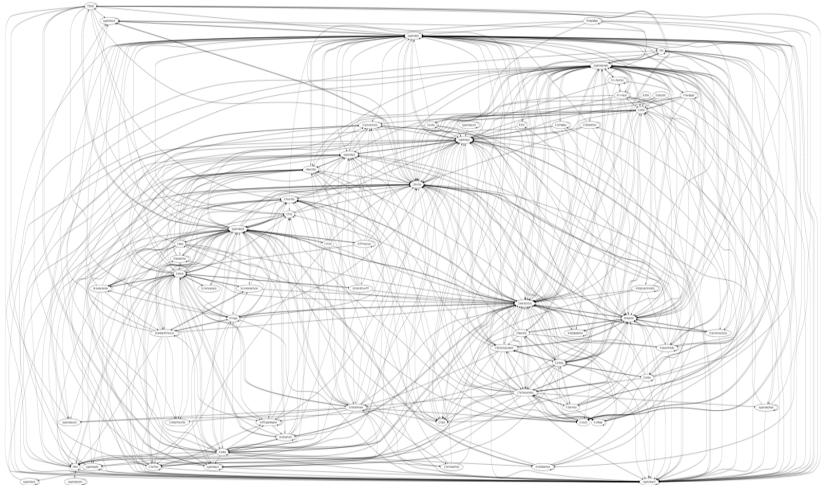
Hadoop as-built

HDFS + MapReduce



Hadoop as-built

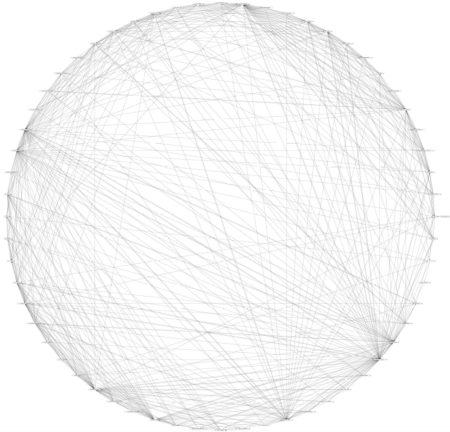
Полная архитектура



© Nenad Medvidovic

Hadoop as-built

Граф зависимостей



© Nenad Medvidovic