Распределенные информационные системы

Проектирование РИС

Вопросы

- Общие понятия:
  - Определения
  - Метод проектирования
  - Технология
- Типовые архитектуры
  - HLD vs. LLD
  - Архитектура, паттерны
  - Классификация и описание типовых архитектур (Шоу и Гарлан, другие…)
- Стандарты и документы
  - Применимые стандарты
    - Отечественные и зарубежные
    - Эскизный проект vs. ADD
  - ISO 42010
  - Zachman framework
- Обобщенный алгоритм проектирования

Методы

Технологии

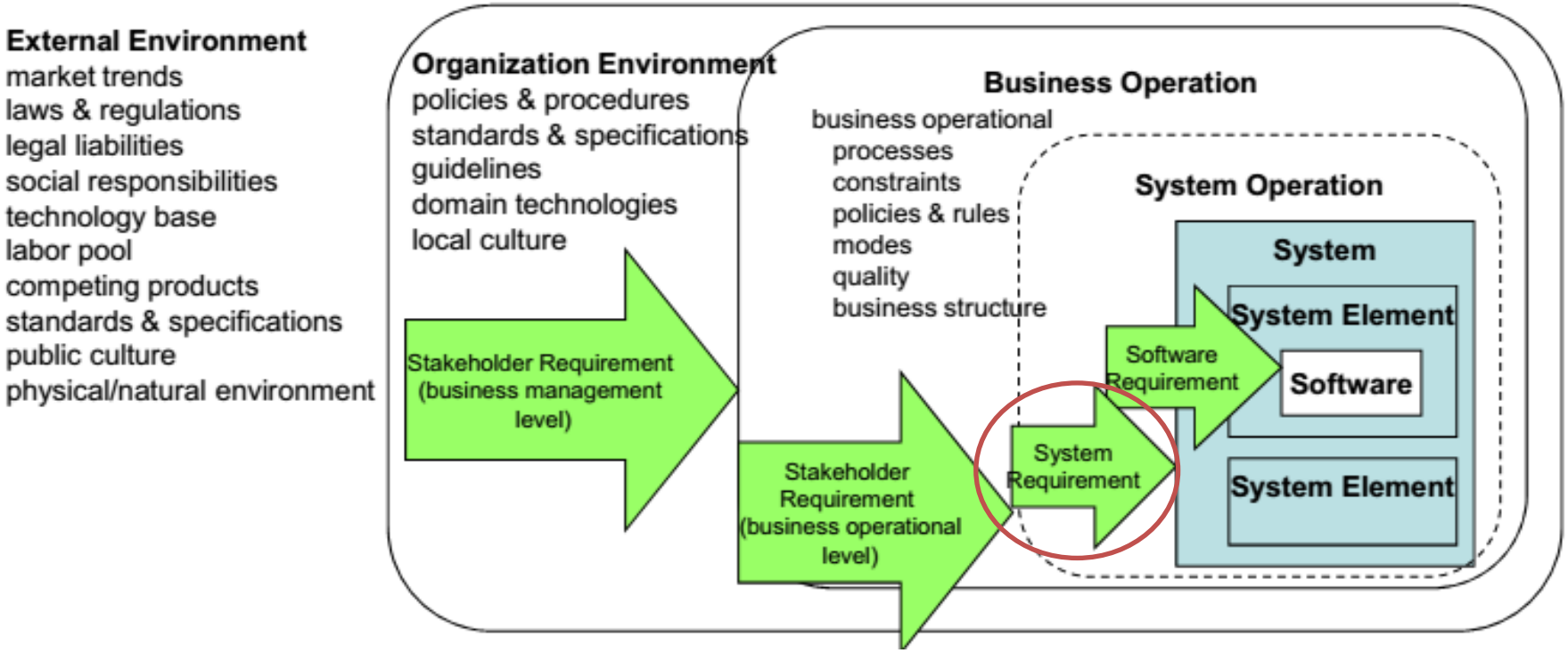
Стандарты

ПРОЕКТИРОВАНИЕ: ОБЩИЕ ПОНЯТИЯ

Определения

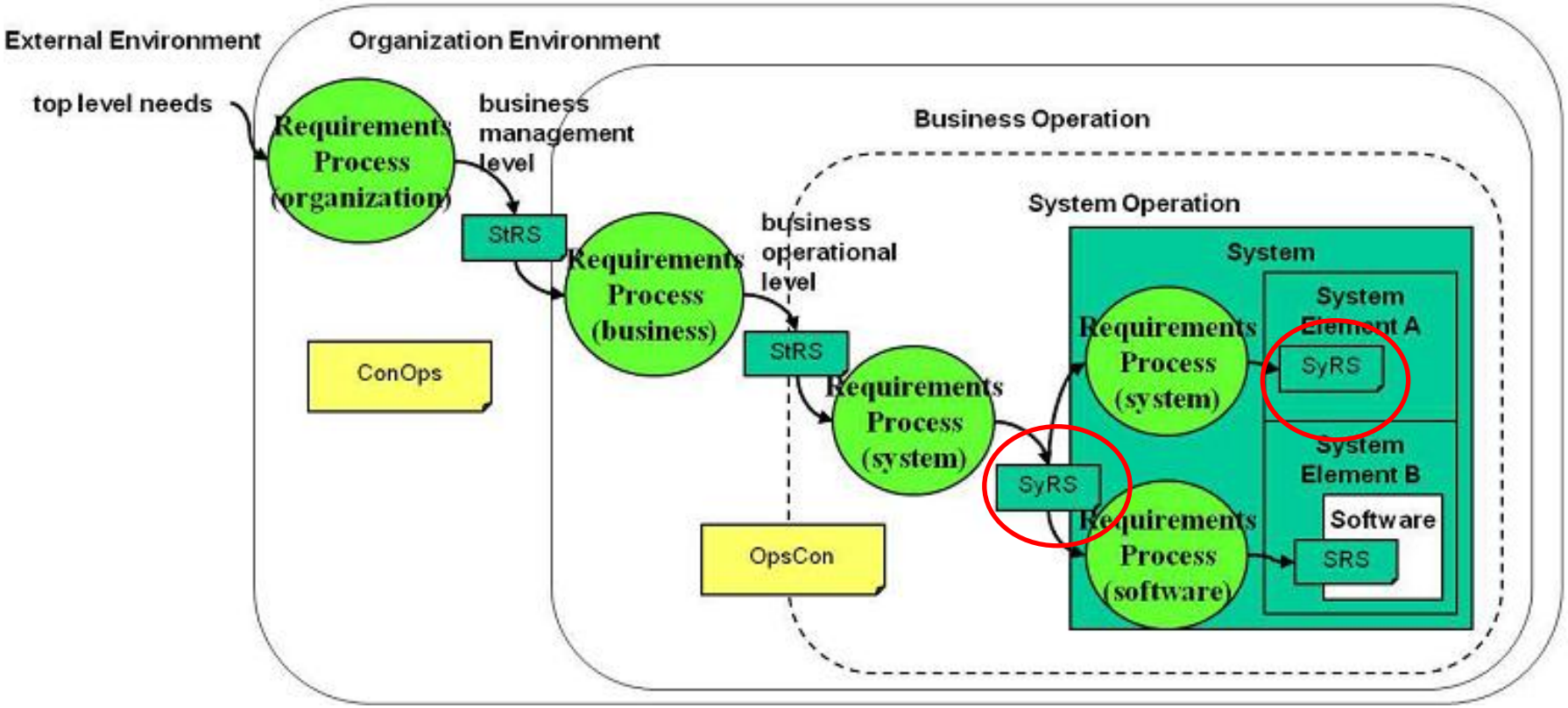
- Проектирование ПО — процесс создания проекта ПО
  - Ведется на основе требований к ПО
  - Подвергаются анализу:
    - Требования
    - Предметная область – в целом и область предполагаемого применения
- Цель проектирования:
  - Определение внутренних свойств системы
  - Детализация её внешних (видимых) свойств
- Проектированию подлежат:
  - Архитектура ПО
  - Устройство компонентов ПО
  - Пользовательские интерфейсы

Место ТЗ в системе – 1/3



- IEEE 29148-2011 – раздел 5.4 «Requirement information items»

Место ТЗ в системе – 2/3



Место ТЗ в системе – 3/3

- Stakeholder – заинтересованная сторона, в нашем случае – заказчик
- ConOps (Concept of Operation)
  - Текстовое и графическое представление стратегических планов заказчика
- OpsCon (System Operational Concept)
  - Текстовое и графическое представление использования системы (производимых ею действий и их предполагаемых результатов)

Метод проектирования – 1/3

- Метод проектирования ПО – организованная совокупность процессов создания ряда моделей, описывающих различные аспекты разрабатываемой системы. В метод входят:
- Концепция – теоретическая основа
  - Например: функциональный подход, модульный подход, структурный подход, объектно-ориентированный подход

Метод проектирования – 2/3

- Нотация – способ представления моделей (обычно графические)
  - Блок-схема
  - ER-диаграмма
  - UML-диаграммы (ООП): диаграммы вариантов использования, диаграммы классов
  - Для структурного проектирования: диаграммы потоков данных, и диаграммы «сущность – связь»
  - IDEF-диаграммы для описания бизнес-процессов (IDEF0)

Метод проектирования – 3/3

- Процедуры – определяют практическое применение метода
  - Последовательность и правила построения моделей
  - Критерии, используемые для оценки результатов

Технология проектирования

- Методы проектирования реализуются через конкретные:
  - Технологии и поддерживающие их методики
  - Стандарты
  - Инструментальные средства
- Технология проектирования – совокупность технологических операций проектирования в их последовательности и взаимосвязи, приводящая к разработке проекта ПО

Требования к технологии – 1/2

- Общие формальные:
  - Соответствие стандарту (стандартам)
  - Гарантированное достижение целей разработки при соблюдении ограничений:
    - Бюджет
    - Заданное качество
    - В установленное время
- Организационные для заказчика:
  - Минимальное время до получения рабочего прототипа или отдельной рабочей подсистемы

Требования к технологии – 2/2

- Организационные для разработчика:
  - Разбиение проекта на части, доступные для [независимой] реализации небольшими группами разработчиков (3-7 человек) с последующей интеграцией
  - Независимость проекта от реализации (ЯП, СУБД)
  - Наличие поддерживающих CASE средств

Подходы к проектированию – 1/2

- Модульный
  - Разбиение ПС на модули – составные части, которые могут быть независимо написаны и отлажены
- Структурный
  - Представление ПС в виде (иерархической) многоуровневой системы компонентов
- Функциональный
  - Разделение на части, реализующие отдельные функции
- ОО = 1 + 2 + 3

Подходы к проектированию – 2/2

- Сверху вниз:
  - Начинают с верхнего уровня (обычно включающего внешние связи всей проектируемой системы), разбивают систему на части
  - Каждую часть разбивают далее
- Снизу вверх:
  - Сначала описывают наиболее мелкие элементы системы
  - Из них выстраивают более крупные

HLD vs. LLD

Архитектуры и паттерны [программирования]

ТИПОВЫЕ АРХИТЕКТУРЫ [ПО]

Процесс проектирования [SWEBOK KA-2]

- Проектирование [программных систем] можно рассматривать как деятельность, порождающую:
  - Архитектурный или высокоуровневый дизайн (software architectural design, top-level design, high-level design – HLD)
    - описание высокоуровневой структуры и организации компонентов системы
  - Детализированную архитектуру (software detailed design, low-level design – LLD)
    - описание каждого компонента в объеме, достаточном (необходимом?) для его конструирования (воплощения)

Архитектуры и паттерны

- Паттерн (образец [проектирования]) – эффективный (? распространенный) способ решения характерных задач проектирования, в частности проектирования компьютерных программ
- Термины «System architectures» и «Design patterns» предполагают, что описываемые ими решения находятся на разных «уровнях» представления систем. В нашем случае будем полагать, что:
  - Архитектура относится к уровню HLD
  - Паттерн относится к уровню LLD
- Однако:
  - Подходы, описываемые архитектурами могут проецироваться на более низкий уровень
  - Паттерны достаточно просто обобщаются на более высокий уровень

«Состав» архитектуры

- Будем понимать под архитектурой комбинацию:
  - Специфической структуры, определяемой набором [типов] элементов и связей между ними
  - Обобщенный алгоритм функционирования (взаимодействия между элементами)
  - Ограничений, уменьшающих разнообразие реализаций

Типы архитектур: введение

- Существует множество «стандартных» архитектур. Они могут быть классифицированы разными способами:
  - Классификация Шоу и Гарлана (An Introduction to Software Architecture, David Garlan, Mary Shaw)
  - По назначению (Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm)
  - По Марку Ричардсу
  - etc.

ТИПОВЫЕ АРХИТЕКТУРЫ ПО ШОУ И ГАРЛАНУ

Шоу и Гарлан – История

- Классификация Шоу и Гарлана относится к уровню архитектур. Считается классической, так как была сформулирована одной из первых (1994)
- Претендует на всеобщность, так как включает очень общие архитектуры
- Описанная ниже классификация доработана Э.Дж.Брауде («Технология разработки программного обеспечения», 2004)

Шоу и Гарлан: Архитектуры – 1/2

- Архитектуры потоков данных:
  - Каналы и фильтры
  - Последовательные пакеты
- Независимые компоненты:
  - Параллельные взаимодействующие процессы
  - Клиент-серверные системы
  - Системы, управляемые событиями
- Виртуальные машины:
  - Интерпретаторы
  - Системы, основанные на правилах

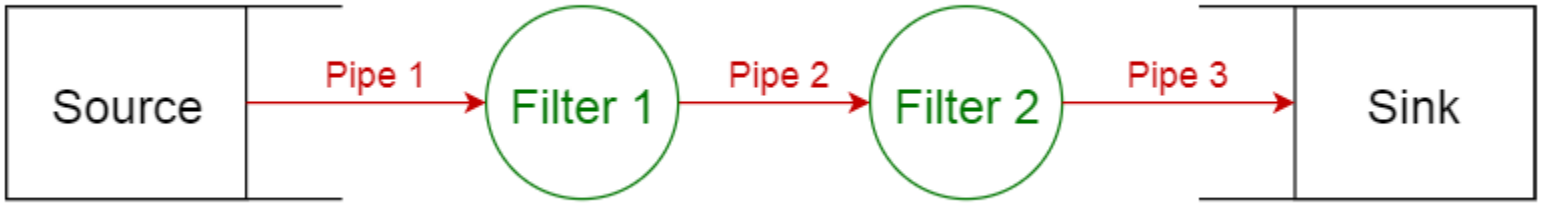
Шоу и Гарлан: Архитектуры – 2/2

- Репозиторные архитектуры:
  - Базы данных
  - Гипертекстовые системы
  - Доски объявлений
- Уровневые архитектуры

Каналы и фильтры – 1/3

- Каналы и фильтры (конвейеры и фильтры) (ориг. Pipes and filters):
  - Структура и алгоритм:
    - Каждый компонент имеет вход и выход
    - Данные передаются потоками (и входные, и выходные)
    - Выходной поток начинает поступать до окончания загрузки входного
    - Процессы обработки – «фильтры», связи данных – «каналы»
  - Ограничения:
    - Фильтры независимы (хотя синхронизация работы допускается)
    - Фильтры могут определять требования ко входным данным, но не к тому, какие [другие] фильтры используют их результаты
    - Каждый отдельный элемент потока данных не имеет прагматически значимого смысла
  - Варианты архитектур:
    - Конвейер
    - Каналы с ограничением пропускной способности
    - Типизованные каналы

Каналы и фильтры – 2/3



Данное представление показывает конвейерный вариант архитектуры (с единственным путем потока данных)

Каналы и фильтры – 3/3

- Примеры:
  - Компилятор, который последовательно выполняет:
    - Предобработку (preprocessing)
    - Разбор на лексемы (парсинг)
    - Синтаксический анализ
    - Семантический анализ
    - Генерацию кода
  - Потоки заданий в биоинформатике (workflows in bioinformatics)
  - Ретрансляция сигналов спутникового телевидения

Последовательные пакеты

- Последовательные пакеты
  - Отличается от «Каналов и фильтров» тем, что:
    - Данные организованы в пакеты
    - Пакет обрабатывается как неделимая информационная сущность
    - Поступление пакета на вход системы и прохождение ее по фильтрам рассматривается как одна операция (транзакция)

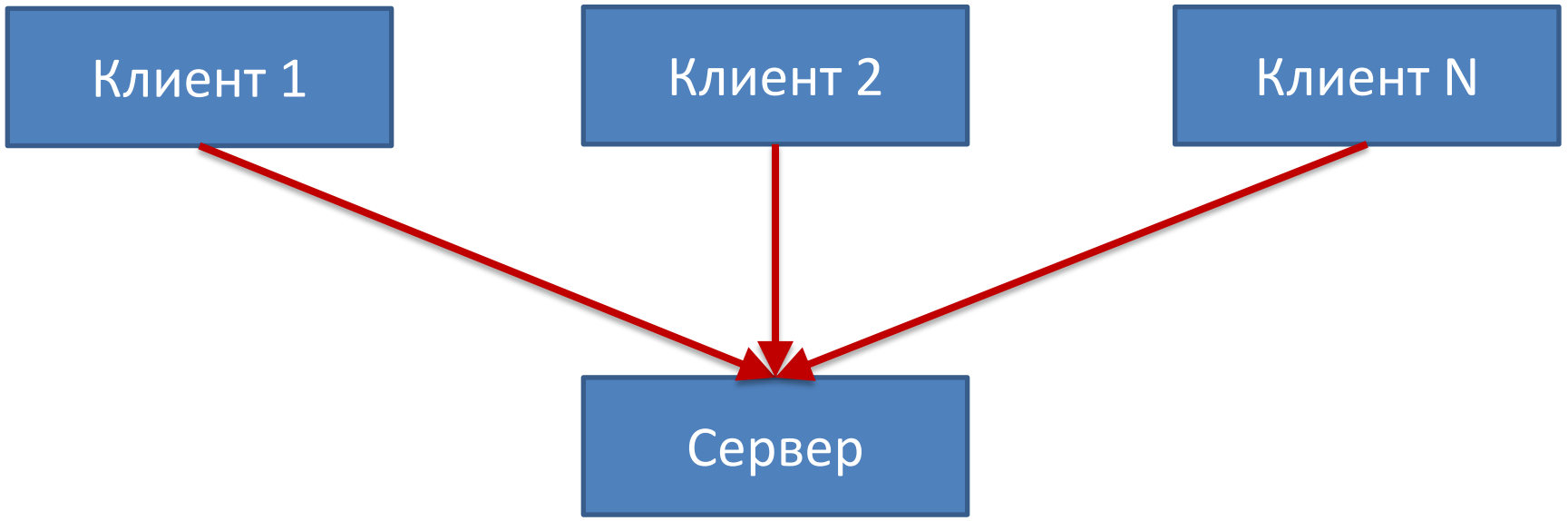
Параллельные взаимодействующие процессы

- Параллельные взаимодействующие процессы
  - Структура и алгоритм:
    - Компоненты системы представляются в виде процессов (или потоков), выполняющихся параллельно
    - Они могут взаимодействовать: передавать данные, создавать, удалять, управлять выполнением друг друга
  - Ограничения:
    - Нет! Поэтому «все сложно» (с)

Клиент-сервер – 1/3

- Клиент-сервер
  - Структура и алгоритм:
    - Два типа процессов: сервер и клиент
    - Активным действующим началом выступают клиенты
    - Каналы связи только клиент-сервер
  - Ограничения:
    - Нет горизонтального взаимодействия клиентов
  - Варианты архитектур:
    - Типов клиентов может быть несколько
    - Однотипных серверов может быть несколько. Запрос от клиента проходит через менеджер запросов (балансировщик нагрузки)
    - Несколько разнотипных серверов, предоставляющих разные сервисы (вариант – ESB Enterprise Service Bus – сервисная шина предприятия)

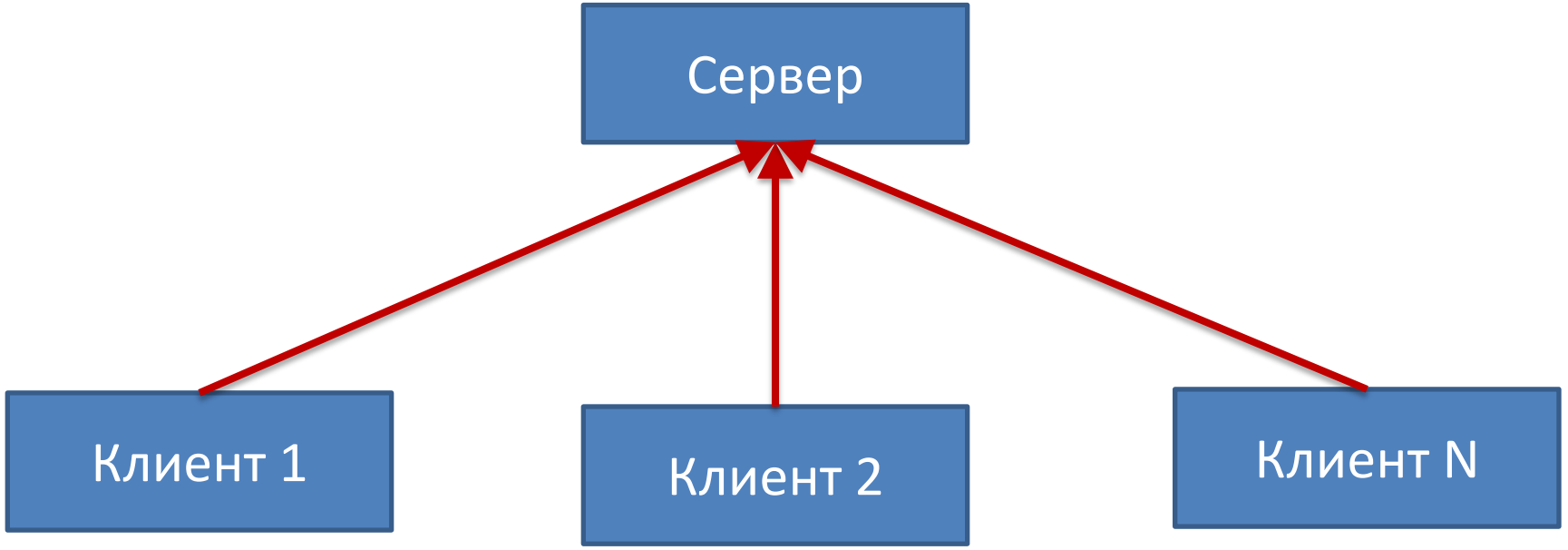
Клиент-сервер – 2/3



Данное представление:

- Подчеркивает пассивность сервера
- Согласуется с уровневым представлением в том, что предоставляемый интерфейс находится «сверху»

Клиент-сервер – 3/3



Данное представление:

- Подчеркивает факт, что сервер является постоянно действующей частью системы
- ~~Демонстрирует унижение клиентов, выпрашивающих подачки~~

Системы, управляемые событиями

- Системы, управляемые событиями
  - Структура и алгоритм:
    - Набор компонентов, находящихся в режиме ожидания до момента наступления события
    - Поведение компонентов и системы в целом описывается набором состояний и переходами из одного состояния в другое
  - Ограничения:
    - Значительное время компоненты простаивают
  - Варианты архитектур:
    - Есть только состояние системы в целом, компоненты используют его для своей работы
    - Смена состояния компонента не вызывает действие непосредственно, а:
      - Меняет набор доступных сервисов компонента
      - Изменяет алгоритм обработки данных

Интерпретаторы

- Интерпретаторы
  - Структура и алгоритм:
    - Система рассматривается как специализированный компьютер, исполняющий код на [высокоуровневом] ЯП (сценарий)
    - Обычно, есть несколько разных сценариев
  - Ограничения:
    - Описывается ЯП и доступные в нем команды
    - ЯП обычно человеко-читаем или существует компонент, позволяющий оператору редактировать его в читаемом виде
  - Варианты архитектур:
    - Исполняется одновременно один или несколько сценариев
    - Кто интерпретирует программу:
      - Отдельный компонент, управляющий работой других
      - Каждый из компонентов независимо, выбирая касающиеся его задачи

Системы, основанные на правилах

- Системы, основанные на правилах
  - Структура и алгоритм:
    - Система рассматривается как специализированный компьютер, реализующий исполнение правил
    - Входные данные рассматриваются как «левая часть» правил
  - Ограничения:
    - Задается набор возможных реакций («правых частей» правил)
  - Варианты архитектур:
    - Человеко-читаемость правил:
      - Правила могут быть заданы в человеко-читаемом виде или существует компонент, позволяющий оператору редактировать его в читаемом виде (пример: Экспертные системы)
      - Правила могут быть принципиально неинтерпретируемы человеком (пример: Нейронные сети)
    - Источники правил (дополняет предыдущие варианты):
      - Эксперты-люди
      - Обучающие выборки

Базы данных

- Базы данных
  - Структура и алгоритм:
    - Единое хранилище формализованных данных
    - Способы манипуляции данными стандартизованы
    - Единственный или основной интерфейс компонентов – с СУБД
  - Ограничения:
    - Поведение компонентов зависит:
      - От поступающих извне данных
      - От хранящихся в БД данных
  - Варианты архитектур:
    - Современные «not only SQL» СУБД

Доски объявлений

- Доски объявлений (ориг. blackboard):
  - Структура и алгоритм:
    - Независимые приложения (источники знаний) формируют пакеты данных, чей формат завит от использующих их приложений
    - Данные хранятся иерархически, скомпонованные по их источникам
    - Система «управляется» состоянием доски объявлений – приложения реагируют на обновление данных доски
  - Ограничения:
    - Правила поведения компонентов и использования доски являются системно-специфическими
  - Варианты архитектур:
    - Управление может фактически реализовываться: ПО доски, источниками данных, отдельным компонентом и/или их комбинацией

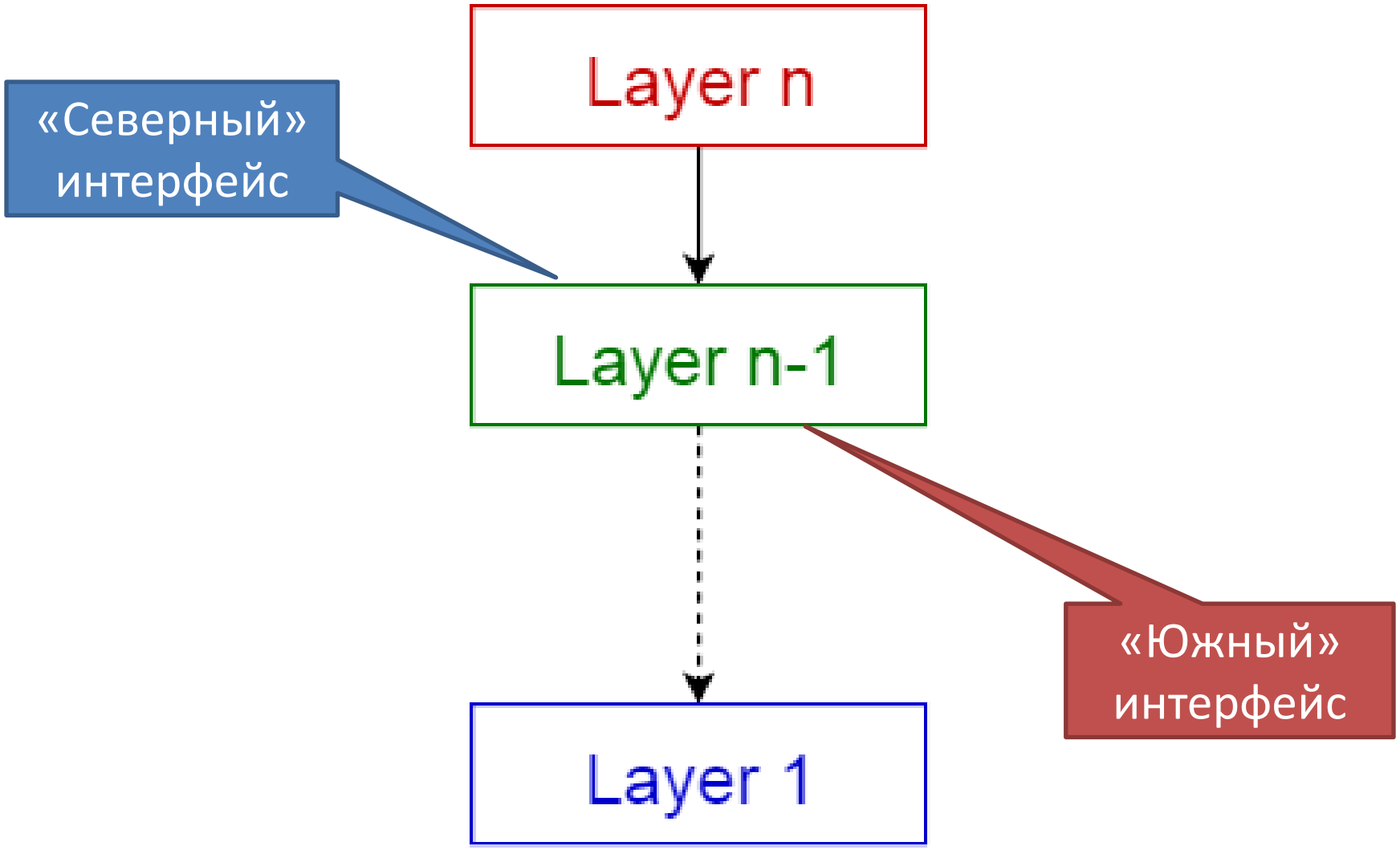
Гипертекстовые системы

- Гипертекстовые системы:
  - Структура и алгоритм:
    - Данные хранятся в виде гипертекста
    - Количество дополнительных форматов данных внутри гипертекста ограничено (?). Основной текстовый формат используется многими приложениями
    - Кроме прочего, в гипертексте присутствуют ссылки-активаторы приложений
  - Ограничения:
    - Значительная часть гипертекста человеко-читаемая
  - Варианты архитектур:
    - Предназначение гипертекста:
      - Хранение данных/знаний – сам гипертекст активно изменяется/пополняется
      - Организация взаимодействия программ – гипертекст относительно стабилен и служит пользовательским интерфейсом системы

Уровневые архитектуры – 1/3

- Уровневые архитектуры:
  - Структура и алгоритм:
    - Уровень – логически связанная коллекция процессов/потоков/классов (подсистема)
    - Уровень предоставляет набор сервисов более высокому уровню через «северный» интерфейс
    - Уровень использует сервисы нижележащего уровня. Требующиеся от нижнего уровня сервисы формируют «южный» интерфейс
  - Ограничения:
    - Через-уровневые связи не допускаются
  - Варианты архитектур:
    - Количество вышестоящих уровней может быть больше одного
    - Количество нижележащих уровней может быть больше одного
    - [Немного] сквозь-уровневых связей имеется

Уровневые архитектуры – 2/3



Уровневые архитектуры – 3/3

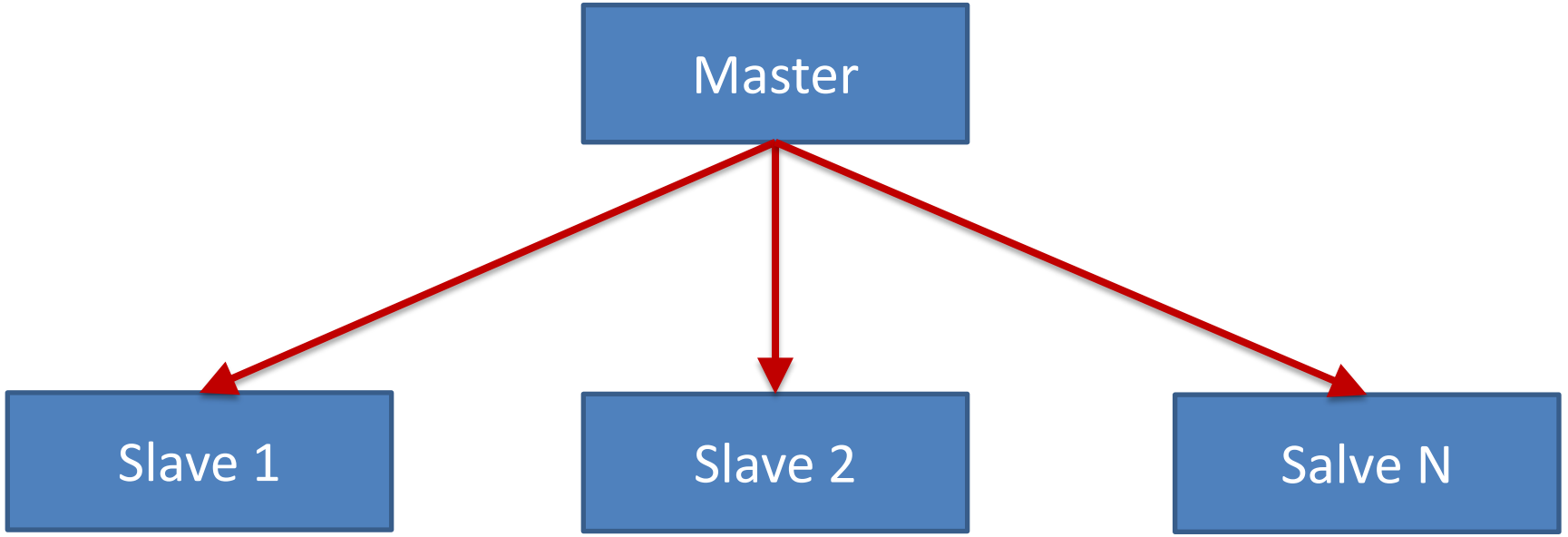
- Примеры разбиений:
  - Четырехуровневые
    - Уровень представления (данных пользователям через пользовательский интерфейс)
    - Уровень сервисов (приложений)
    - Уровень бизнес-логики (доменный)
    - Уровень доступа к данным
  - «Трехзвенка»:
    - Клиенты
    - Сервер приложений
    - СУБД

ЕЩЕ ТИПОВЫЕ АРХИТЕКТУРЫ

Master-slave – 1/3

- Master-slave
  - Структура и алгоритм:
    - Два типа процессов: master и slave
    - Активным компонентом является master
    - Компоненты slave ожидают команд от мастера и выполняют их
  - Ограничения:
    - Нет горизонтального взаимодействия компонентов slave
  - Варианты архитектур:
    - Финальный результат вычисляется master
    - Одной из задач slave (одного из них) является формирование финального результата
    - Разное расположение внешних интерфейсов системы

Master-slave – 2/3



NB! Такая архитектура является (с некоторой точки зрения) вырожденным вариантом архитектуры сервисная шина предприятия – с единственным клиентом

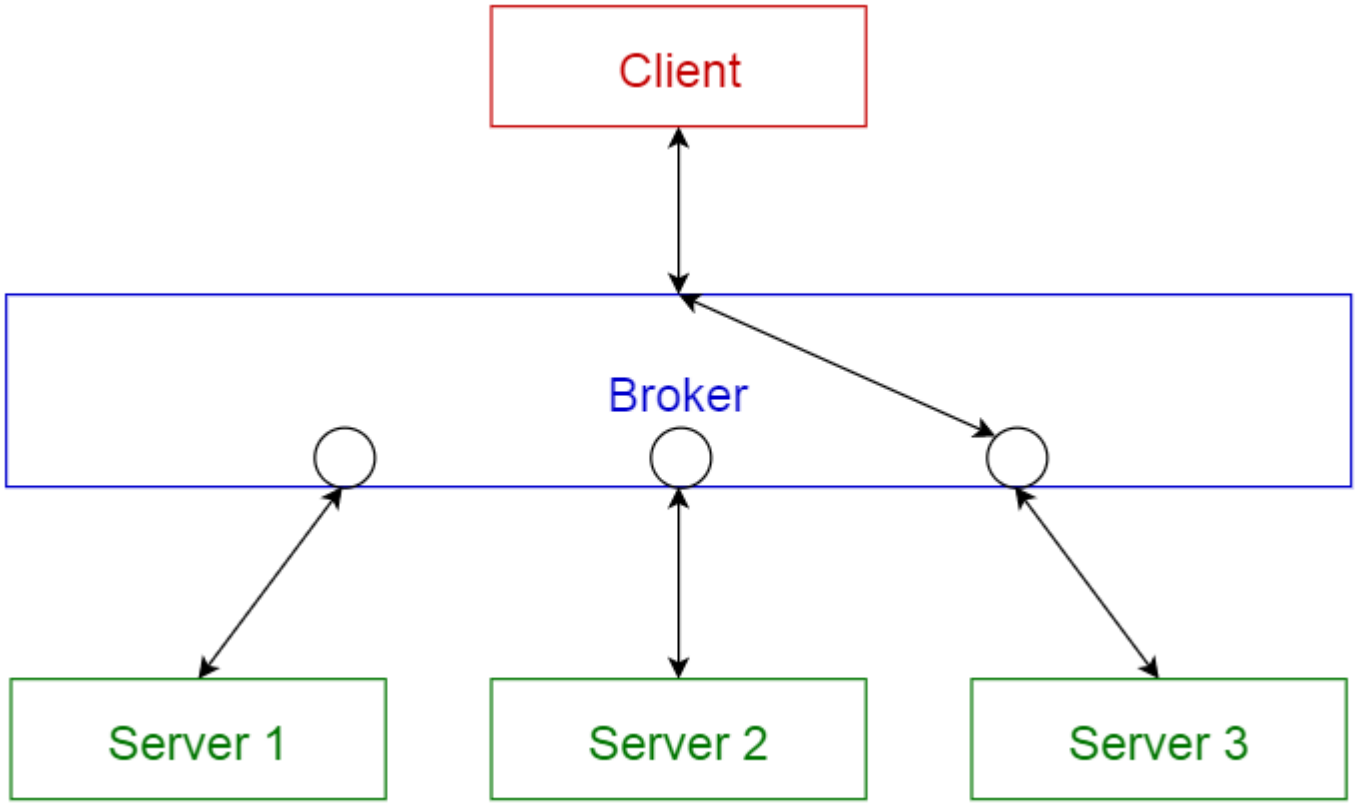
Master-slave – 3/3

- Примеры:
  - Один из вариантов распределенной БД, где БД-мастер является референсным источником, а подчиненные БД синхронизируются с ним
  - Периферийные устройства, подключенные к ЭВМ

Брокер – 1/3

- Брокер
  - Структура и алгоритм:
    - Три типа процессов: сервера, клиенты и брокер
    - Сервера регистрируют (публикуют) свои сервисы в брокере
    - Клиенты запрашивают использование сервисов у брокера, брокер перенаправляет запросы клиентов на походящие [зарегистрированные] сервера
  - Ограничения:
    - Прямой связи клиент-сервер нет
  - Варианты архитектур:
    - Брокер по запросу сервиса выдает клиенту адрес подходящего сервера, и дальше они взаимодействуют непосредственно
    - Сервера могут быть однотипными, тогда брокер является распределителем нагрузки

Брокер – 2/3



На схеме:

- Не показан процесс регистрации сервисов
- Показан единственный клиент

Брокер – 3/3

- Примеры:
  - Технология CORBA
  - Любое связующее ПО построенное с использованием парадигмы «подписка-публикация»:
    - *MQ*: ZeroMQ, RabbitMQ, AMQP
    - Apache Kafka, Jboss Messaging
    - HLA
    - DDS

ТИПОВЫЕ АРХИТЕКТУРЫ ПО РИЧАРДСУ

Марк Ричардс – 1/2

- Классификация относится к уровню архитектур приложений. Не является классической
- Подход легко расширяется до архитектур высокого уровня
- Рассматривается небольшое количество архитектур, использующихся в современной ИТ-индустрии для построения корпоративных (в-основном офисных) систем

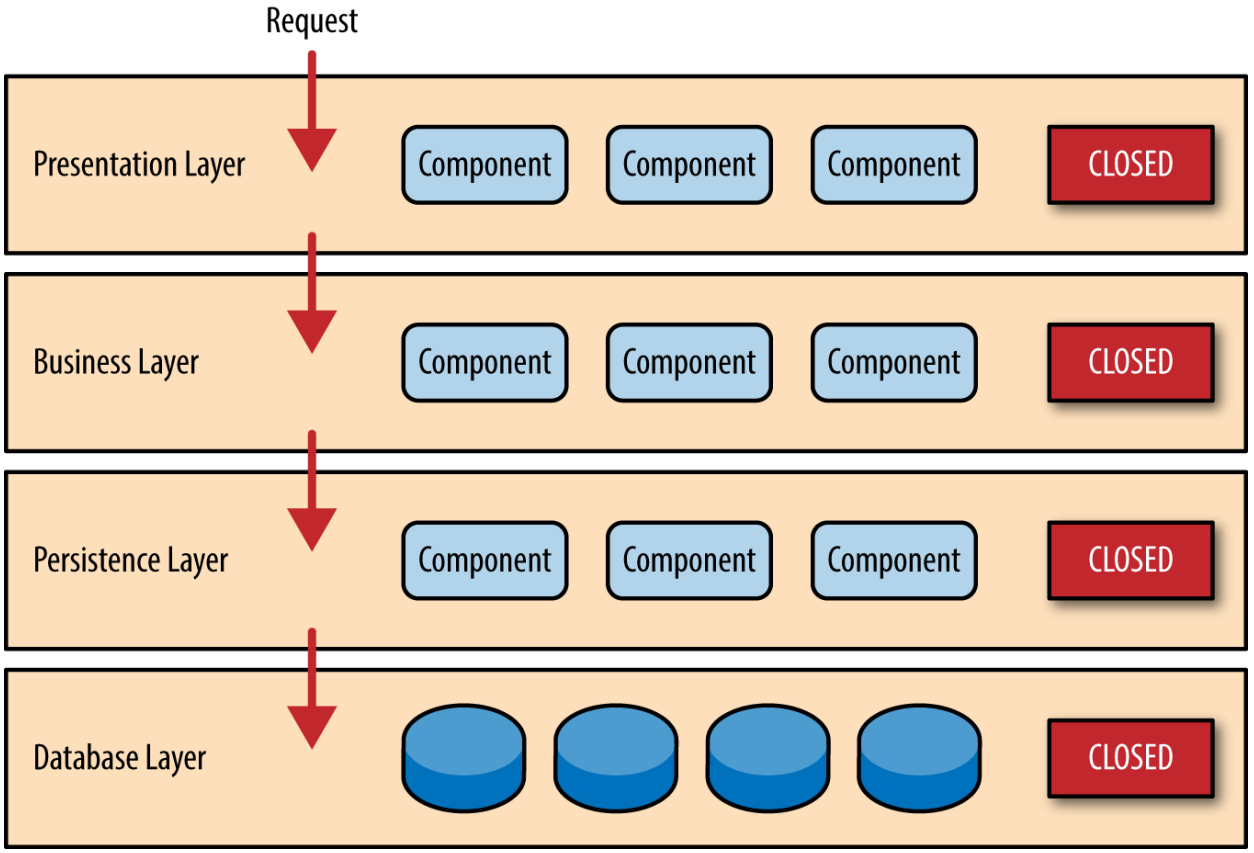
Марк Ричардс – 2/2

- Выделяются следующие архитектуры:
  - Уровневая
  - Управляемая событиями
  - Микроядерная
  - Микросервисная
  - Облачная (space-based)

Ричардс: Уровневая – 1/3

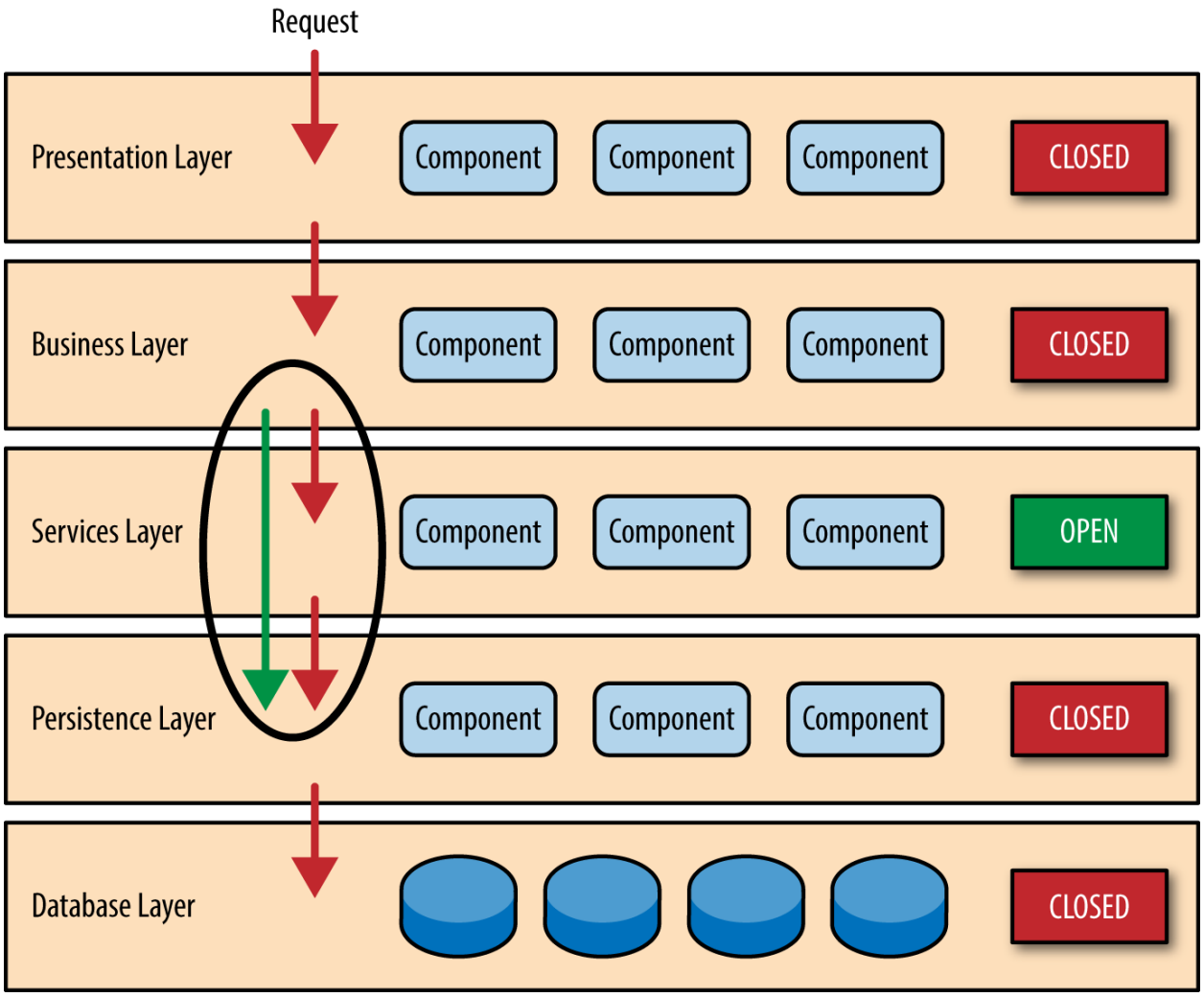
- Структура и алгоритм
  - Компоненты организуются в горизонтальные уровни, выполняющие специфические задачи
  - Особенности реализации сервисов скрываются от других уровней
  - «separation of concerns» вышележащий уровень использует только сервисы нижележащего
- Варианты архитектур:
  - «Закрытые» уровни – используются только сервисы соседних уровней, «открытые» уровни – есть возможность «сквозного» обращения

Ричардс: Уровневая – 2/3



Закрытые уровни

Ричардс: Уровневая – 3/3

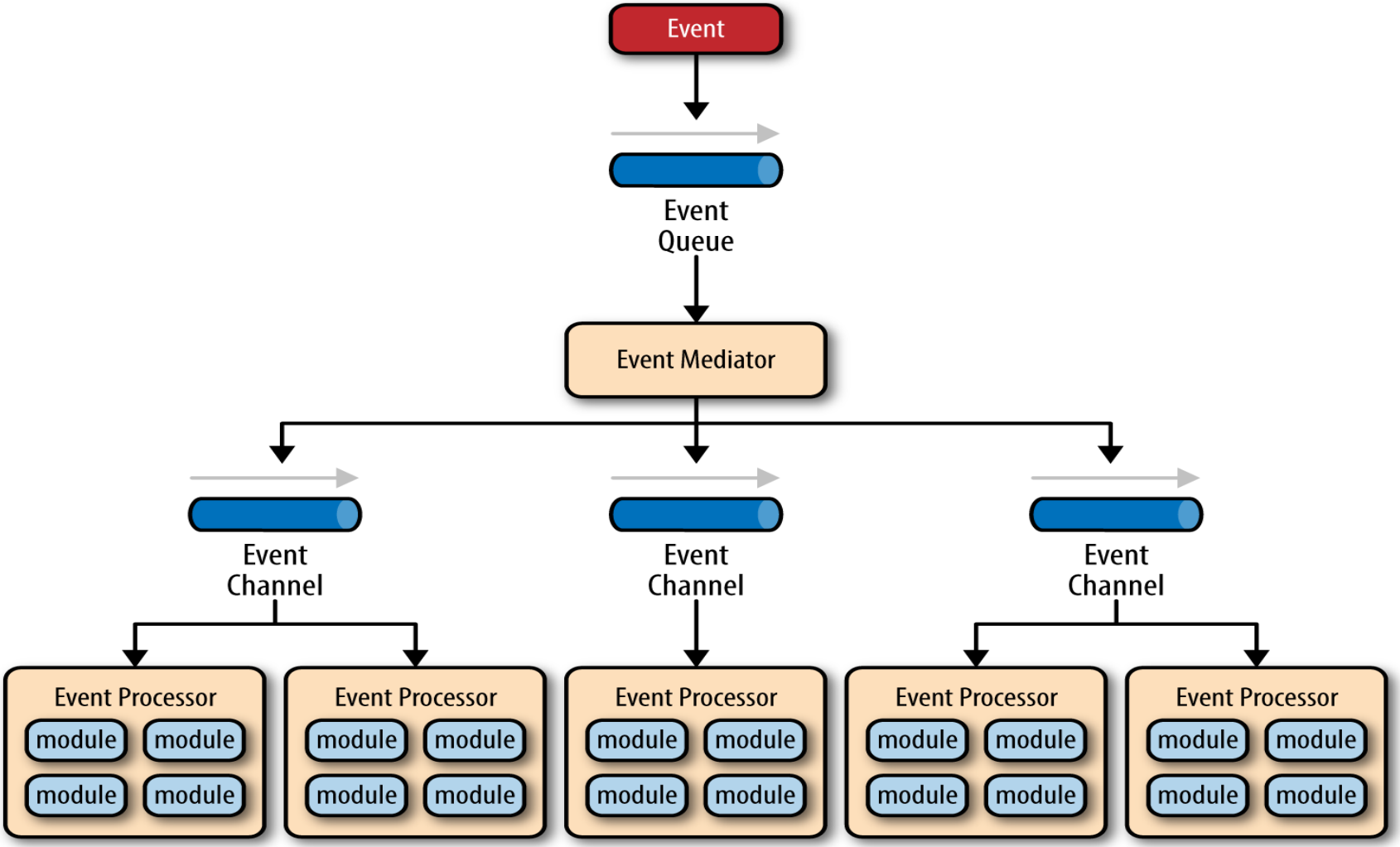


Открытые уровни

Ричардс: Управляемая событиями – 1/3

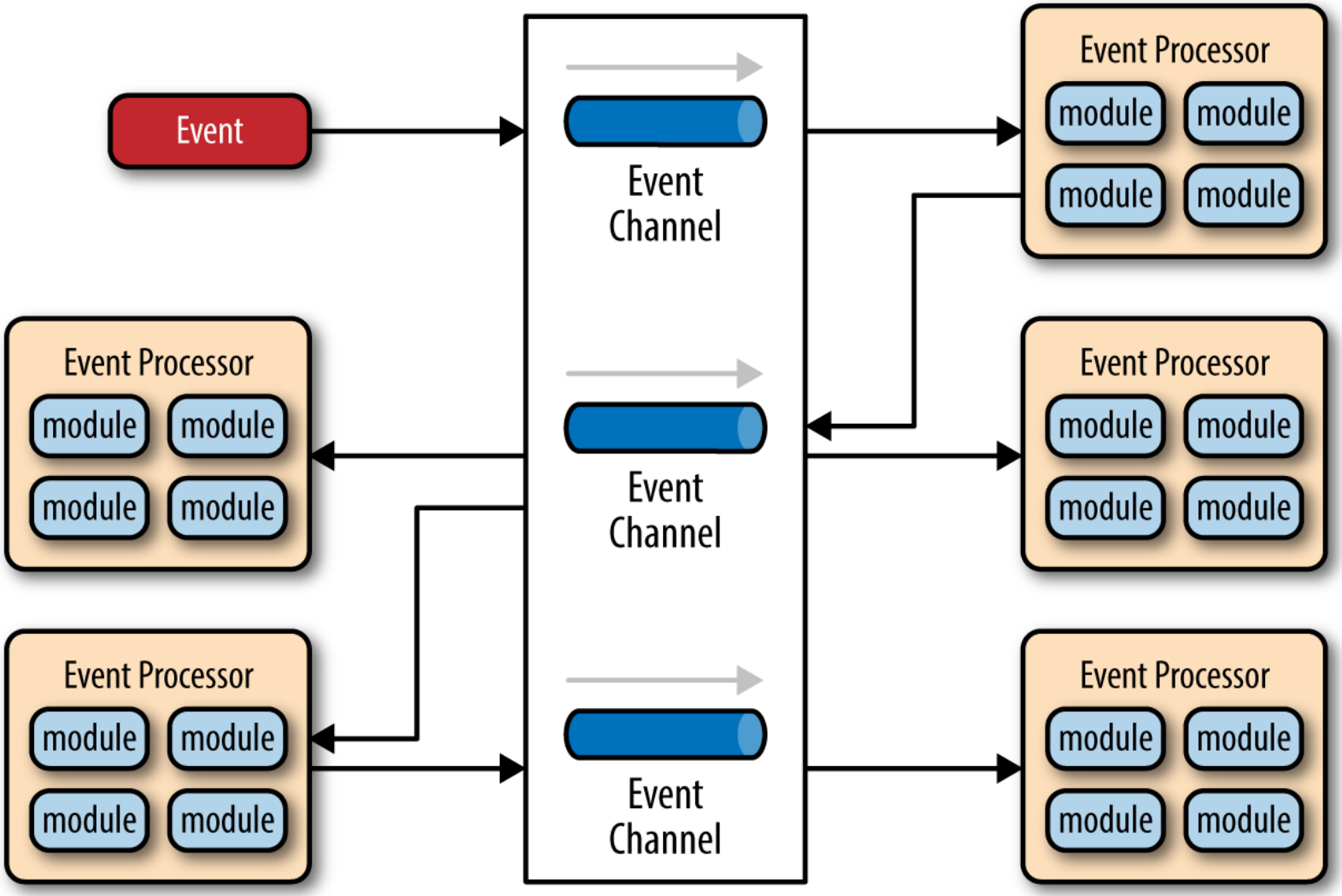
- Структура и алгоритм
  - Система строится из простых напрямую не связанных компонентов, выполняющих единственную функцию
  - Управление системой осуществляется путем генерации и реакции на события, принимаемые компонентами
  - Реакция на событие может быть распространена по разным компонентам и требует оркестровки (автоматическое размещение, координация и управление сложными компьютерными системами и службами)
- Варианты архитектур:
  - С центральным медиатором – пропускает через себя все события, согласовывая их обработку различными компонентами
  - С брокером – события выстраиваются в цепочки (связываются) без использования центрального компонента

Ричардс: Управляемая событиями – 2/3



С медиатором

Ричардс: Управляемая событиями – 3/3

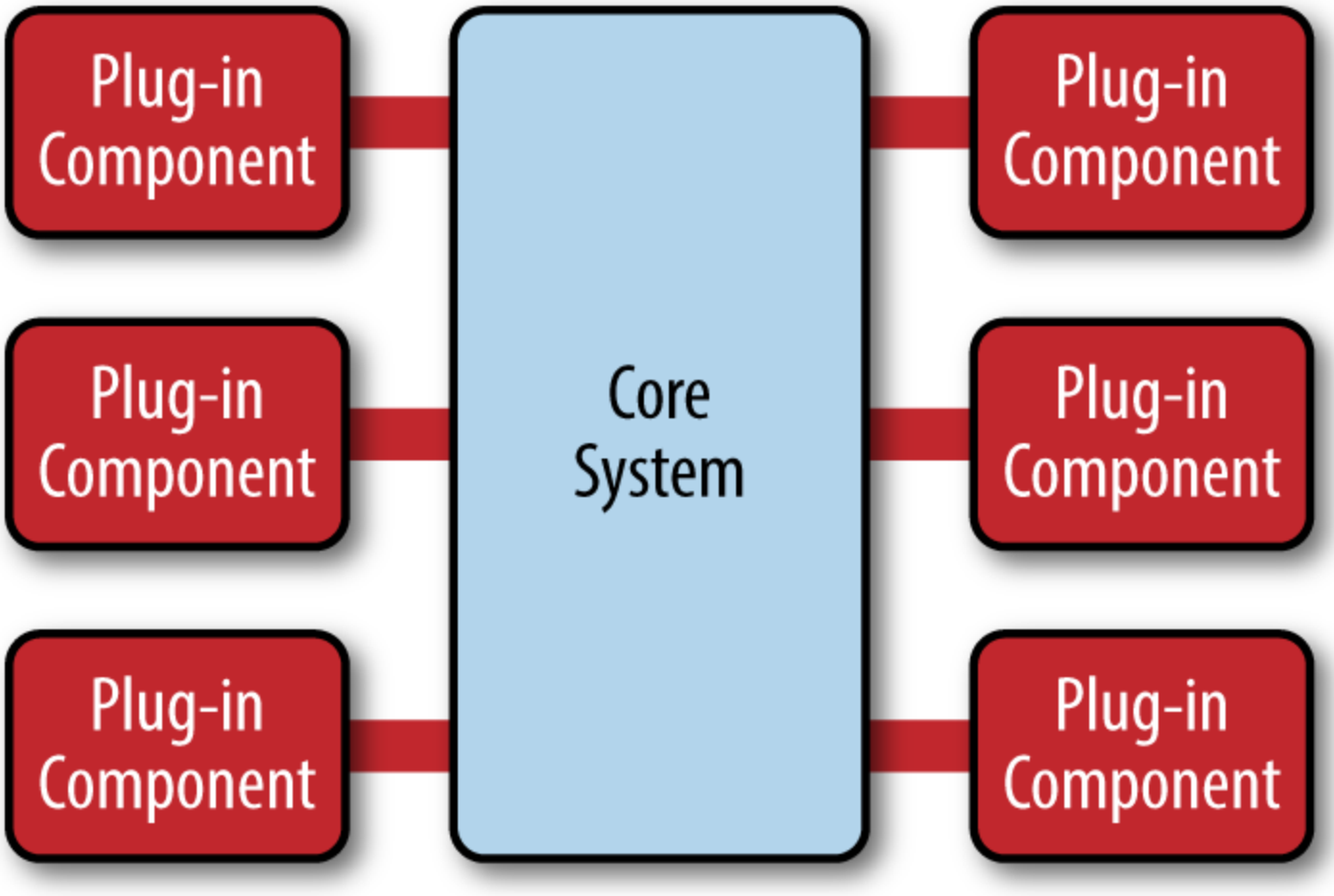


С брокером

Ричардс: Микроядерная – 1/2

- Структура и алгоритм
  - Система включает ядро, выполняющее обобщенный алгоритм
  - К ядру подключаются плагины, реализующие дополнительные функции
- Варианты архитектур:
  - Внешние интерфейсы обеспечивает ядро, а плагины выполняют специфическую обработку данных
  - Интерфейсы реализуются также в виде плагинов

Ричардс: Микроядерная – 2/2



Ричардс: Микросервисная – 1/4

- Структура и алгоритм
  - Каждый сервис размещается как отдельный компонент
  - В системе присутствует уровень Пользовательский интерфейс, обеспечивающий трансляцию пользовательских запросов в сервисные компоненты
  - Сервисные компоненты взаимодействуют друг с другом через «remote access protocol» (напр. JMS, AMQP, REST, SOAP, RMI, etc.)
- Варианты архитектур:
  - С API REST интерфейсом
  - REST топология

Вставка: REST – 1/5

- REST – Representational State Transfer («передача состояния представления») – архитектурный стиль взаимодействия компонентов РИС
  - Расширение клиент-серверной архитектуры
  - Каждый REST-запрос клиента содержит исчерпывающую информацию о желаемом ответе сервера → сервер не обязан сохранять информацию о состоянии клиента (нет клиентской сессии)
  - REST является альтернативой RPC

Вставка: REST – 2/5

- Признаки (требования) REST-системы:
  - Приведение архитектуры к модели клиент-сервер
  - Отсутствие состояния – в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится (Stateless protocol, «протокол без сохранения состояния»)
    - Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса
    - Состояние сессии при этом сохраняется на стороне клиента

Вставка: REST – 3/5

- Признаки REST-системы (продолжение) :
  - Кэширование – клиенты и промежуточные узлы могут выполнять кэширование ответов сервера
    - Ответы сервера, должны иметь обозначение (явное или неявное) как кэшируемые или некэшируемые
    - Используется для повышения производительности и расширяемости системы
  - Слои – клиент обычно не способен точно определить, взаимодействует он напрямую с сервером или же с промежуточным узлом
    - Балансировка нагрузки и распределённое кэширование
    - Промежуточные узлы также могут использоваться для обеспечения конфиденциальности информации

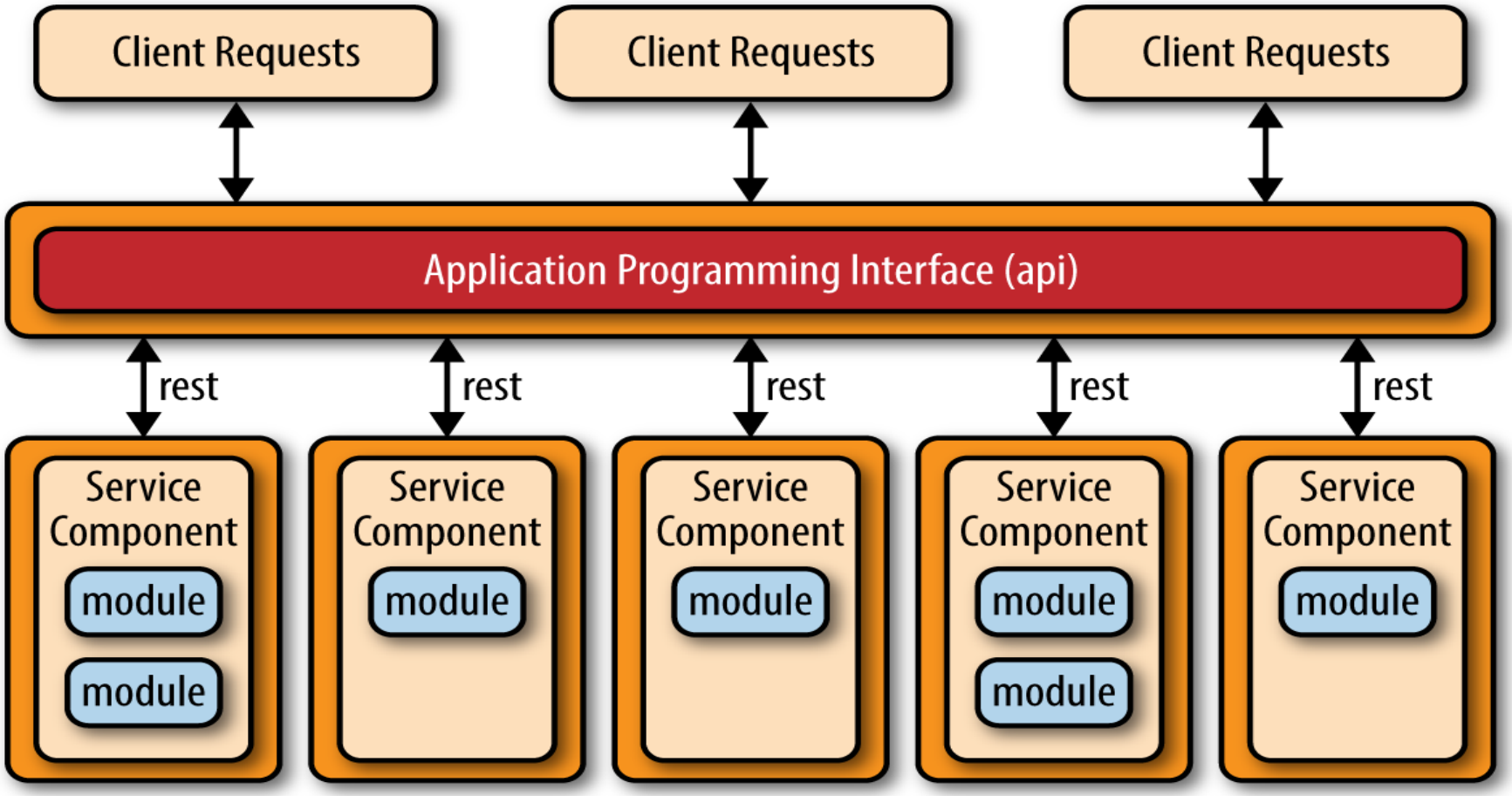
Вставка: REST – 4/5

- Признаки REST-системы (продолжение) :
  - Единообразие интерфейса
    - Идентификация ресурсов
      - Все ресурсы идентифицируются в запросах (e.g. URI)
      - Формат ресурса отличен от формата его поставки клиенту
    - Манипуляция ресурсами через представление
      - Если клиент хранит представление ресурса, включая метаданные — он обладает достаточной информацией для модификации или удаления ресурса.
    - «Самоописываемые» сообщения
      - Есть идентификация типов сообщений
    - Гипермедиа как средство изменения состояния приложения
      - Клиентам доступны действия:
        - » Простые точки входа в приложение
        - » Полученные как ответ на запрос (e.g. гиперссылки в гипертексте).
      - Не существует универсального формата для предоставления ссылок между ресурсами

Вставка: REST – 5/5

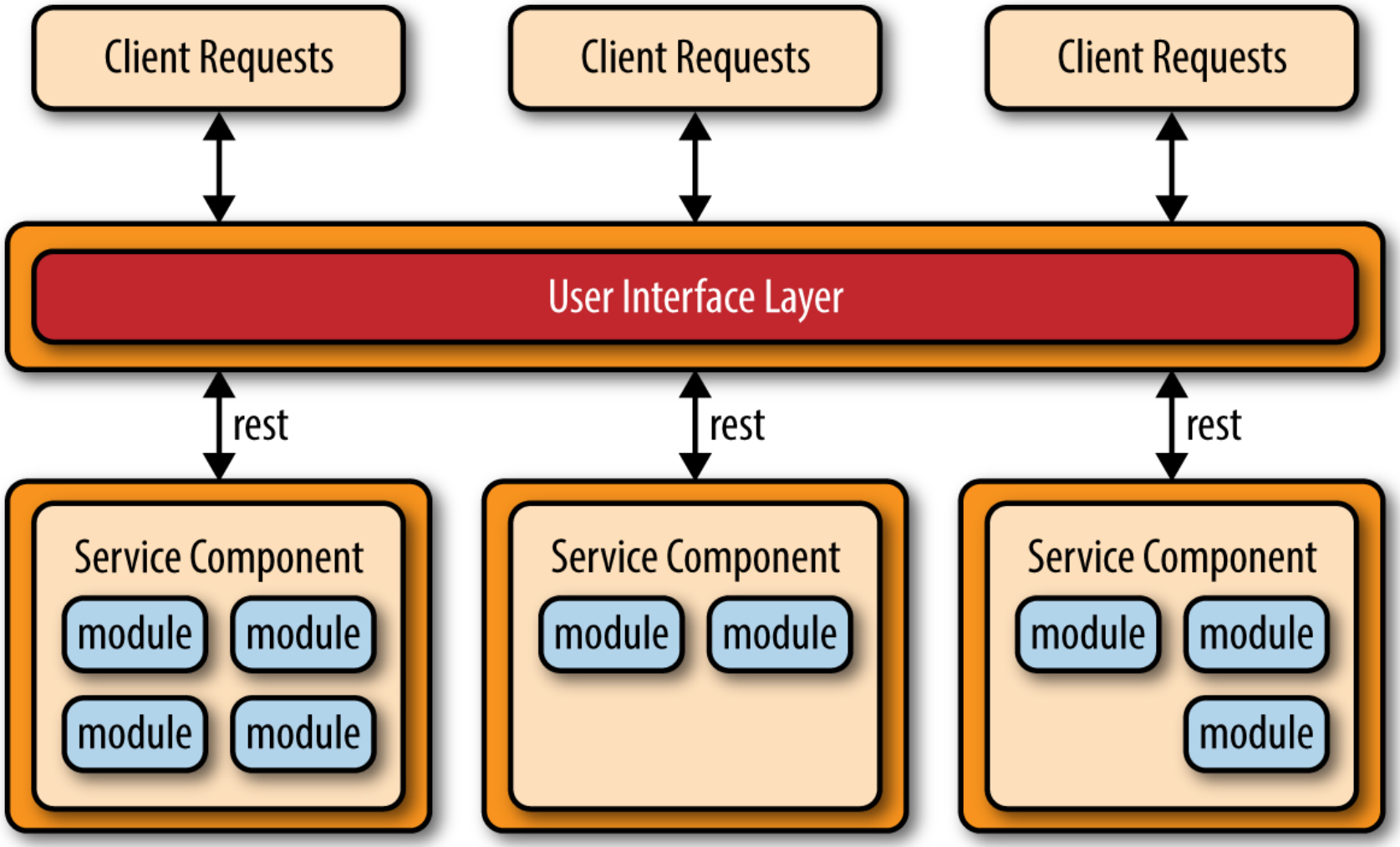
- Признаки REST-системы (продолжение) :
  - Код по требованию (необязательное ограничение)
    - Можно расширить функциональность клиента за счёт загрузки кода с сервера в виде апплетов или сценариев

Ричардс: Микросервисная – 2/4



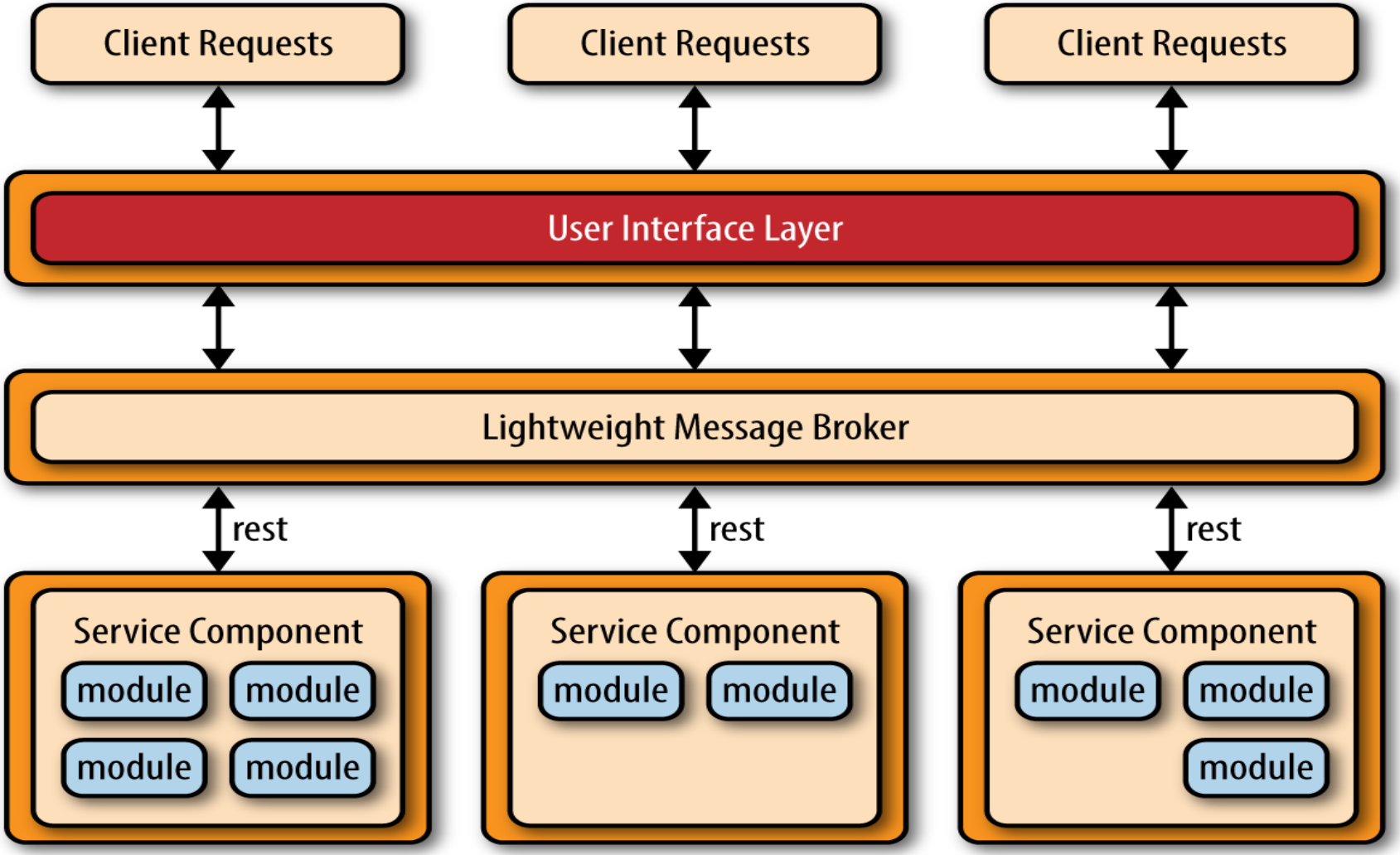
С API → REST интерфейсом

Ричардс: Микросервисная – 3/4



С UI → REST интерфейсом

Ричардс: Микросервисная – 4/4

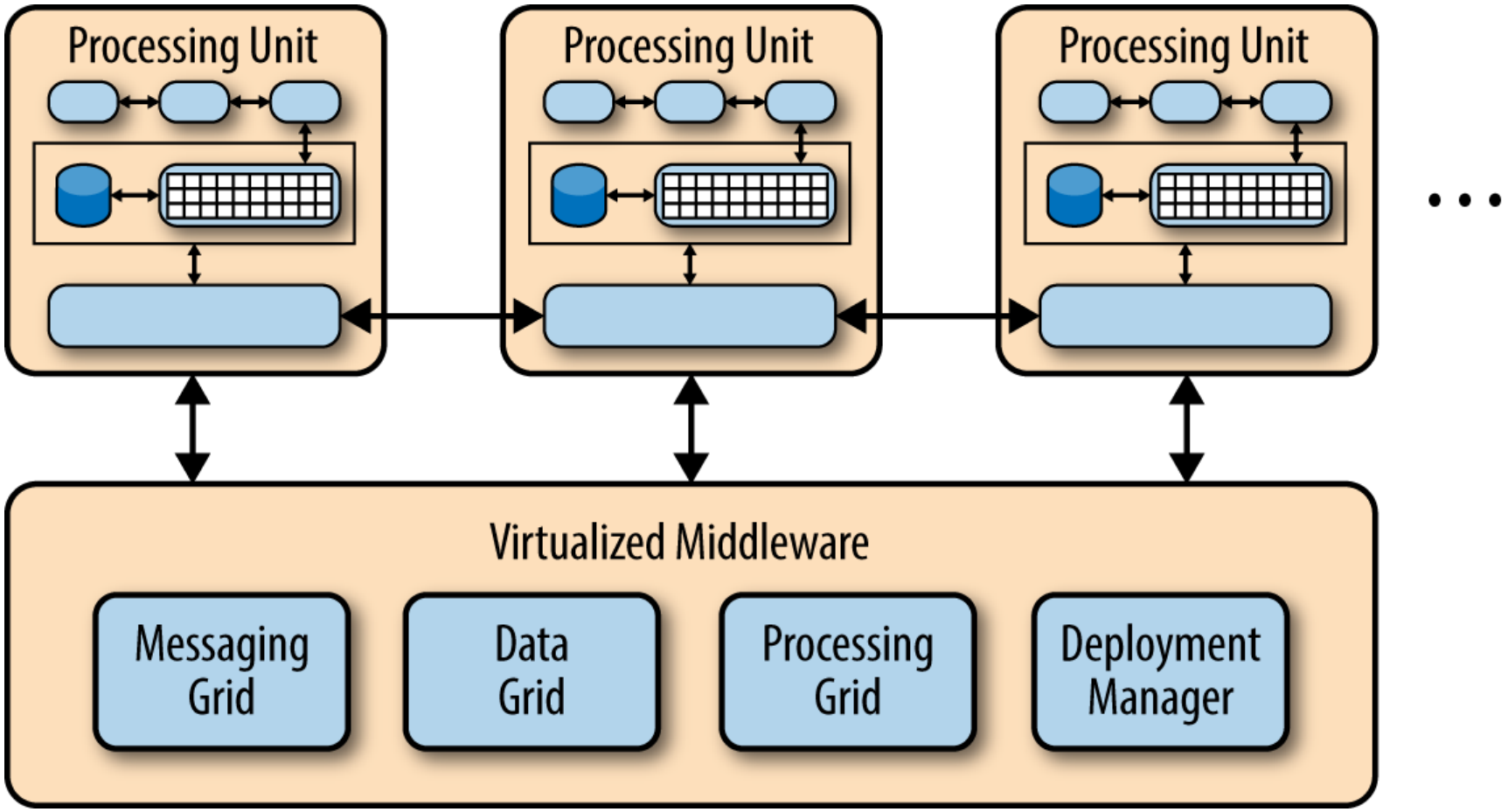


С брокером сообщений

Ричардс: Облачная – 1/8

- Структура и алгоритм
  - Вместо единой БД используются сети данных (data grids) в памяти, реплицируемые по хостам по мере необходимости
  - Два ключевых компонента архитектуры: обрабатывающий модуль (processing unit) и «виртуализированное» связующее ПО (virtualized middleware)

Ричардс: Облачная – 2/8

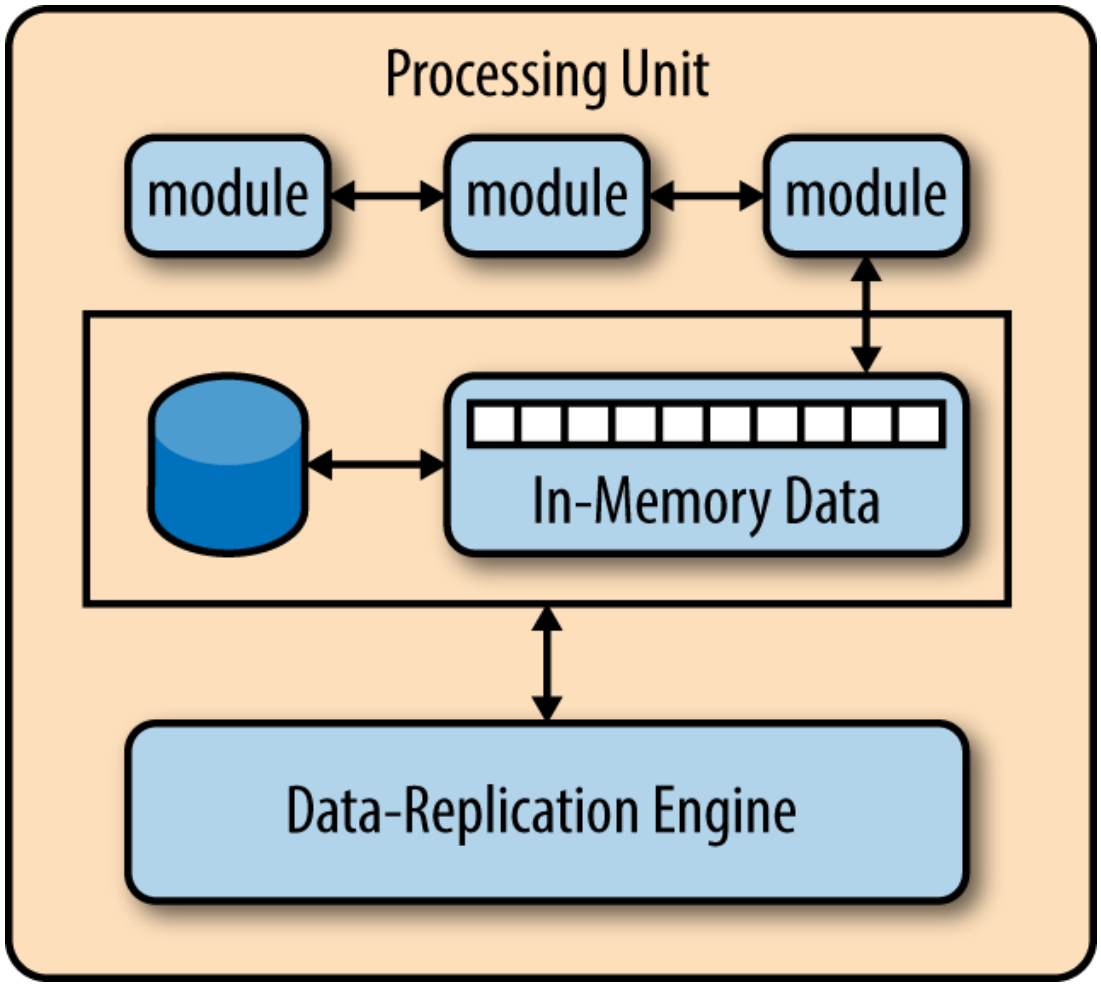


Обобщенное представление облачной архитектуры

Ричардс: Облачная – 3/8

- Обрабатывающий модуль содержит приложения (или их части), включая:
  - Реализацию основной бизнес-логики (backend business logic)
    - В зависимости от соотношения сложности приложений и производжительности обрабатывающих модулей одно приложение может размещаться на одном или быть разделено по нескольким модулям
  - Интерфейсные веб-компоненты, а также:
  - Часть сети данных в оперативной памяти
    - Возможно также наличие постоянной памяти для восстановления после сбоев
  - ПО репликации и синхронизации содержимого сети данных

Ричардс: Облачная – 4/8



Структура обрабатывающего модуля

Ричардс: Облачная – 5/8

- «Виртуализированное» связующее ПО отвечает за поддержки функционирования и обмен данными. Оно содержит компоненты, отвечающие за синхронизацию во всем ее разнообразии и обработку запросов. Связующее ПО включает:
  - Сеть сообщений (messaging grid)
  - Сеть данных
  - Сеть обработки
  - Менеджер размещения

Ричардс: Облачная – 6/8

- Сеть сообщений (messaging grid)
  - Отвечает за входящие запросы и сессии обслуживания
  - После получения запроса сеть сообщений выбирает доступный активный обрабатывающий модуль и передает ему запрос на обработку
  - Алгоритм выбора может варьироваться от простого последовательного перебора (round-robin) до предиктивных алгоритмов, отслеживающих состояние прохождения текущих запросов в обрабатывающих модулях

Ричардс: Облачная – 7/8

- Сеть данных
  - Управляет ПО репликации и синхронизации содержимого сети данных в каждом обрабатывающем модуле с тем, чтобы обновления данных в модуле распространялись на все копии
  - Так как запрос может быть отправлен в произвольный обрабатывающий модуль такая синхронизация крайне важна
  - Синхронизация выполняется асинхронно с высокой скоростью (порядка микросекунд)

Ричардс: Облачная – 8/8

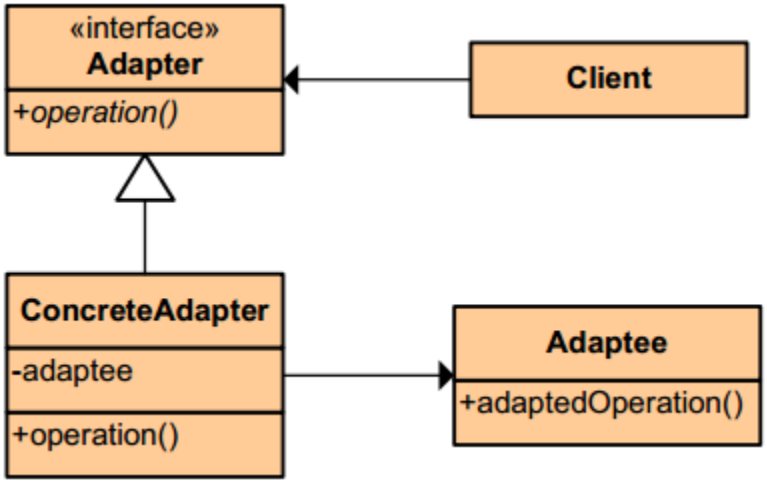
- Сеть обработки
  - Необязательная составная часть связующего ПО, требующаяся в случае, обработка запроса выполняется несколькими обрабатывающими модулями
- Менеджер размещения
  - Реализует концепцию «мощности по запросу», динамически запуская и останавливая обрабатывающие модули в зависимости от текущей загрузки системы
  - Отслеживает времена ответа на запросы и количество запросов в единицу времени
  - Критический компонент обеспечивающий масштабируемость облачных систем

ТИПОВЫЕ ШАБЛОНЫ ПРОЕКТИРОВАНИЯ

Шаблоны проектирования

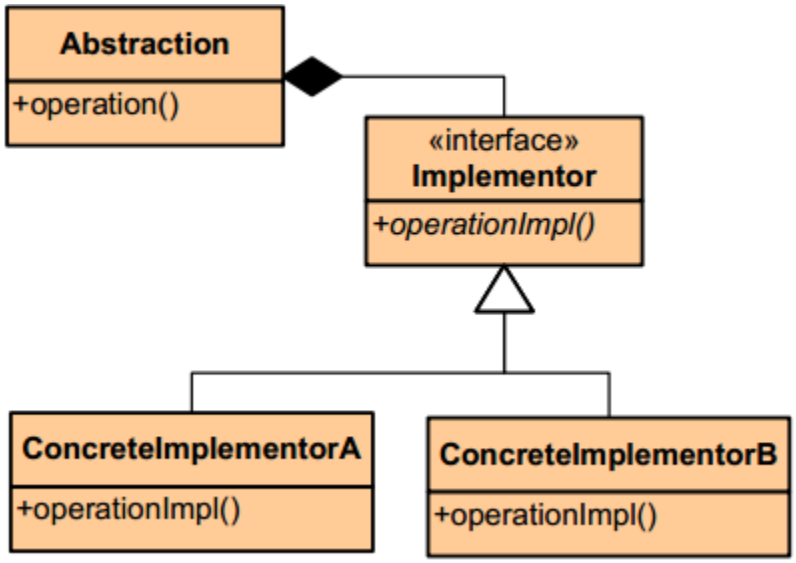
- Шаблон (паттерн) проектирования (design pattern) – повторяемая архитектурная конструкция, представляющая собой вариант решения проблемы проектирования в рамках некоторого часто возникающего контекста
- Проще: Задача + Ограничения → Повторяющийся способ решения
- Выделяют [как минимум] следующие структурные паттерны:
  - Адаптер
  - Мост
  - Компоновщик
  - Декоратор
  - Фасад
  - Прокси

Адаптер



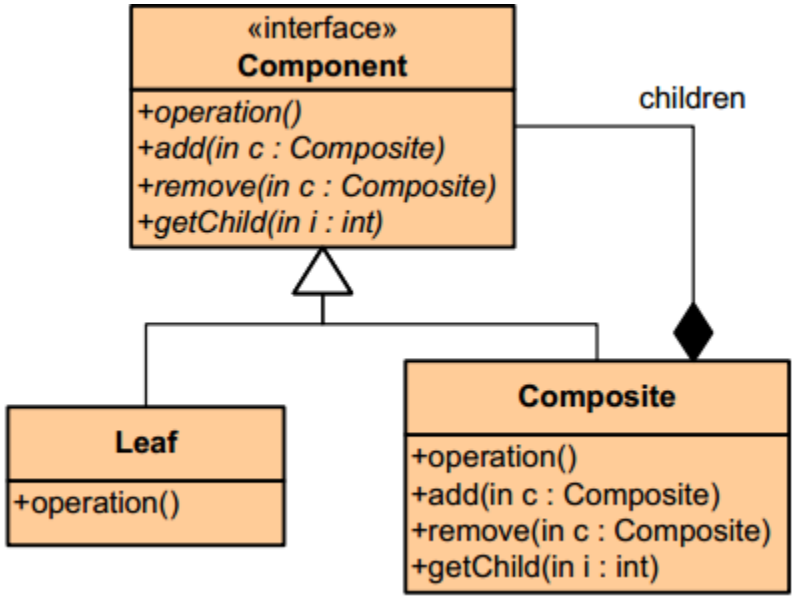
- Объект, обеспечивающий взаимодействие двух других объектов, один из которых использует второй. При этом второй объект предоставляет синтаксически несовместимый с первым интерфейс. При этом семантически и прагматически они совместимы
- Адаптеры могут работать на уровнях:
  - Синтаксическом
  - Семантическом
  - Прагматическом

Мост



- Структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо. Является расширением принципа инкапсуляции, так как позволяет менять не только реализацию, но и интерфейс

Компоновщик



- TBD

Зачем

Как

ПОСТРОЕНИЕ АРХИТЕКТУРЫ

Архитектура

- Проектирование можно разбить на:
  - Разработка архитектуры
  - Детальная разработка проекта
- Архитектура – представление структуры и поведения системы на самом верхнем уровне

Критерии выбора архитектуры

- Простота:
  - Понимания
  - Реализации
- Расширение
  - Добавление функциональности
  - Масштабирование
- Изменение – смена требований
- Эффективность:
  - Скорость выполнения
  - Малый размер

Декомпозиция

- Цели
  - Связность внутри модуля – максимизация
  - Сцепление – минимизация
- Рекомендуемое число модулей одного уровня: 7±2
- Последовательное разбиение модулей верхнего уровня на более мелкие – рекурсивное проектирование

Вопросы

- Архитектура:
  - Как сделать: Алгоритм построения архитектуры
  - Какая: Образцы проектирования, выбор шаблона архитектуры
- Описание архитектуры с помощью UML
  - Activity diagram
  - Sequence diagram
  - Class diagram
  - Deployment diagram (доработать)
  - Sequence diagram (доработать)
  - ToDo: Component diagram

Как сделать?

Какая?

ПОСТРОЕНИЕ АРХИТЕКТУРЫ

Алгоритм построения архитектуры – 1/2

- Разбейте систему на замкнутые модули
- Сравните со стандартными архитектурами → улучшите декомпозицию
  - Создайте несколько (sic!)
- Сделайте выбор среди альтернативных
- Добавьте к классам, полученным из требований, классы, обеспечивающие согласование с выбранной архитектурой
- Примените шаблон проектирования, если найдете полезный

Алгоритм построения архитектуры – 2/2

- Распределите классы по пакетам
  - Практическое правила:
    - 4-8 пакетов на уровне
    - В больших проектах использовать иерархию
- Проверить признаки «хорошего» разбиения:
  - Связность – высокая
  - Сцепление – низкое
- Рассмотрите возможность добавления facade-класса (объекта) для управления интерфейсами пакетов

Образцы проектирования – 1/2

- Образец проектирования – найденная опытным путем комбинация компонентов, решающая общие задачи проектирования
- Полезные образцы:
  - Facade – единый интерфейс для множества объектов различных классов
  - State – объект ведет себя в соответствии с текущим состоянием
  - Iterator – позволяет выбирать пути «посещения» объектов в коллекции во время выполнения клиентского кода

Образцы проектирования – 2/2

- Observer – обеспечивает реакцию нужных элементов на изменения в источнике данных
- Interpreter – интерпретирует выражения на языке формальной грамматики

Выбор стандартной архитектуры – 1/4

Признак	Стандартная архитектура
Есть поток данных между обрабатывающими станциями?	Архитектура последовательных пакетов
Отдельный элемент данных не имеет семантики, только поток?	Архитектура каналов и фильтров
Процесс обеспечивает обслуживание пользовательских процессов?	Архитектура клиент-сервер

Выбор стандартной архитектуры – 2/4

Признак	Стандартная архитектура
Процесс реагирует только на происходящие события?	Системы, управляемые событиями
Приложение состоит из процессов, которые выполняются по сценарию?	Архитектура «интерпретатор»
Есть правила, распознающие [особые] ситуации в наблюдаемом мире и реакция на них?	Системы, основанные на правилах

Выбор стандартной архитектуры – 3/4

Признак	Стандартная архитектура
Приложение строится для хранения данных?	Репозиторные архитектуры
• Данные формализованы общим для всех образом?	БД
• Формат одинаков для пар приложений или маленьких подгрупп?	Доска объявлений
• Данные оформлены в виде гипертекста?	Гипертекстовые системы

Самые общие архитектуры – 4/4

Признак	Стандартная архитектура
Существует упорядочение по уровням?	Уровневые архитектуры
Процессы выполняются параллельно?	Архитектура параллельных взаимодействующих процессов NB! Не забыть, что процессы независимы (появление, исчезновение, сбои, etc.)

Стандарты

Состав

Зависимость проектных решений

ОПИСАНИЕ АРХИТЕКТУРЫ

Стандарты: мировые (и наши) – 1/4

- Группа стандартов ISO/IEC/IEEE Systems and software engineering – системная и программная инженерия, в том числе:
  - ISO/IEC/IEEE 42010:2011 Systems and software engineering. Architecture description – Описание архитектуры
  - ISO/IEC/IEEE 15288:2015 – Systems and software engineering. System life cycle processes – Процессы ЖЦ систем в целом (включая процессы создания архитектуры)

Стандарты: наши – ЕСКД – 2/4

- ЕСКД – Единая система конструкторской документации:
  - ГОСТ 2.103-68 ЕСКД. Стадии разработки
  - ГОСТ 2.119-73 ЕСКД. Эскизный проект
  - ГОСТ 2.120-73 ЕСКД. Технический проект

Стандарты: наши – ЕСПД – 3/4

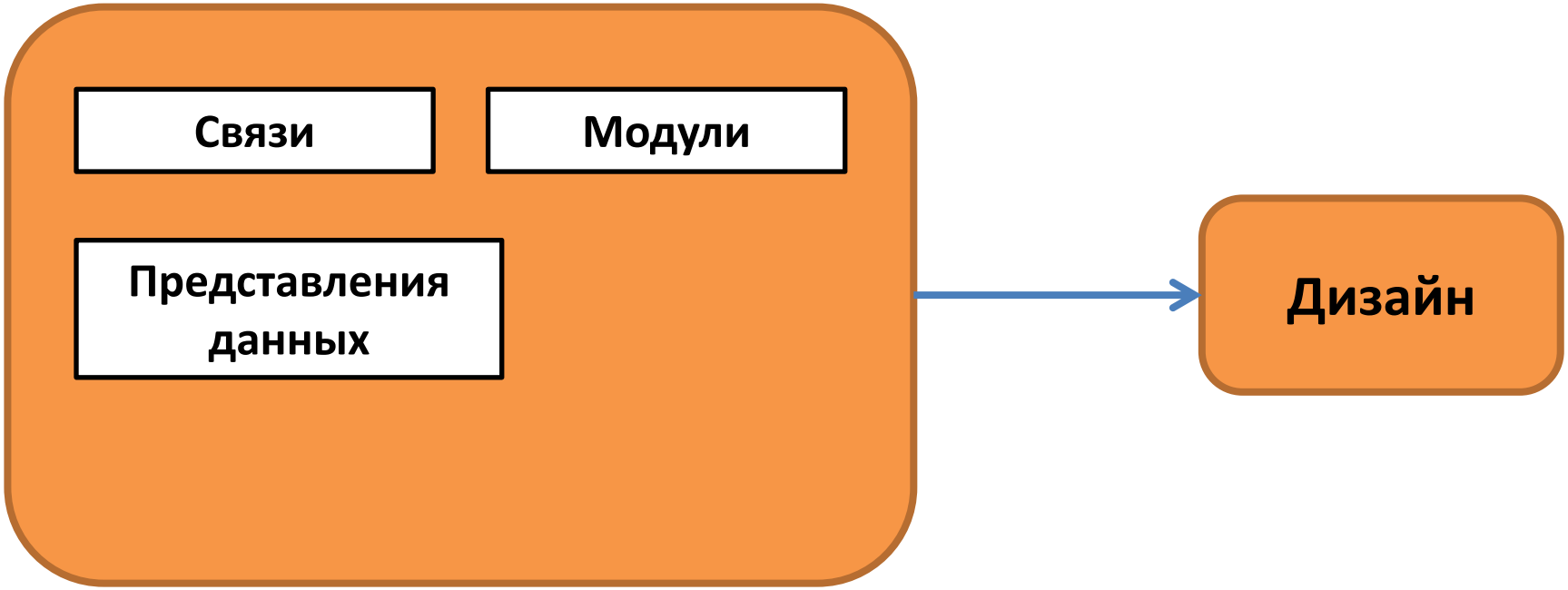
- ЕСПД – Единая система программной документации:
  - ГОСТ 19.001-77. ЕСПД. Общие положения
  - ГОСТ 19.701-90. (ИСО 5807-85). ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения
  - ГОСТ 19.781-90. Обеспечение систем обработки информации программное. Термины и определения
  - ГОСТ 19.005-85. ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения

Стандарты: наши – ЕСПД – 4/4

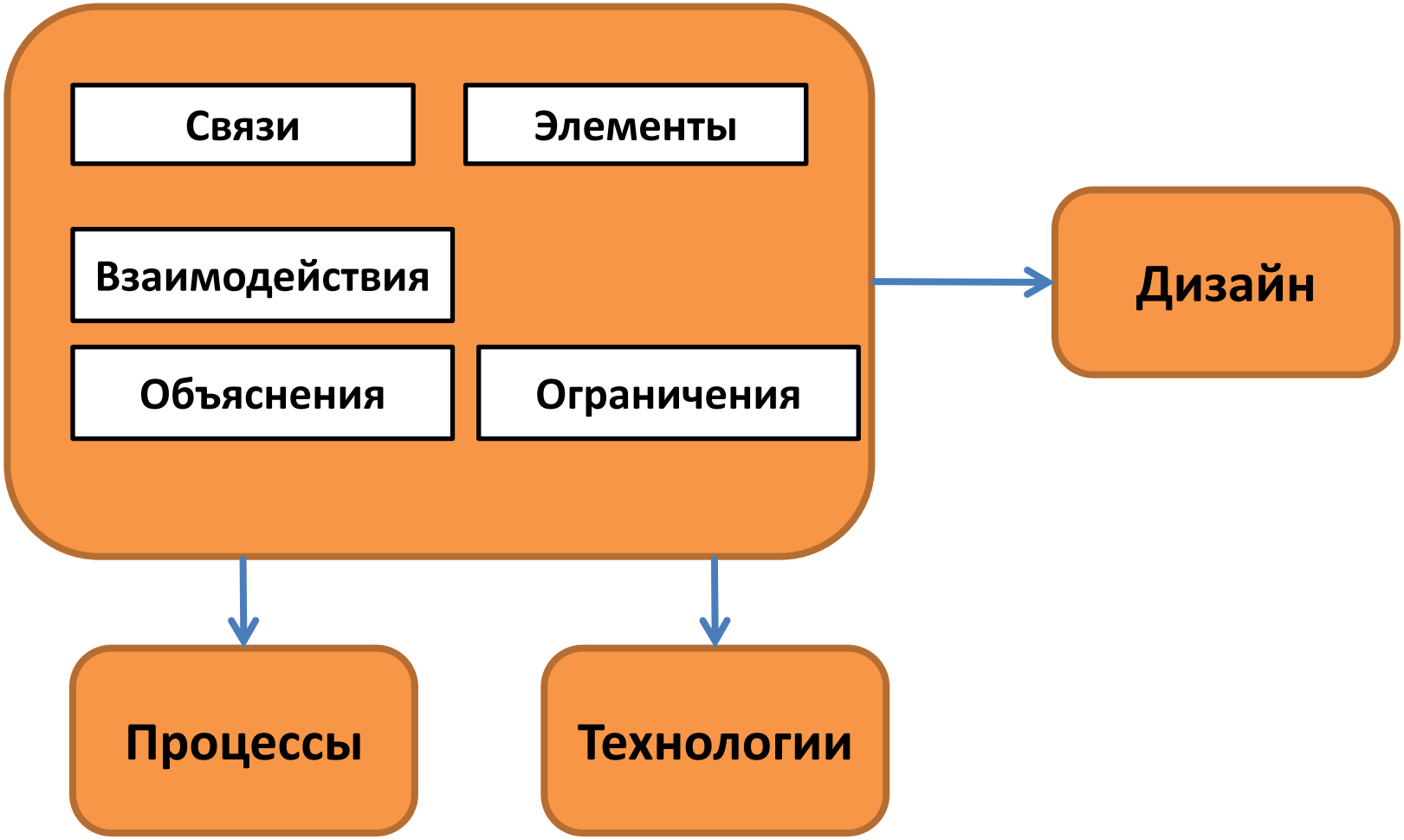
- ЕСПД (продолжение):
  - ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов
  - ГОСТ 19.102-77. ЕСПД. Стадии разработки
  - ГОСТ 19.402-78. ЕСПД. Описание программы
  - ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению
  - ГОСТ 19.502-78. ЕСПД. Описание применения. Требования к содержанию и оформлению

Что входит в архитектуру – 1/4

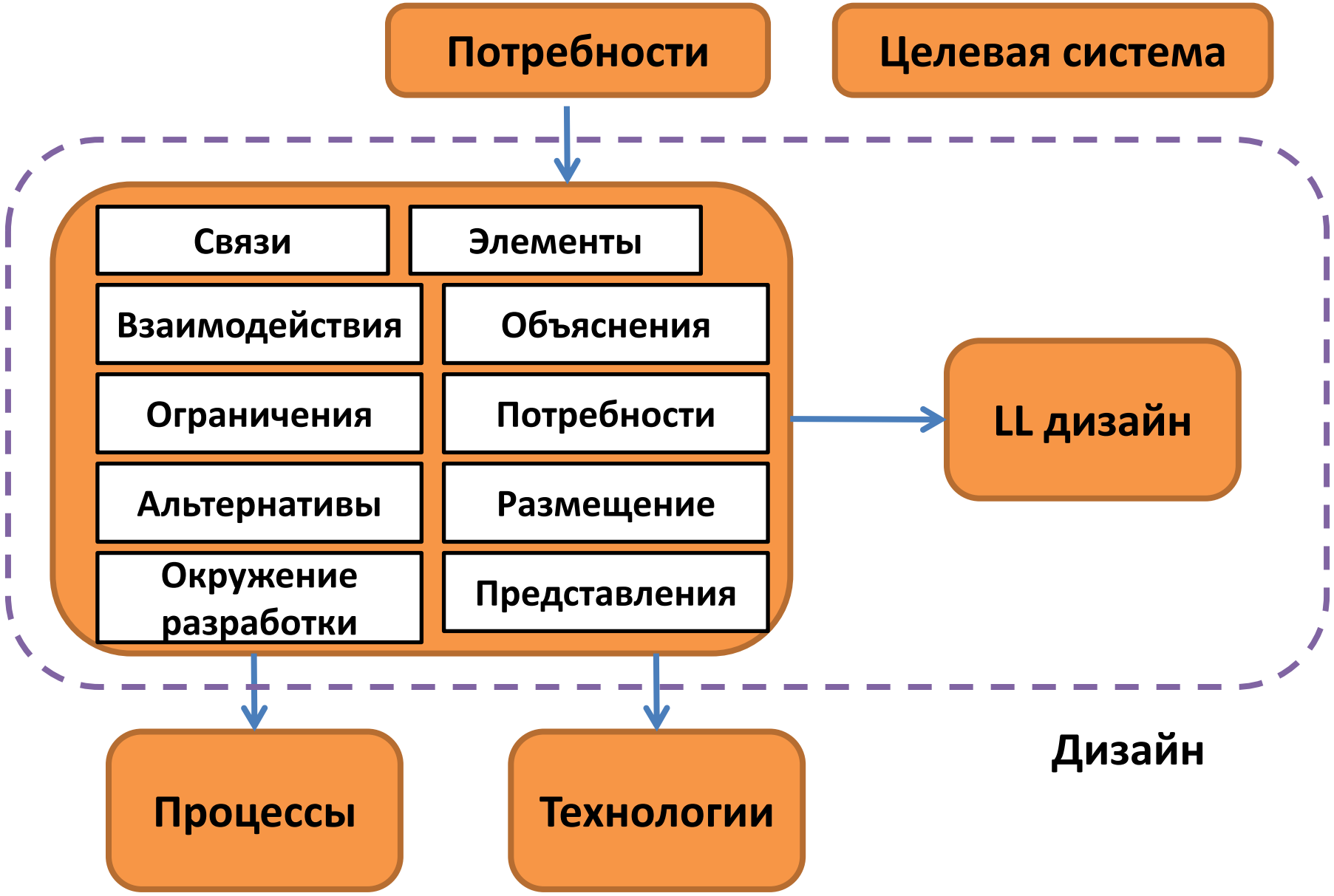
До 1990-х



Что входит в архитектуру – 2/4



Что входит в архитектуру – 3/4

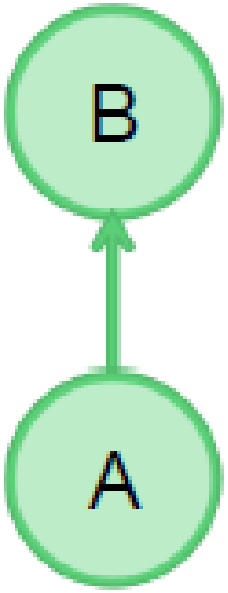


Что входит в архитектуру – 4/4

- Элементы, модули, связи, структуры, виды, формы, каркасы, разделения, ограничения, интересанты, потребности, окружение, размещение, управление, интерфейсы, слои, артефакты, технологии, статика и динамика, поведение, свойства, отношения, принципы, стили…
- → То есть самые разнообразные **проектные решения**

Зависимость проектных решений – 1/7

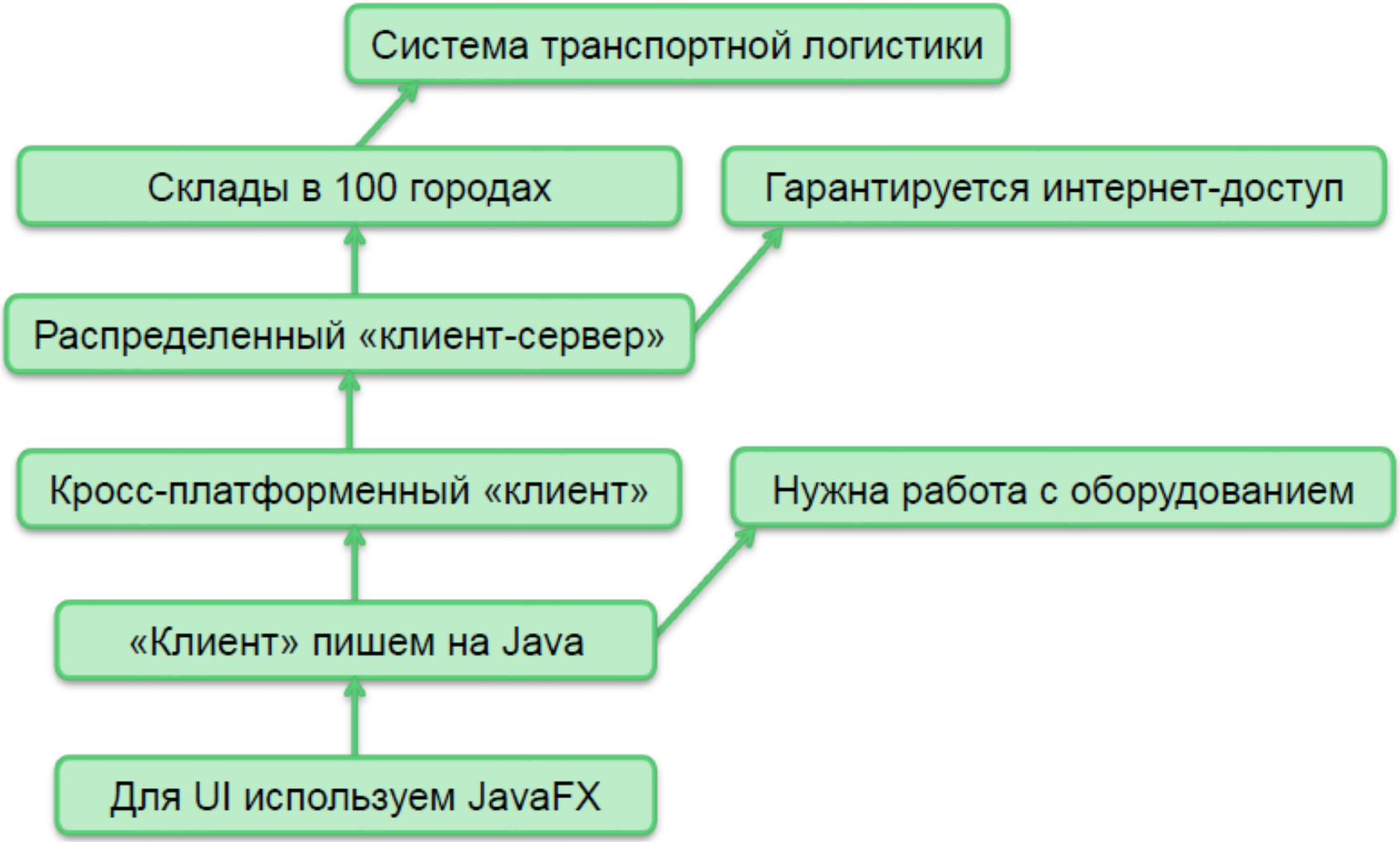
- Решение А зависит от решения В, если А имеет смысл или целесообразно только в том случае, когда принято и актуально решение В



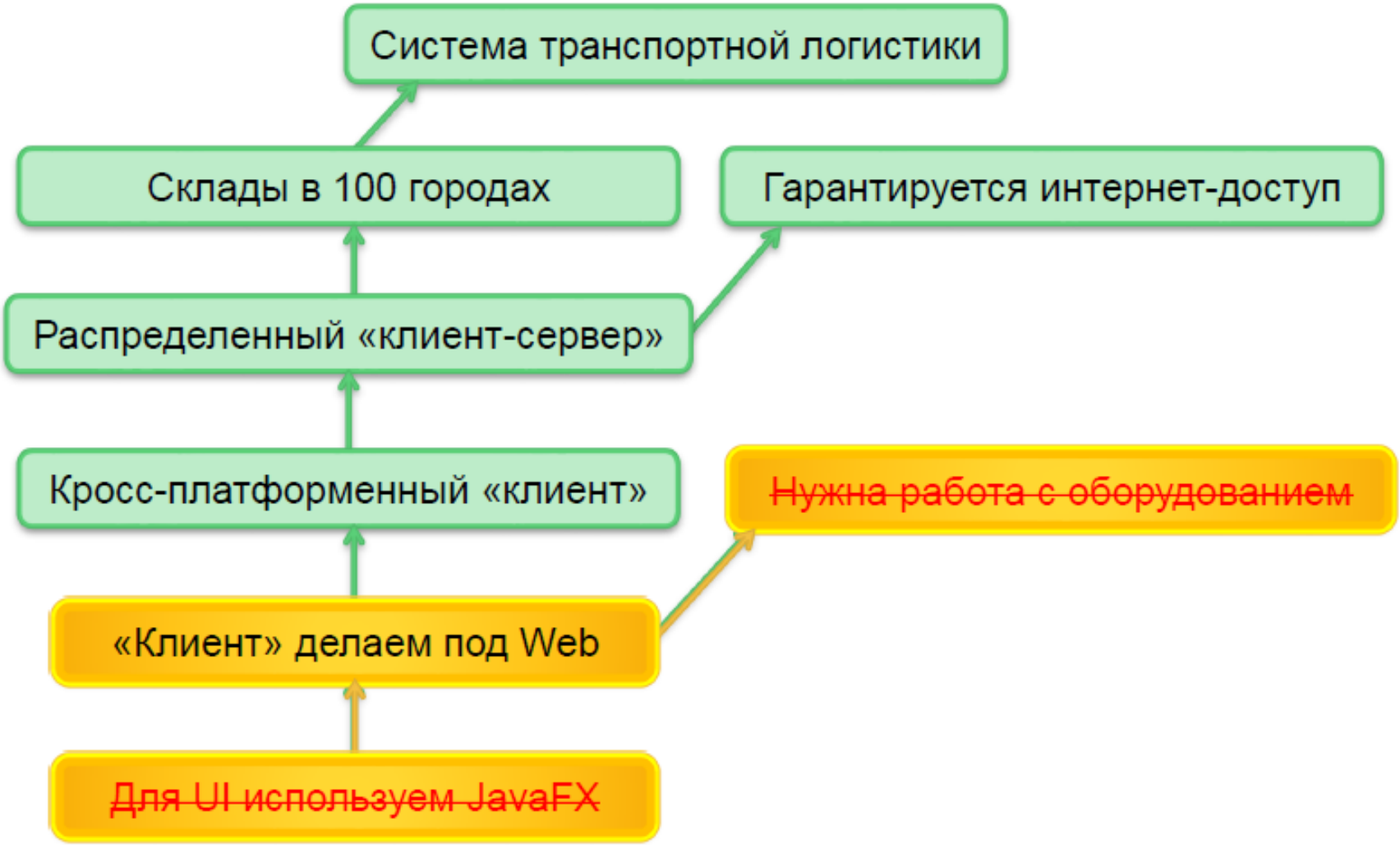
Зависимость проектных решений – 2/7

- Зависимости не бывают цикличны, решения образуют направленный граф («дерево решений»)
- Если изменяется некоторое решение, то придется пересмотреть все решения, которые прямо или косвенно зависят от него
- Фундаментальные решения (содержание архитектуры) – решения, от которых зависит слишком много и изменение которых обойдется слишком дорого

Зависимость проектных решений – 3/7

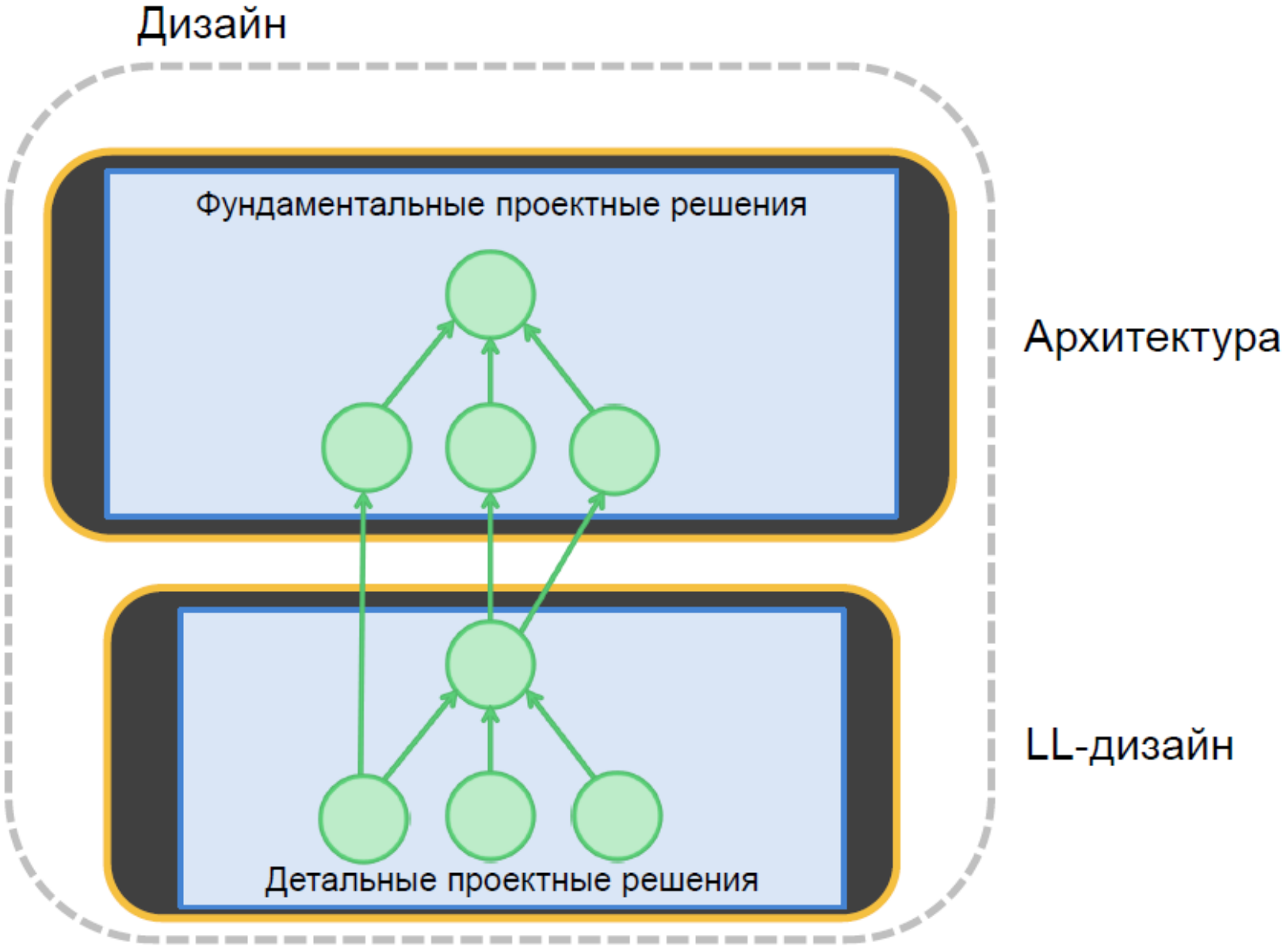


Зависимость проектных решений – 4/7

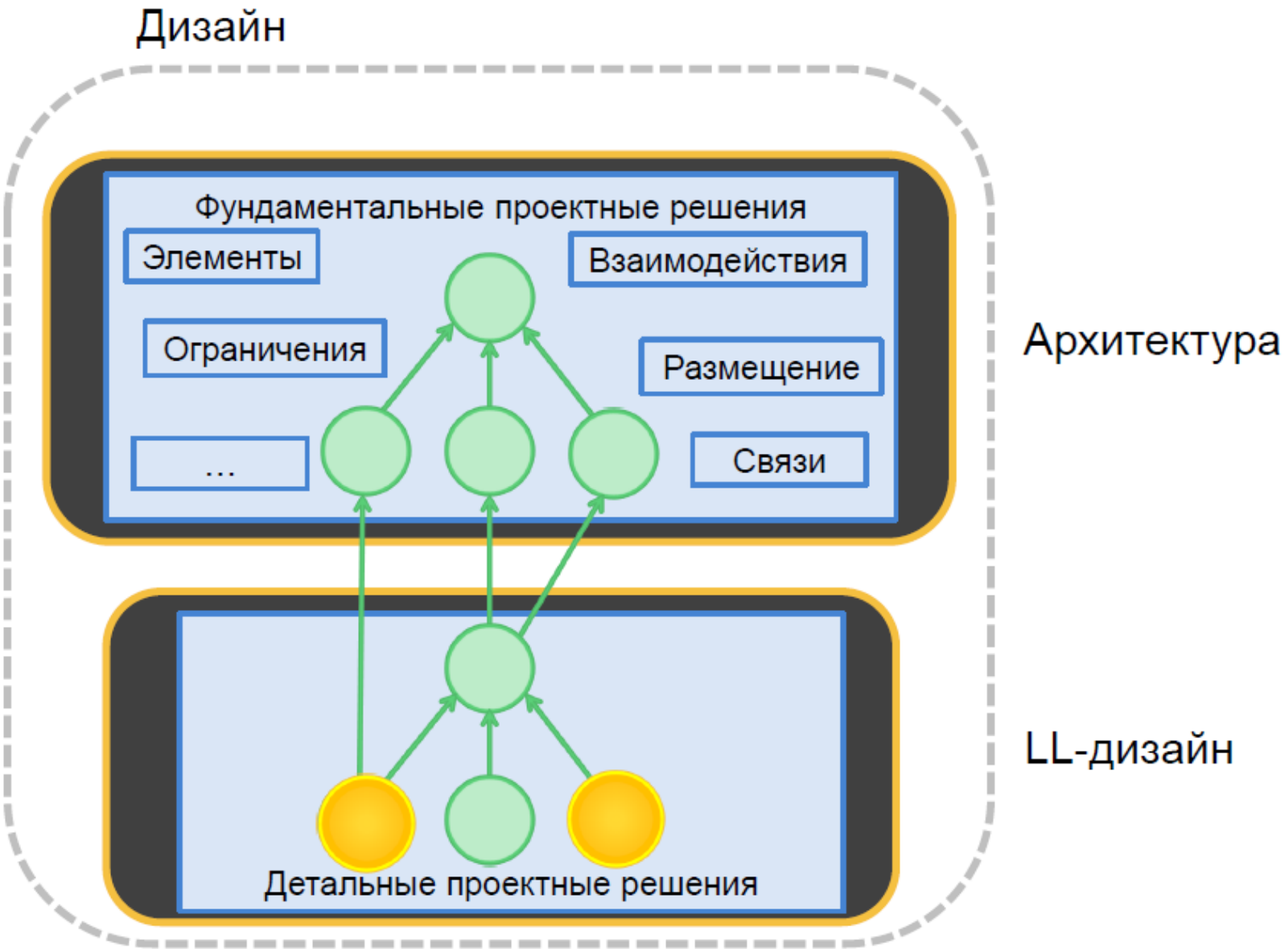


Зависимость проектных решений – 5/7

Зависимость проектных решений – 6/7



Зависимость проектных решений – 7/7



Цели проектирования

«Плохое» и «хорошее» проектирование

РАЗРАБОТКА АРХИТЕКТУРЫ

Как ставить цель архитектору?

- Назначение системы
- Атрибуты качества
- Ограничения (включая $ и t)
- Свойства окружения
  - Designtime
  - Runtime
- Не любые требования и ограничения!
  → Только фундаментальные (нужен анализ)

«Плохое» проектирование – 1/12а

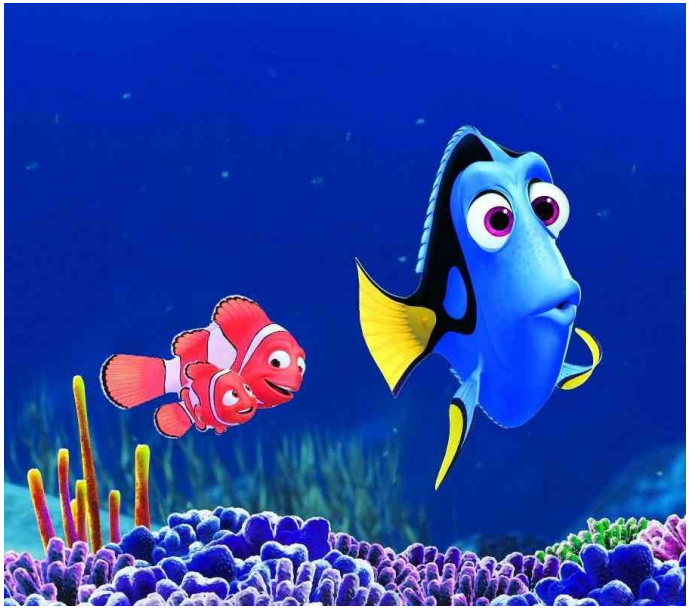
- Лима
  - Архитектуры нет (нет целого)
  - Про целое ничего нельзя сказать, кроме объема

«Плохое» проектирование – 2/12a

- Скрыта за кодом
  - Архитектура есть, но «скрыта за кодом» → не проявлена
  - Кто-то придумал, остальные постоянно «реверсируют» (с переменным успехом), ответственность потеряна
  - Обсуждение предмета невозможно

«Плохое» проектирование – 3/12а

- Memento
  - Решения не фиксируются, забываются
  - Принимаются заново
  - Или искажаются из-за потери оснований

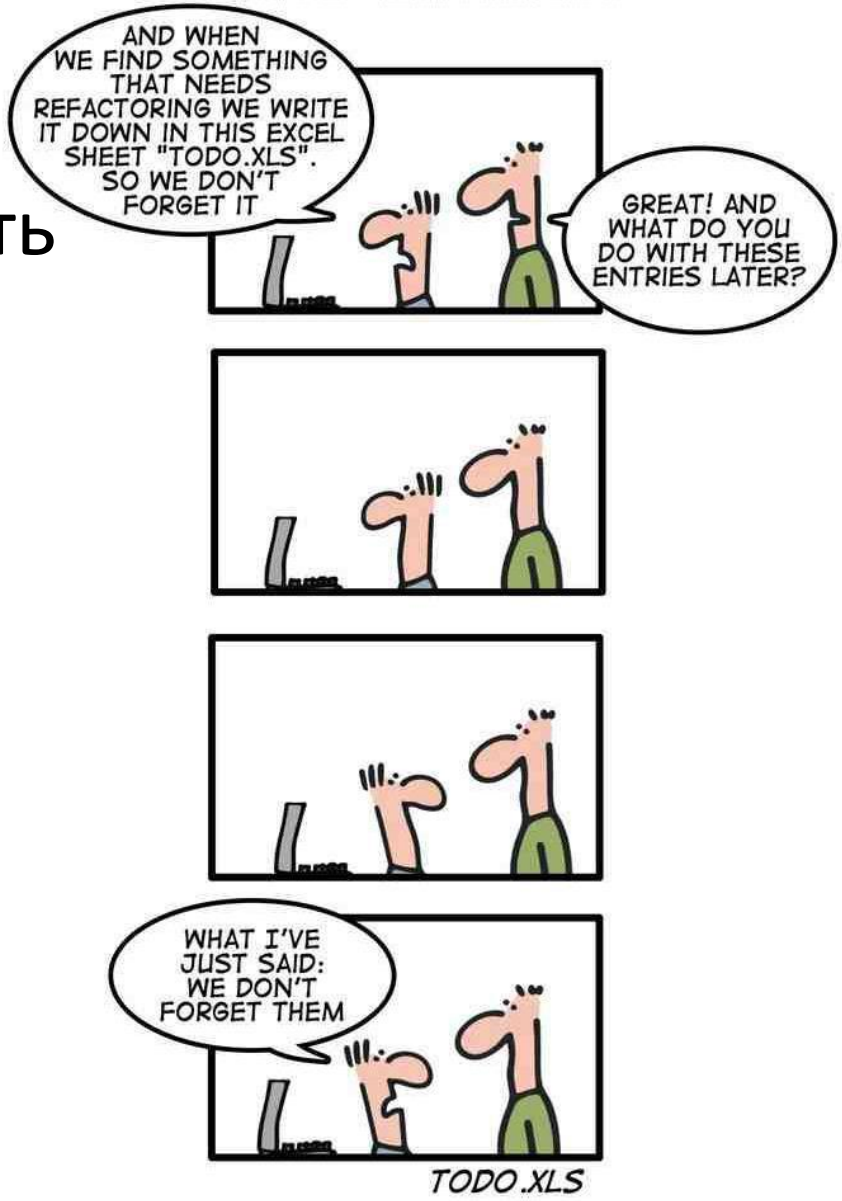


«Плохое» проектирование – 4/12а

- Big Design Up Front
  - Попытка спроектировать все до мелочей и сразу
  - Все проектирование произвести до кодирования
  - Весь дизайн запихивается в архитектуру

«Плохое» проектирование – 5/12а

- Технический долг
  - Осознание долга есть
  - Но с ним ничего не делается



«Плохое» проектирование – 6/12а

- Тайное качество
  - Ненаправленность на требуемое качество или работа на «внутреннее» качество
    - → Не бывает «внутреннего качества»! то есть такого, которое не обеспечивает внешнее, может быть, на долгом периоде
  - Само требуемое качество (как краткосрочное, так и долгосрочное) тоже нуждается в анализе и фиксации
  - Здесь же: мода, авангардизм

«Плохое» проектирование – 7/12а

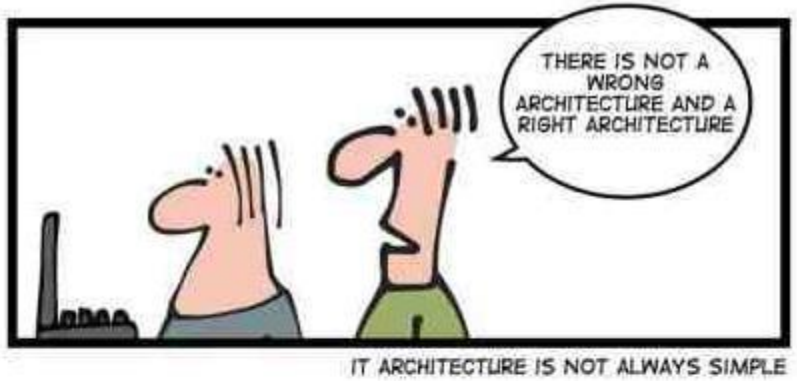
- Башня из слоновой кости
  - Архитектор творит нечто, что ему кажется прекрасным
  - Всем остальным это непонятно и неудобно
  - Сам архитектор при этом «код не пишет» –он выше этого

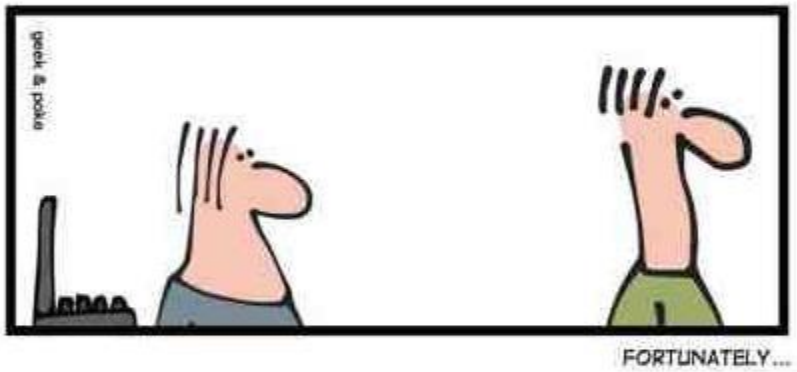
«Плохое» проектирование – 8/12а

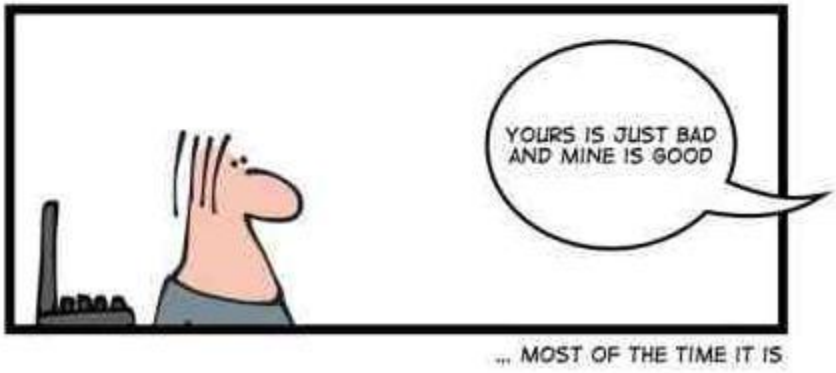
- Анархический Agile
  - Отрицается любое главенство
  - Страшилки Ivory Tower и BDUF как оружие
  - «Мы все равны»
  - И поэтому никто не отвечает
  - Принцип «равенства»важнее результата

«Плохое» проектирование – 9/12а

- Главенство авторитета
  - Главенство авторитета над целями
  - Переход на личности – сравниваются не решения, а регалии архитекторов
  - Статус важнее результата







«Плохое» проектирование – 10/12а

- Лучший программист
  - Архитектор = самый сильный программист?
  - Специфический предмет. Специфический материал и свойства
  - Специфические компетенции

«Плохое» проектирование – 11/12а

- После нас – хоть потоп
  - Успех оценивается по короткому результату (сдача системы)

«Плохое» проектирование – 12/12а

- Отлито из бронзы
  - «Архитектура от старой системы проверена, возьмем для новой системы ее»
  - Фрэнк Ллойд Райт: «Забудьте обо всех архитектурах мира, если не понимаете того, что они были хороши в своём роде и в своё время»

«Плохое» проектирование – 12а/12а

- Швейцарский нож
  - Универсальная архитектура – одна на сильно непохожие (но кажущиеся похожими) проекты
    - «Там же везде БД, документы, трехзвенка и гриды»
  - Неверно выбран масштаб повторного использования, получается сложно и дорого
  - Software Product Lines

«Хорошее» проектирование

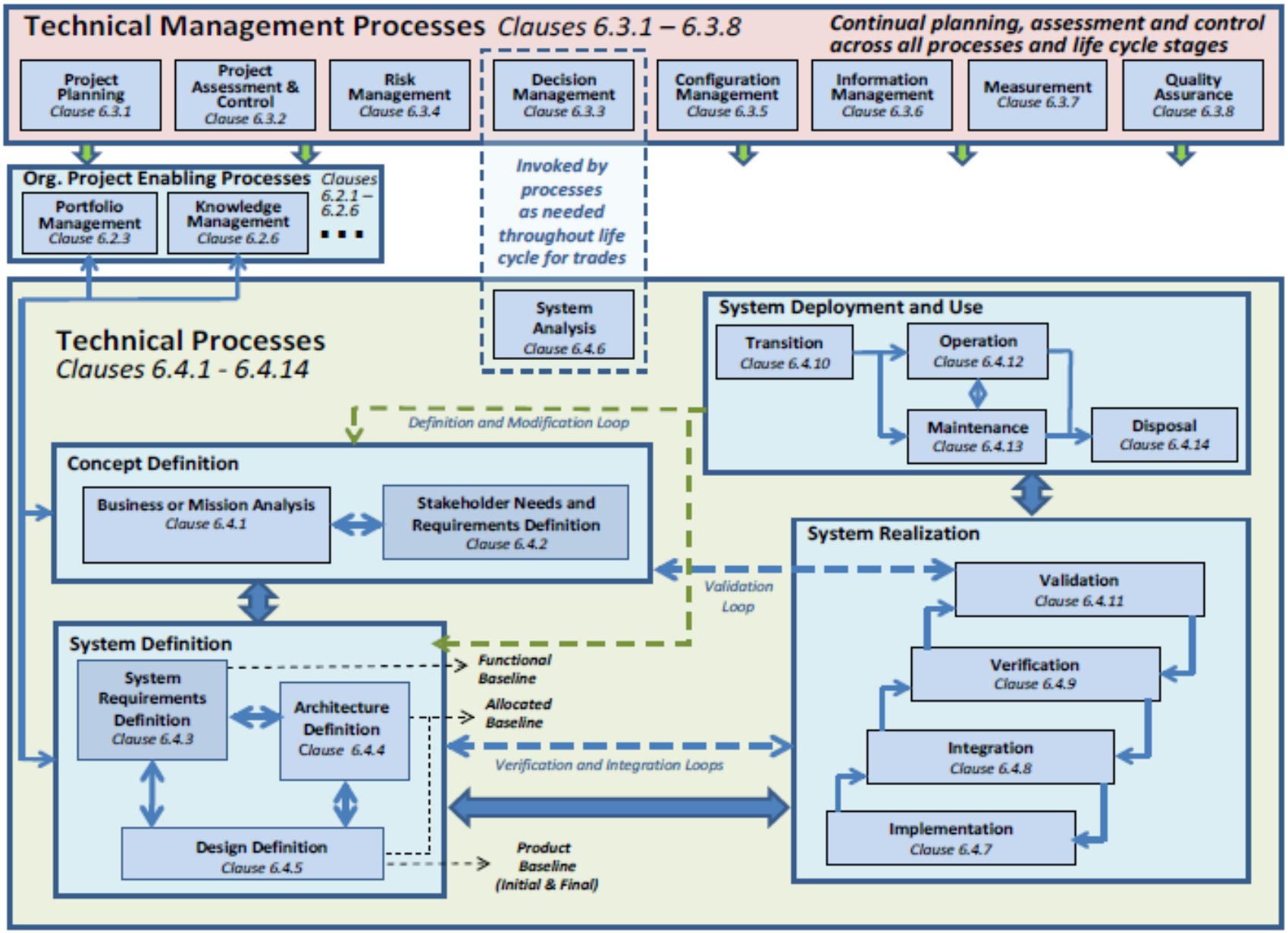
- Экспертная оценка архитектуры
- Архитектура для клиента
  - Обсуждение архитектуры с представителями заказчика и другими интересантами
  - Архитектура представляется в адекватных для них представлениях, отражающих их (различные!) интересы

ISO 15288 – ПРАКТИКИ ЖИЗНЕННОГО ЦИКЛА СИСТЕМНОЙ ИНЖЕНЕРИИ

Общие понятия

- ISO/IEC/IEEE 15288:2015 – Systems and software engineering – System life cycle processes – описывает процессы и этапы ЖЦ [любой системы]
  - Разработка стандарта началась, когда осознали необходимость обобщения Systems Engineering process framework
  - Ранее применялся стандарт MIL STD 499A от 1969 года

Взаимосвязь процессов



Особенность стандарта

- Для каждого процесса определены:
  - Назначение
  - Результаты
  - Операции / деятельность при реализации (activities)
- Стандарт обобщает 25 процессов, 123 результата 403 операций

ISO 42010 – АРХИТЕКТУРНОЕ ОПИСАНИЕ

Общие понятия

- ISO/IEC/IEEE 42010:2011 – Systems and software engineering — Architecture description – определяет требования к описанию системной, программной и enterprise архитектур
  - Стандартизует практики описания архитектур, задавая термины, представляя концептуальные основы отображения архитектуры (expressing), ее обсуждения и reviewing
  - Задает требования, применимые к описаниям архитектур, архитектурных frameworks и языков описания архитектур
- NB! Различаются архитектура и описание архитектур
- ГОСТ Р 57100-2016 – перевод этого стандарта

Определения (ГОСТ Р 57100-2016) – 1/3

- Процесс архитектуризации (architecting): Процесс понимания, определения, выражения, документирования, взаимодействия, соответствующей сертификации при реализации, сопровождении и улучшении архитектуры в ЖЦ системы
  - В РИС будем называть «проектирования высокого уровня» или «разработка архитектуры», ~~иначе кровь из глаз и ушей пойдет~~
  - Процесс идет в конкретной организации и/или проекте
- Архитектура (системы) (architecture): Основные понятия или свойства системы в окружающей среде, воплощенной в ее элементах, отношениях и конкретных принципах ее проекта и развития
  - Ср. определение РИС
- Описание архитектуры (architecture description): Рабочий продукт, используемый для выражения архитектуры
- Структура архитектуры (architecture framework): Условности, принципы и практики для описания архитектур, установленные в пределах заданной области применения и/или объединения заинтересованных сторон

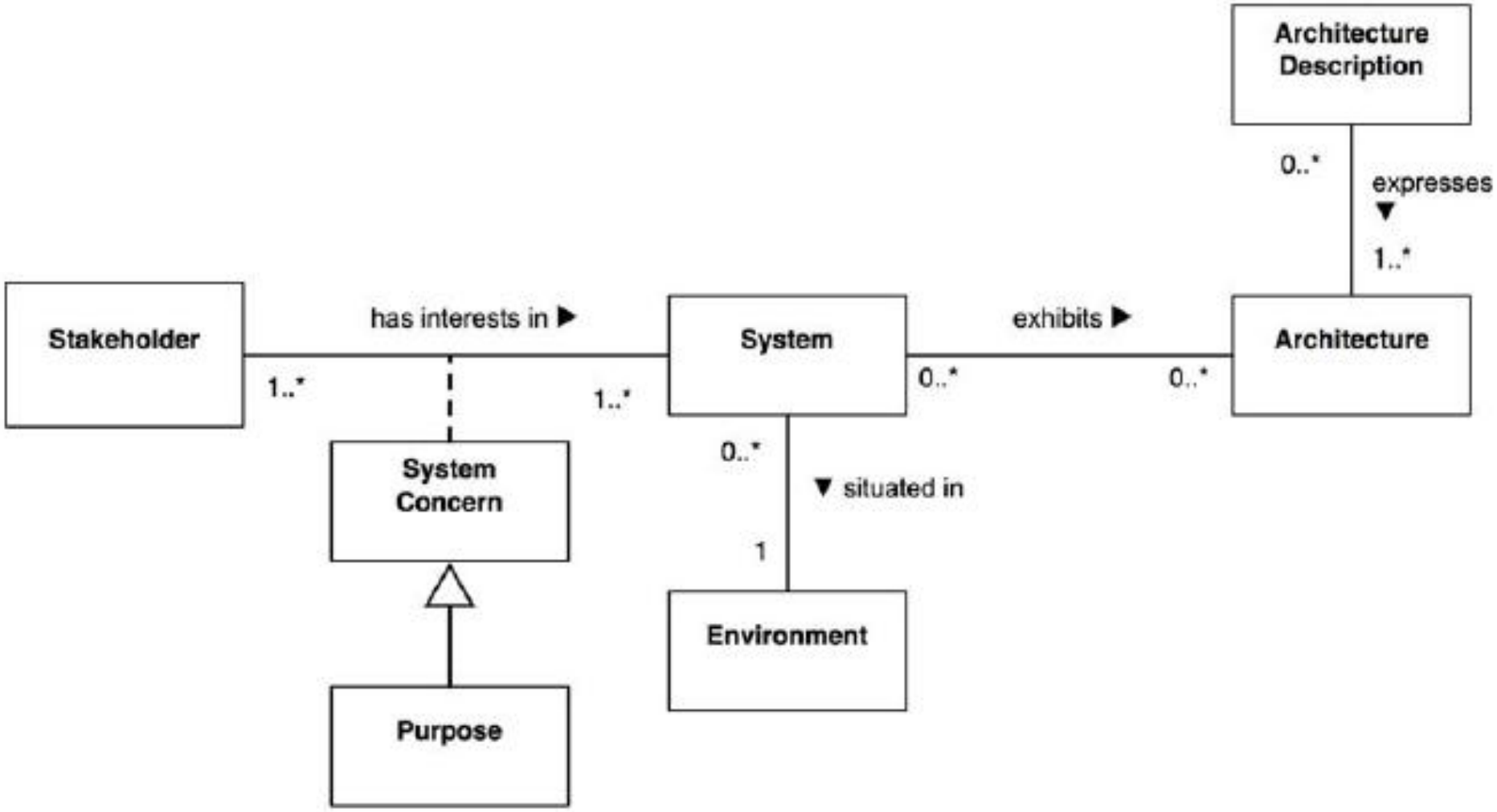
Определения – 2/2

- Заинтересованная сторона, ~~правообладатель~~ (stakeholder): Индивидуум, команда, организация или их группы, имеющие интерес в системе
- Интерес (системы) (concern): Польза или проблемы в системе, относящиеся к одной или нескольким заинтересованным сторонам
  - Относится к любому воздействию на систему в ее окружающей среде, включая воздействия разработки, технологические, деловые, эксплуатационные, организационные, политические, экономические, юридические, регулирующие, экологические и социальные воздействия
- Окружающая среда (системы) (environment): Контекст, определяющий параметры и обстоятельства всех воздействий на систему
  - Включает воздействия разработки, технологические, деловые, эксплуатационные, организационные, политические, экономические, юридические, регулирующие, экологические и социальные воздействия
  - В РИС используем термин «операционное окружение»

Определения – 3/3

- Архитектурное представление (architecture view): Рабочий продукт, выражающий архитектуру некоторой системы с точки зрения определенных системных интересов
- Точка зрения на архитектуру (architecture viewpoint): Рабочий продукт, устанавливающий условности конструирования, интерпретации и использования архитектурного представления для структуризации определенных системных интересов
- Вид модели (model kind): Условности для типа моделирования
  - В РИС: способ представления некоторых свойств системы
  - Например: диаграммы потока данных, диаграммы классов, сети Петри, бухгалтерские балансы, организационные структуры, диаграмма состояний, etc.
- Примечание: Рабочий продукт может представлять собой документ, раздел документа, онтологию, программный компонент – нечто, позволяющее читателю / пользователю воспринять информацию

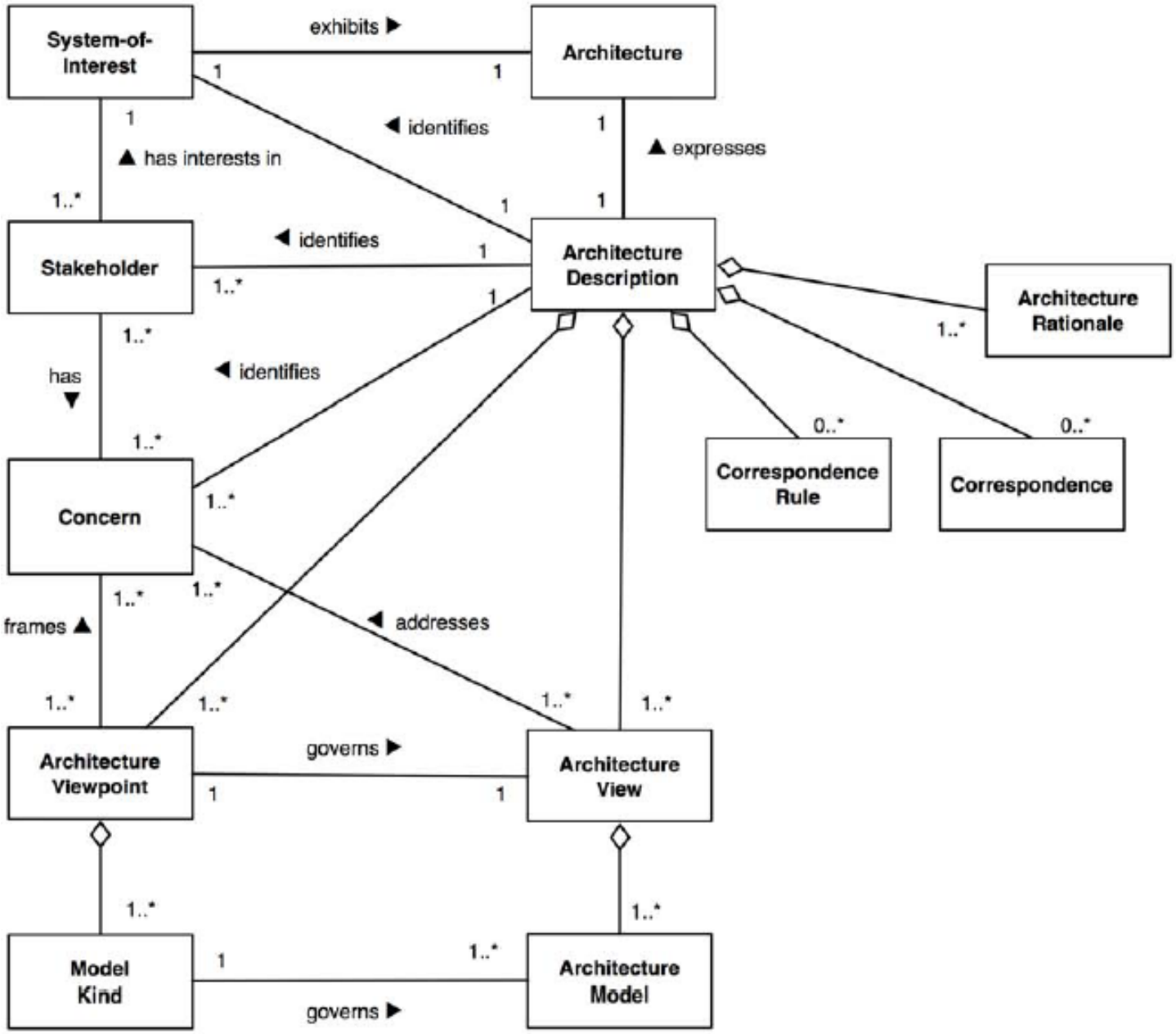
Связь архитектуры, системы и СХ



Описание архитектуры

- Описывает (expresses) архитектуру
- Идентифицирует целевую систему (system of interest)
- Идентифицирует 1 или более СХ
- Обозначает (identifies) 1 или более интересов СХ
- Включает 1 или более пар «точка зрения на (сторон рассмотрения) архитектуру (viewpoints)» – «архитектурное представление (views)»
- Включает 0 или более отношений
- Включает 0 или более правил отношений
- Включает 1 или более обоснований архитектуры

Состав архитектуры



Architecture viewpoint

- Часть архитектурного описания
- Покрывает (frames) 1 или более интересов СХ, касающихся целевой системы
- Задает (governs) точно одно архитектурное представление, ему соответствующее
- Состоит из 1 или более типов моделей

Architecture viewpoint: состав – 1/4

- Включает несколько подразделов. Некоторые могут быть опущены:
  - Имя
    - Включает также синонимы
  - Общее описание (overview)
    - Введение или краткое описание, включая ключевые особенности такой точки рассмотрения
  - Concerns and «anti-concerns»
    - Список того, что конкретно интересует или, наоборот, совсем не интересует заинтересованные стороны
      - В том аспекте (стороне рассмотрения) системы, которая будет описана в данном разделе
    - Помогает создателю архитектуры принять и обосновать решение о необходимости включения данной viewpoint в архитектурный документ системы
    - «Anti-concerns» полезны для исключения из рассмотрения ненужных моделей, неподходящих для данной системы

Architecture viewpoint: состав – 2/4

- Типичные СХ (список СХ, которым интересны views из данного viewpoint)
  - Примечание: Разработчик архитектуры должен описать какие concerns относятся к каждому СХ
- Типы моделей (Model kinds). Описывается:
  - Идентификация типа моделей
  - Описание языков, нотации, соглашений, методик моделирования (conventions, language or modelling techniques)
  - Описываются ресурсы, которые будут доступны в Viewpoint и словари построения view. Также описываются аналитические методы и/или другие операции, которые будут использоваться на моделях этого вида

Описание моделей

Описание типа моделей не стандартизовано.

Можно описать, например, с помощью:

1. Метамодели

   могут быть описаны с использованием метамодели для вида модели, который определяет структуру и соглашения для ее моделей

2. Шаблона модели для заполнения пользователями
3. Языком описаний или ссылкой на существующий язык описаний моделей
4. Комбинацией способов 1…3

Architecture viewpoint: состав – 3/4

- Зависимости (Correspondence rules)
  - Описывает зависимости, описанные в данном viewpoint или в использованных типах моделей. Обычно такие зависимости будут «меж – модельными» или «меж – view» так как ограничения типа моделей описаны как часть описания типа моделей
- Операции с views
  - Методы, применяемые к представлениям или их моделям. Операции можно разделить на категории:
    - **Методы создания** могут описываться в виде:
      - Процедурных руководств по основным шагам реализации (process guidance): как начать, что делать дальше
      - Руководство по содержанию (work product guidance): шаблоны для представлений этого типа
      - Эвристика, стилей, шаблоны или другими способами
    - **Методы интерпретации** задают способы обеспечить облечение понимания представлений читателем (CX)
    - **Методы анализа** используются для проверки, анализа, преобразования, прогнозирования, применения и оценки архитектурных решений из этого представления
    - **Методы проектирования или реализации** используются для создания (конструирования) системы, используя архитектурные решения из этого представления

Architecture viewpoint: состав – 4/4

- Примеры
  - Данный раздел содержит примеры, то есть любую дополнительную информацию, полезную читателям (CX)
- Источники
  - Описывает источники информации данного viewpoint, если они есть, включая авторство, историю, ссылки на литературу, предшествующий уровень техники (решений)

Architecture view

- Часть описания архитектуры
- Соответствует точке зрения на архитектуру (1:1)
- Отражает один или более интересов, заданных СХ
- Состоит из 1 или более архитектурных моделей типов, заданных точкой зрения на архитектуру

Блок-схема

HIPO-диаграмма

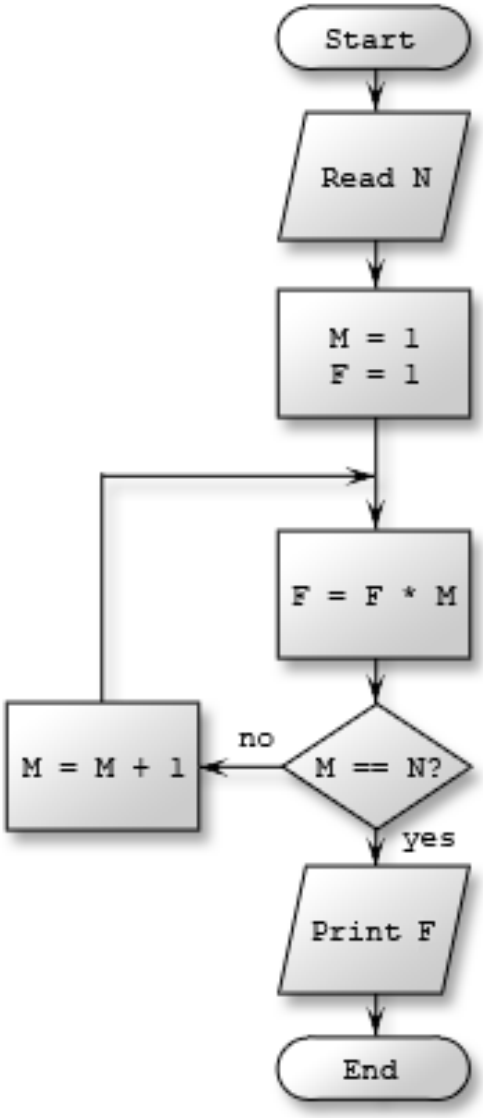
ERM-диаграмма

DFD

Псевдо-ЯП

НОТАЦИИ

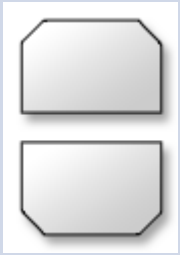
Нотация: блок-схема – 1/4



Нотация: блок-схема – 2/4

Блок	Описание
	Начало-конец (пуск-остановка)
	Блок действия Набор последовательно и неразрывно выполняемых операций
	Логический блок (условие) Условие выбора дальнейшего пути (может делить на 2 и более путей)
	Процедурный блок
	Ввод-вывод данных

Нотация: блок-схема – 3/4

Блок	Описание
	Начало и конец цикла
	Соединитель Разрывы линий на одном листе Переходы на другие листы
	Комментарий

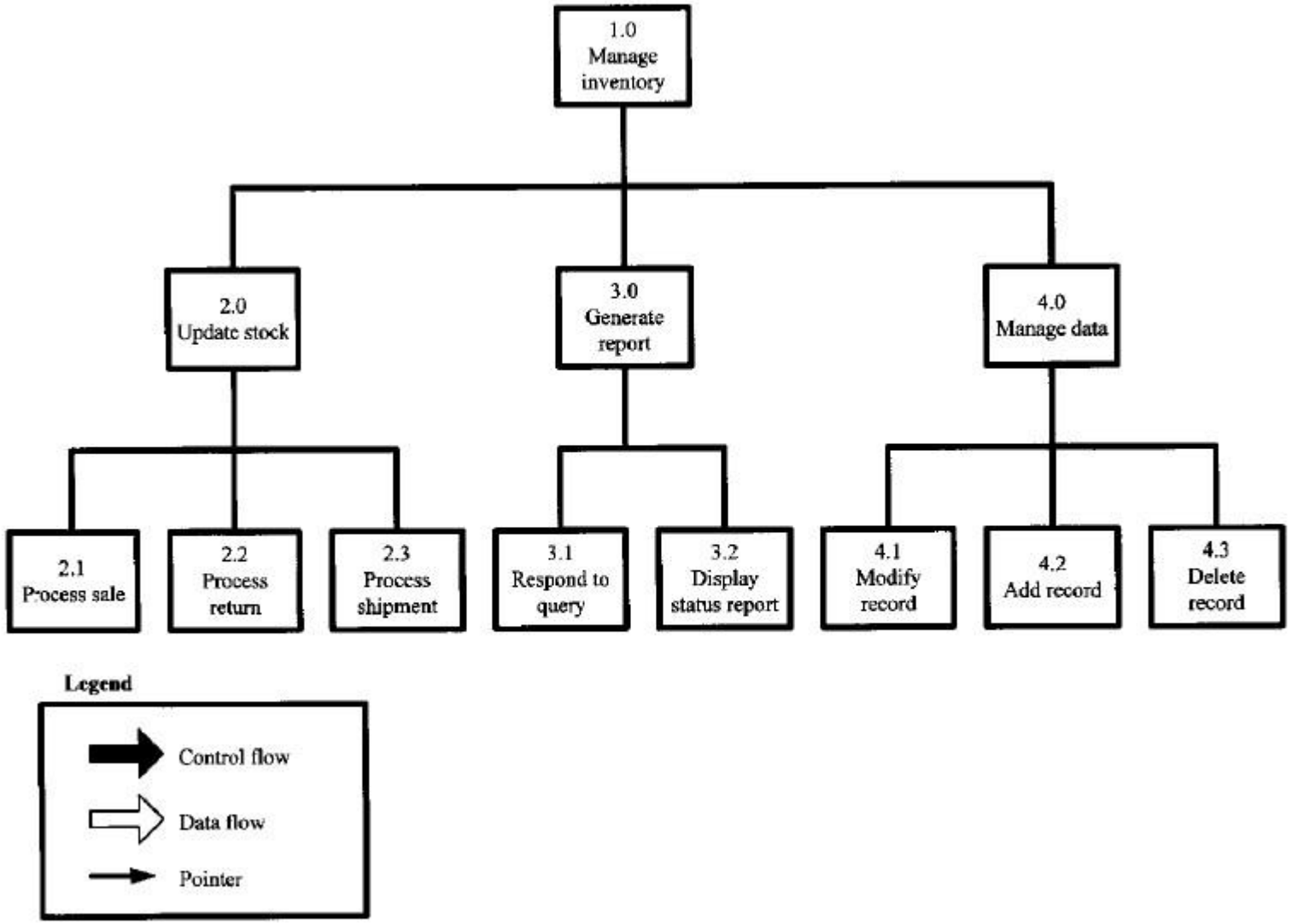
Нотация: блок-схема – 4/4

- Преимущества:
  - Распространенность
  - Простота понимания
- Недостатки:
  - Сложные алгоритмы выглядят сложно
  - Сложные отношения между элементами ПО сложно отобразить
  - Применимо только к нижнему уровню

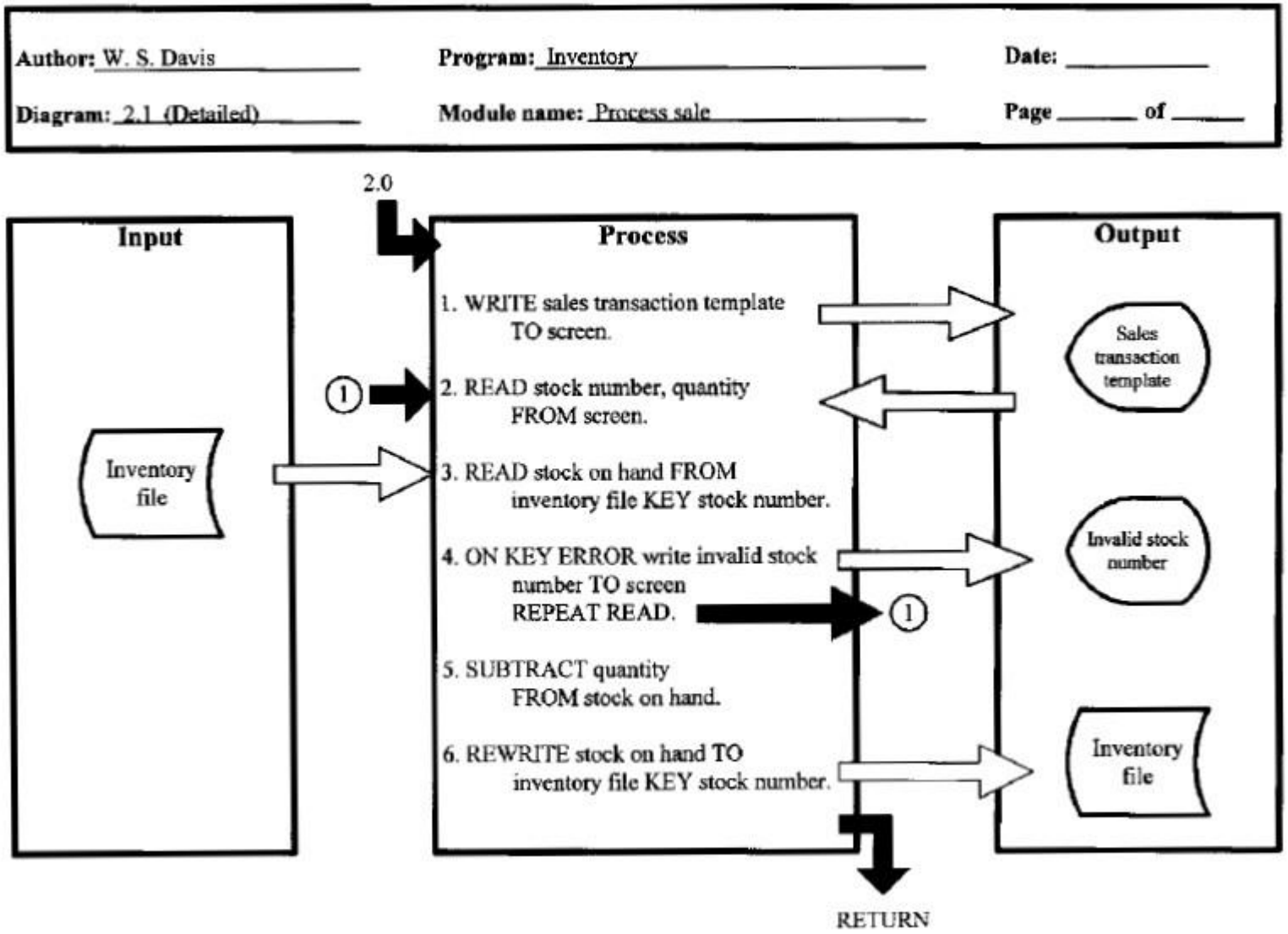
Нотация: HIPO – 1/3

- Hierarchical input process output (HIPO) — технология проектирования и документирования. Состоит из:
  - Диаграммы иерархии
  - IPO – диаграммы процессов с описанием входной и выходной информации
- Недостатки:
  - Алгоритмы тяжело представить
  - Современные программы не всегда иерархичны
- Преимущества:
  - Описывает: иерархию, IO, передачи управления
  - Ностальгия!

Нотация: HIPO – 2/3



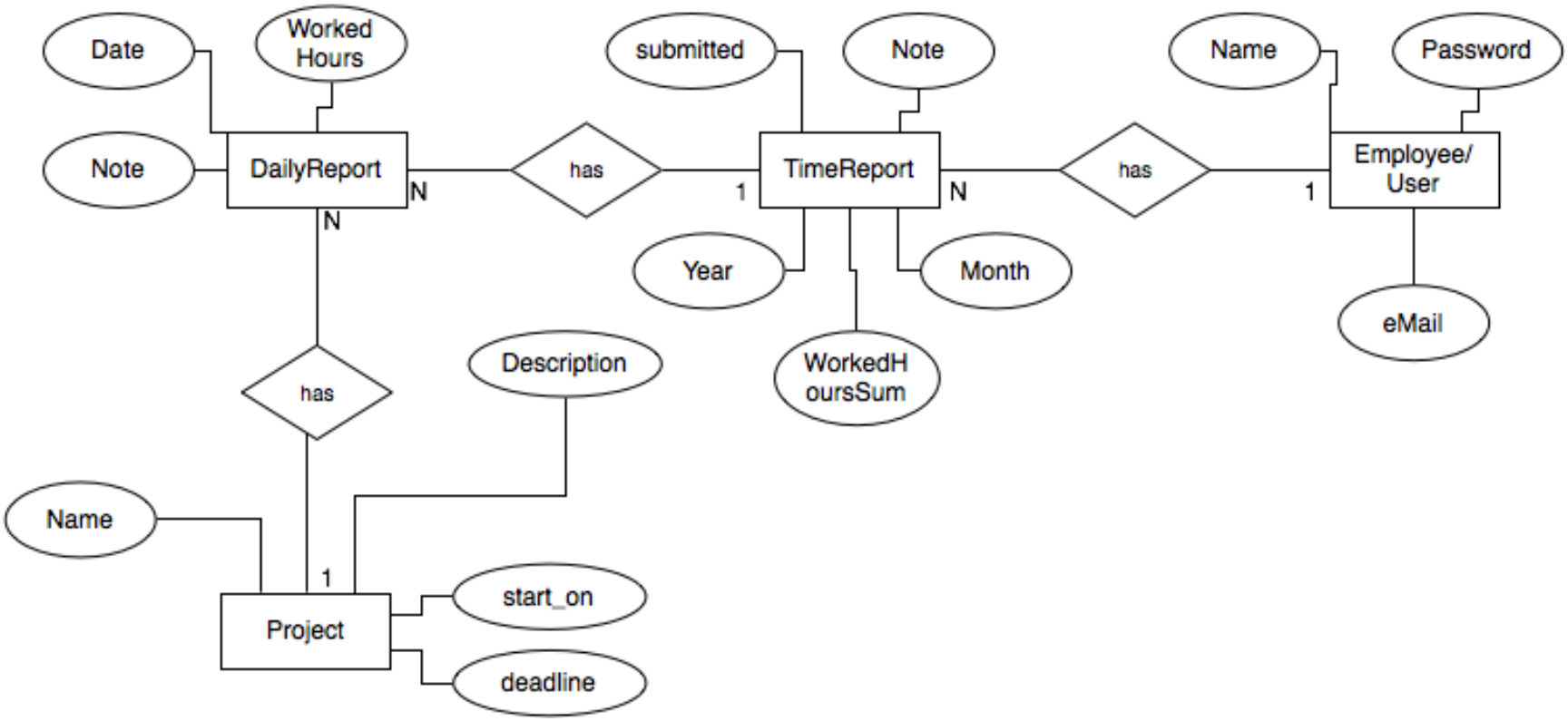
Нотация: HIPO – 3/3



Нотация: ER-диаграмма – 1/2

- Модель сущность-связь (ER-модель – entity-relationship) — модель данных, позволяющая описывать концептуальные схемы предметной области
- Была создана для проектирования БД
- Нотация Питера Чена:
  - Сущности – прямоугольники
  - Отношения – ромбы
  - Линия – отношение между сущностями, если отношение не является обязательным, то линия пунктирная
  - Овалы – атрибуты отношения или сущности

Нотация: ER-диаграмма – 2/2



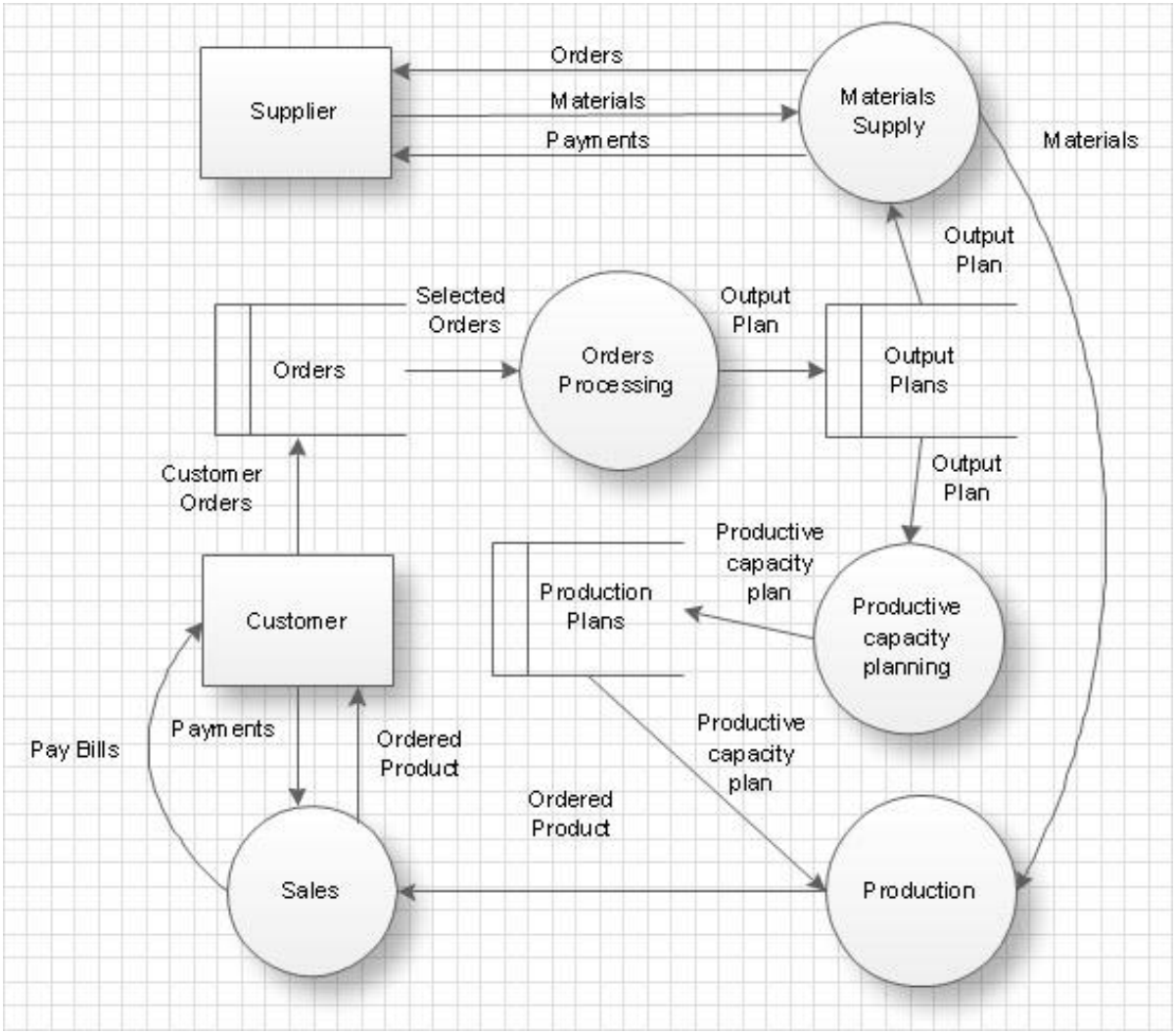
Нотация: Data Flow диаграмма – 1/3

- DFD (Data Flow Diagrams) — диаграммы потоков данных
- Преимущества:
  - Иерархическая
  - Верхний уровень иерархии описывает внешние интерфейсы системы
  - Поддерживает понятие подсистемы
- Недостатки:
  - Сложное поведение (не обработку потока данных) представить затруднительно

Нотация: Data Flow диаграмма – 2/3

- Элементы:
  - Внешняя сущность (External Entity)
  - Системы и подсистемы (Process)
    - Например: подразделение, программа
  - Поток данных

Нотация: Data Flow диаграмма – 3/3



Нотация: Псевдо ЯП – 1/2

- Алгоритм описывается на ЕЯ с использованием для структурирования элементов ЯП
  - Например: PDL (Program Design Language)
- Преимущества:
  - Понятно разработчикам
  - Компактнее и выразительнее чем блок-схема
  - Легко конвертируется в код:
    - ЯП остаются как есть, ЕЯ куски становятся комментариями
- Недостатки:
  - Непонятно не-программистам
  - Применимо только на нижнем уровне

Нотация: Псевдо ЯП – 2/2

```
Boolean купитьКолбасы()
{
    if( яйца есть? )
    then
        купить( 10 )
    else
        купить( 1 )
    endif
    return TRUE
}
```

ПОДХОД ЗАХМАНА

Zachman Framework – 1/4

- Модель описания архитектуры системы, предоставляющая формальный структурированный способ описания. Представляет собой матрицу:
  - 6 столбцов – «стороны» представления системы
  - 6 строк – уровни представления системы
- Обычно при описании используют в качестве примера проектирование здания

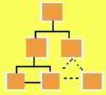
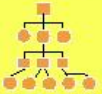
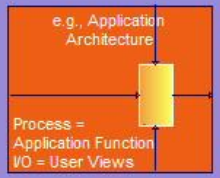
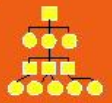
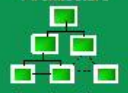
Zachman Framework: Views – 2/4

1. Planner's View (Scope)
   - Представление инвестора или планирующего деятельность для получения общего описания и оценки стоимости и связи с внешним миром
2. Owner's View (Enterprise or Business Model)
   - Представление владельца, использующего систему в повседневной деятельности
3. Designer's View (Information Systems Model)
   - Представление разработчика архитектуры, переводящего требования в архитектурные решения
4. Builder's View (Technology Model)
   - Описание конкретных способов реализации архитектуры
5. Subcontractor View (Detailed Specifications)
   - Подробные описания, достаточные для реализации и/или покупки и настройки конкретных модулей
6. Actual System View
   - Сама реализованная система

Zachman Framework: «стороны» – 3/4

- What
  - Описание данных
- How
  - Описание функций
- Where
  - Описание сети (развертывания)
- Who
  - Описание персонала / пользователей
- When
  - Описание временных зависимостей
- Why
  - Цели и задачи

Zachman Framework – 4/4

	WHAT	HOW	WHERE	WHO	WHEN	WHY
	DATA	**FUNCTION**	**NETWORK**	**PEOPLE**	**TIME**	**MOTIVATION**
SCOPE {contextual} Planner	List of things important to the business  Entity = Class of business things	List of processes the business performs  Process = Class of business process	List of locations in which the business operates  Node = Major business locations	List of organisations important to the business  People = Major business unit	List of event cycles signifcant to the business  Time = Major Business Event Cycle	List of business goals/ strategies  End/Means = Major Business Goal/Strategy
BUSINESS MODEL {Conceptual} Owner	e.g., Semantic Model  Entity = Business Entity Relationship = Business	e.g.,Business Process Model  Process = Business I/O = Business Resource	e.g.,Business Logistics System  Node = Business Location Link = Business Linkage	e.g.,Workflow Model  People = Organisation unit Work = Work Product	e.g.,Master Schedule  Time = Business Event Cycle = Business Cycle	Business Plan  End = Business Objective Means = Business Strategy
SYSTEM MODEL {Logical} Designer	e.g., Logical Data Model  Entity = Data Entity Relationship = Data Relationship	e.g., Application Architecture  Process = Application Function I/O = User Views	e.g., Distributed System Model  Node = I/S Function Relationship = Line Characteristics	e.g., Human Interface Architecture  People = Role Work = Deliverable	e.g., Processing Structure  Time = System Event Cycle = Processing Cycle	e.g., Business Rule Model  End = Structural Assertion Means = Action Assertion
TECHNOLOGY MODEL {Physical} Builder	e.g., Physical Data Model  Entity = Segment/Table Relationship = Pointer/key	e.g., System Design  Process = Computer Function I/O = Data Elements/sets	e.g., Technology Architecture  Node = H/w /System s/w Relationship = Line Specifications	e.g., Presentation Architecture  People = User Work = Screen Formats	e.g., Control Structure  Time = Execute Cycle = Component Cycle	e.g., Rule Design  End = Condition Means = Action
DETAILED REPRESENTATIONS {Out-of-context} Subcontractor	e.g., Data Definition  Entity = Field Relationship = Address	e.g., Program  Process = Language Statement I/O = Control Block	e.g., Network Architecture  Node = Address Link = Protocol	e.g., Security Architecture  People = Identity Work = Job	e.g., Timing Definition  Time = Interrupt Cycle = Machine Cycle	e.g., Rule Specification  End = Sub-condition Means = step
FUNCTIONING ENTERPRISE	e.g DATA	e.g FUNCTION	e.g NETWORK	e.g ORGANISATION	e.g SCHEDULE	e.g STATEGY