Архитектура программного обеспечения инфокоммуникационных систем

Раздел 1

Архитектура и дизайн программного обеспечения

Понятие архитектуры ПО

Архитектура программной системы:

- структура системы, включающая программные элементы, видимые извне свойства этих элементов и взаимоотношения между ними
- базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы
- представляет собой набор существенных решений, касающихся организации программной системы, таких как:
  - выбор структурных элементов и их интерфейсов, из которых слагается система;
  - поведение, определяемое взаимодействием этих элементов;
  - объединение этих структурных и функциональных элементов в более крупные подсистемы;
  - архитектурный стиль, исповедуемый в данной организации

Исходные вопросы при разработке архитектуры ПО

- Как пользователь будет использовать приложение?
- Как приложение будет развертываться и обслуживаться при эксплуатации?
- Какие требования по атрибутам качества, таким как безопасность, производительность, возможность параллельной обработки, интернационализация и конфигурация, выдвигаются к приложению?
- Как спроектировать приложение, чтобы оно оставалось гибким и удобным в обслуживании в течение долгого времени?
- Основные архитектурные направления, которые могут оказывать влияние на приложение сейчас или после его развертывания?

Основы проектирования архитектуры ПО

Архитектура сосредотачивается на следующих конкретных аспектах:

- Структура модели - организационные паттерны, например, деление на уровни (layering)
- Наиболее важные элементы - критические сценарии использования, основные классы, общие механизмы и т.П., Но не все элементы, существующие в модели
- Несколько ключевых сценариев, показывающих основные потоки управления в системе
- Службы, модульность, необязательная функциональность, аспекты, связанные с линейкой продуктов

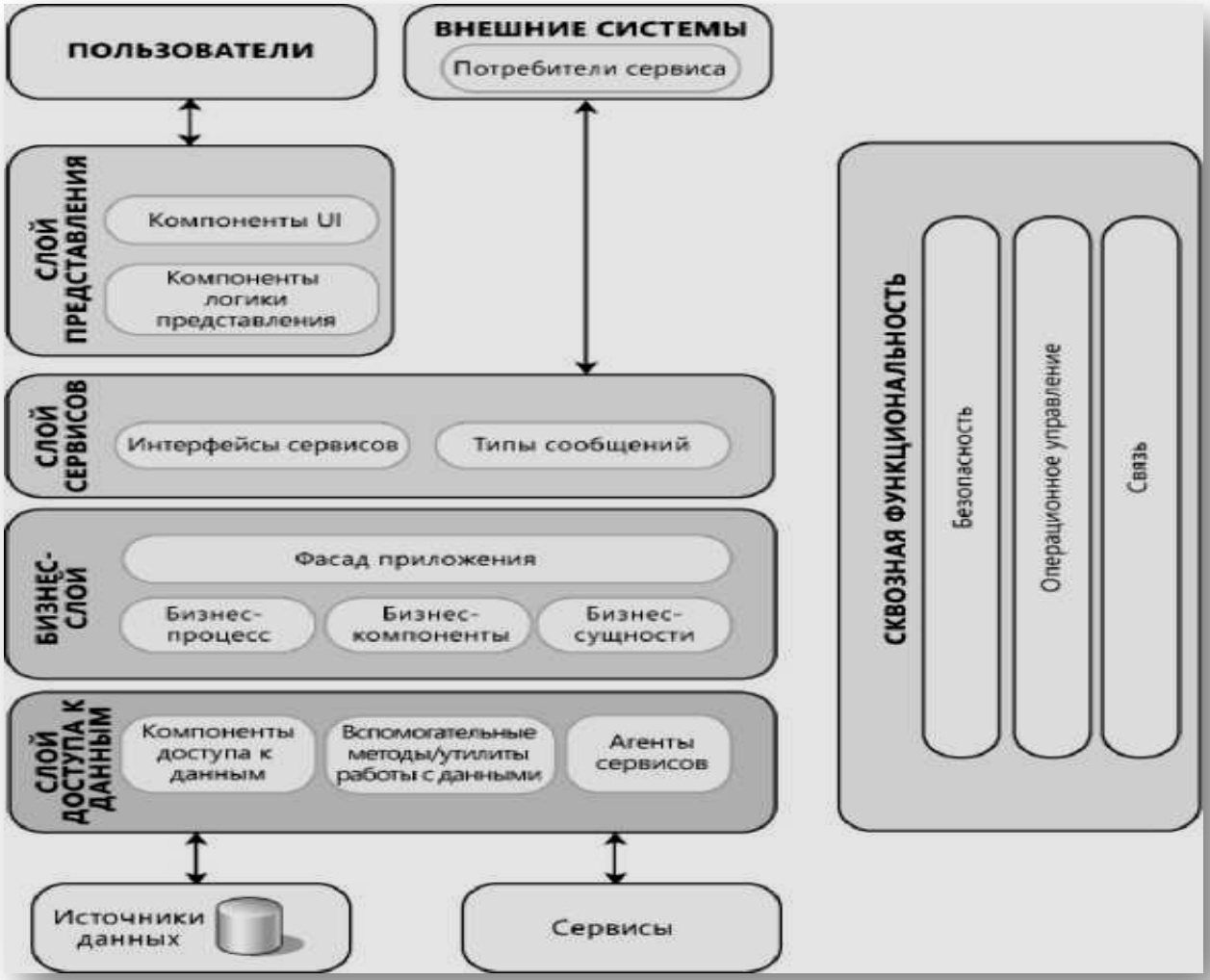
Назначение и цель архитектуры

- Основное назначение: описание использования или взаимодействия основных элементов и компонентов приложения с целью обеспечения определенной функциональности.
- Архитектура приложения должна
  - объединять бизнес-требования и технические требования через понимание вариантов использования с последующим нахождением путей их реализации в ПО.
  - раскрывать структуру системы, но скрывать детали реализации;
  - реализовывать все варианты использования и сценарии;
  - по возможности отвечать всем требованиям различных заинтересованных сторон;
  - выполнять требования по функциональности и по качеству

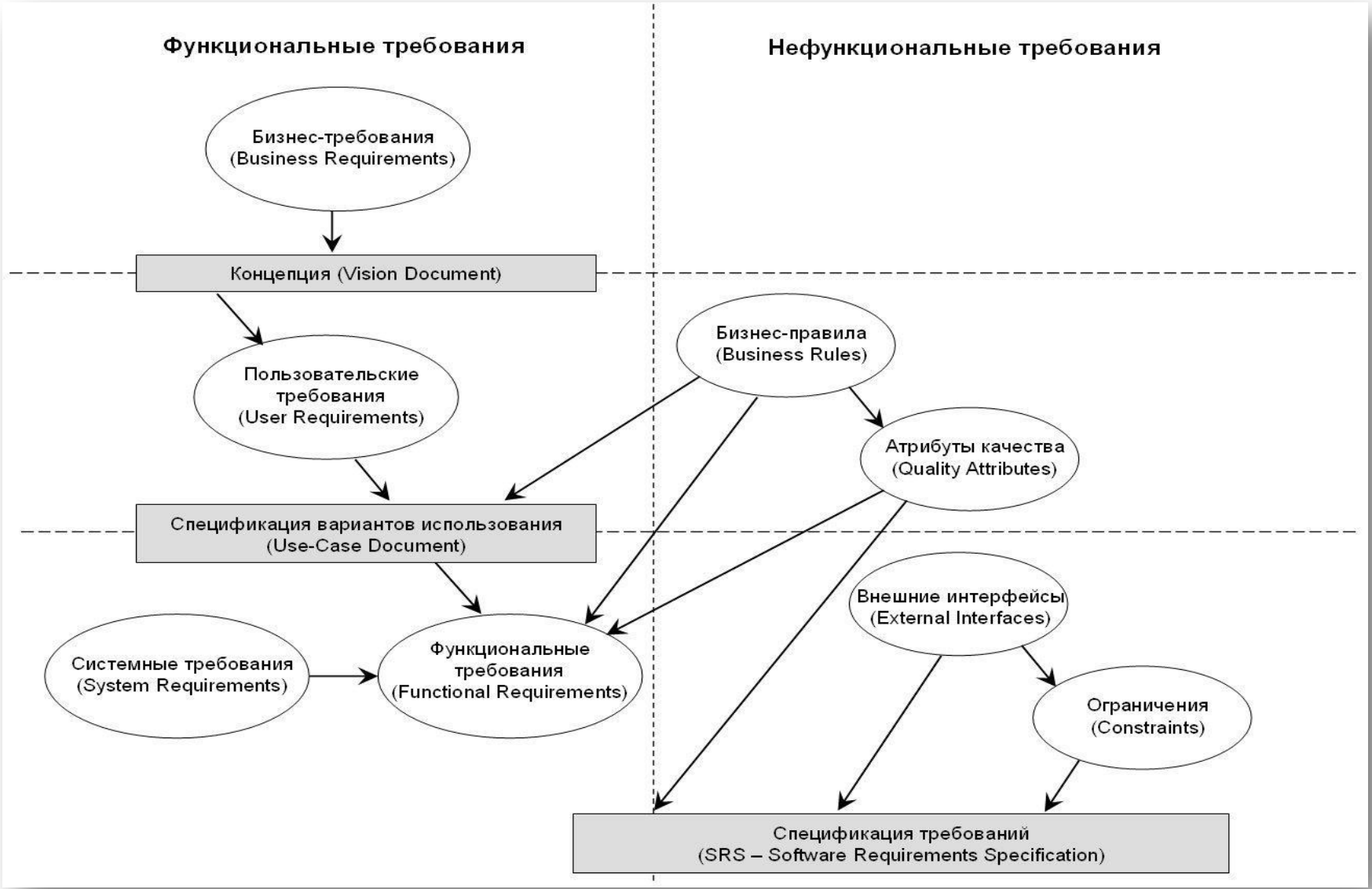
Назначение и цель архитектуры

- Цель архитектуры – выявить требования, оказывающие влияние на структуру приложения
- Хорошая архитектура:
  - снижает бизнес-риски, связанные с созданием технического решения;
  - обладает значительной гибкостью, чтобы справляться с естественным развитием технологий, как в области оборудования и ПО, так и пользовательских сценариев и требований

Типовая архитектура приложения



Формирование архитектурных решений



Отличия архитектуры от дизайна

- Архитектура приложения – фундамент, на который опираются все остальные компоненты системы, в том числе элементы дизайна
- Характерная черта архитектуры:
  - ее трудно изменить на поздних стадиях разработки

Рекомендации при проектировании архитектуры

- Создавайте, чтобы изменять
- Создавайте модели для анализа и сокращения рисков
- Используйте модели и визуализации как средства общения при совместной работе
- Выявляйте ключевые инженерные решения
- Не пытайтесь создать слишком сложную архитектуру и не делайте предположений, которые не можете проверить
- Рассмотрите возможность использования инкрементного и итеративного подхода при работе над архитектурой

Основные принципы проектирования

- Разделение функций
  - Разделите приложение на отдельные компоненты по возможности, минимальным перекрытием функциональности
- Принцип единственности ответственности
  - Каждый отдельно взятый компонент должен отвечать только за одно конкретное свойство/функцию или совокупность связанных функций
- Принцип минимального знания (Закон Деметера (Law of Demeter, LoD))
  - Компоненту или объекту не должны быть известны внутренние детали других компонентов или объектов
- Не повторяйтесь (Don't repeat yourself, DRY)
- Минимизируйте проектирование наперед
  - Проектируйте только то, что необходимо

Рекомендации по проектированию слоев приложения

- Разделяйте функциональные области
- Явно определяйте связи между слоями
  - Принимайте явные решения о зависимостях между слоями и о потоках данных между ними
- Реализуйте слабое связывание слоев с помощью абстракции
- Не смешивайте разные типы компонентов на одном логическом уровне
  - Начинайте с идентификации функциональных областей и затем группируйте компоненты, ассоциированные с каждой из этих областей в логические уровни
- Придерживайтесь единого формата данных в рамках слоя или компонента

Рекомендации для компонентов, модулей и функций

- Компонент или объект не должен полагаться на внутренние данные других компонентов или объектов
  - Это способствует созданию более удобных в обслуживании и адаптируемых приложений
- Не перегружайте компонент функциональностью
- Разберитесь с тем, как будет осуществляться связь между компонентами
  - Это требует понимания сценариев развертывания, которые должно поддерживать создаваемое приложение
- Максимально изолируйте сквозную функциональность от бизнес-логики приложения
- Определяйте четкий контракт для компонентов

Проектирование приложений

Основные принимаемые решения:

- Определение типа приложения
  - Данный выбор определяется конкретными требованиями и ограничениями среды
- Выбор стратегии развертывания
  - При выборе стратегии необходимо найти компромисс между требованиями приложения и соответствующими схемами развертывания, поддерживаемым оборудованием, и ограничениями, налагаемыми
- Выбор соответствующих технологий
  - Ключевым фактором является тип разрабатываемого приложения, а также предпочтительные варианты топологии развертывания приложения и архитектурные стили
- Выбор показателей качества
- Решение о путях реализации сквозной функциональности
  - Сквозная функциональность представляет ключевую область дизайна, не связанную с конкретным функционалом приложения

Архитектурные шаблоны и стили

- Архитектурный стиль (шаблон, высокоуровневая схема) – это набор принципов, обеспечивающая абстрактную инфраструктуру для семейства систем.
- Архитектурный стиль определяет набор компонентов и соединений, которые могут использоваться в экземплярах этого стиля, а также ряд ограничений по их возможным сочетаниям

Категория	Архитектурные стили
Связь	Сервисно-ориентированная архитектура (SOA), шина сообщений
Развертывание	Клиент/сервер, N-уровневая, 3-уровневая
Предметная область	Дизайн на основе предметной области (Domain Driven Design)
Структура	Компонентная, объектно-ориентированная, многоуровневая архитектура