Техническая документация

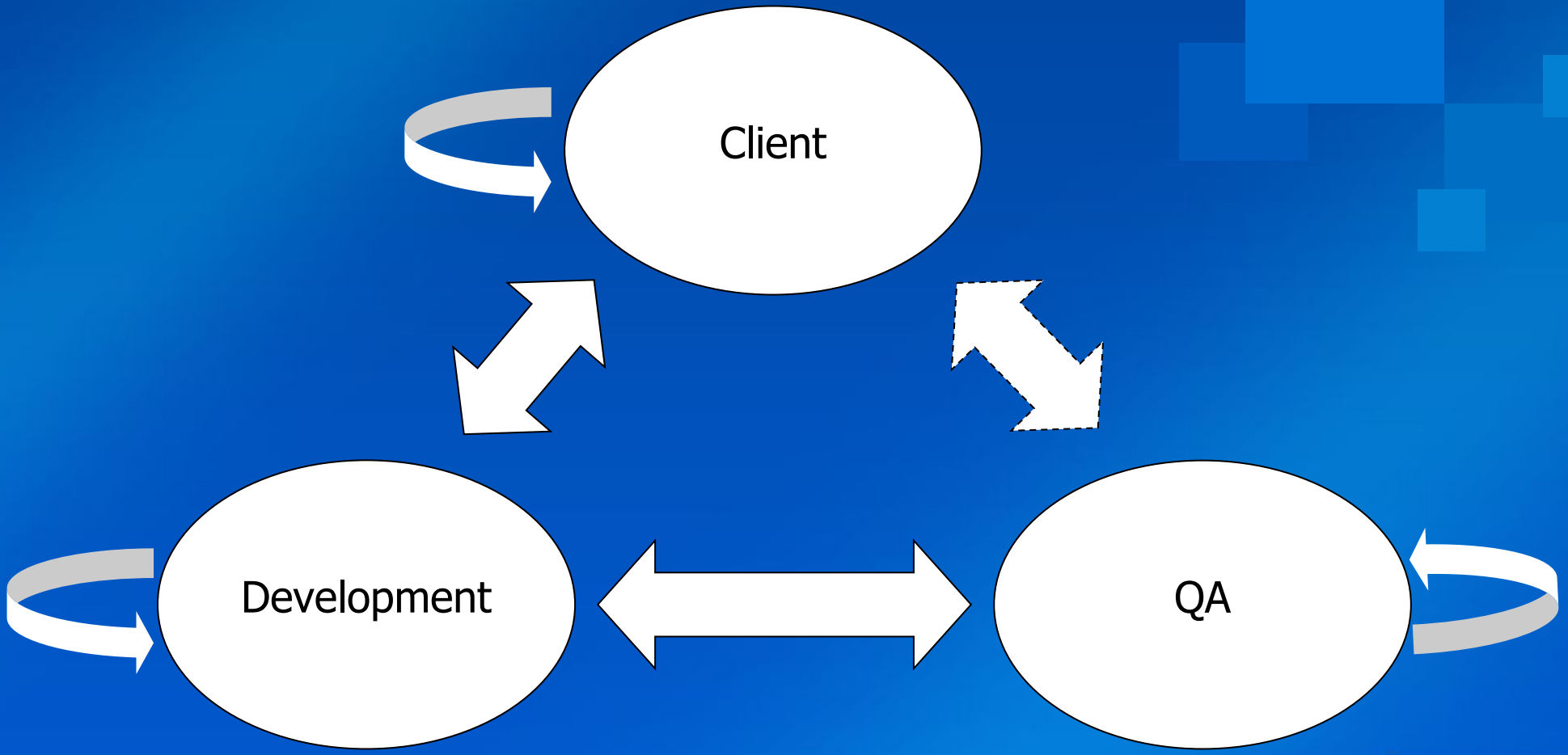
Иван Позняк

Internal Training

Введение

- Документация – набор документов, используемых при проектировании, создании и эксплуатации программного обеспечения

Как используется



Классификация

- Техническая документация
  - Project Vision
  - SAO (System Architecture Overview)
  - SDD (System Design Document)
  - API documentation
  - Deployment/Integration/Installation Guide
- Функциональная документация
  - SRS (System Requirements Specification)
- QA документация
  - Test cases

«За» и «Против»

- Плюсы
  - Единый источник знаний на проекте
    - Облегчает передачу знаний
    - Специфицирует как должен работать конечный продукт
  - Позволяет задуматься о проблемах до их появления
- Минусы
  - Документацию нужно писать
  - Документацию нужно поддерживать в актуальном состоянии

Написание

- Нам нужны
  - Правильный человек
  - Правильный шаблон
  - Время на написание документа
  - Время на ревью документа
- Используем полученную документацию
- По необходимости обновляем
- Шаги 3 и 4 – могут повторяться несколько раз до получения результата нужного качества

Шаблоны RUP

- http://www.ts.mah.se/RUP/RationalUnifiedProcess/wordtmpl/index.htm
- Software Architecture Document
- Примеры

Место документации в разработке проекта

Идея продукта

- Цель
  - Выбрать компанию которая реализует проект
- Форма
  - Документ либо иным образом оформленное видение будущей системы
- Наша реакция
  - Создание RFX

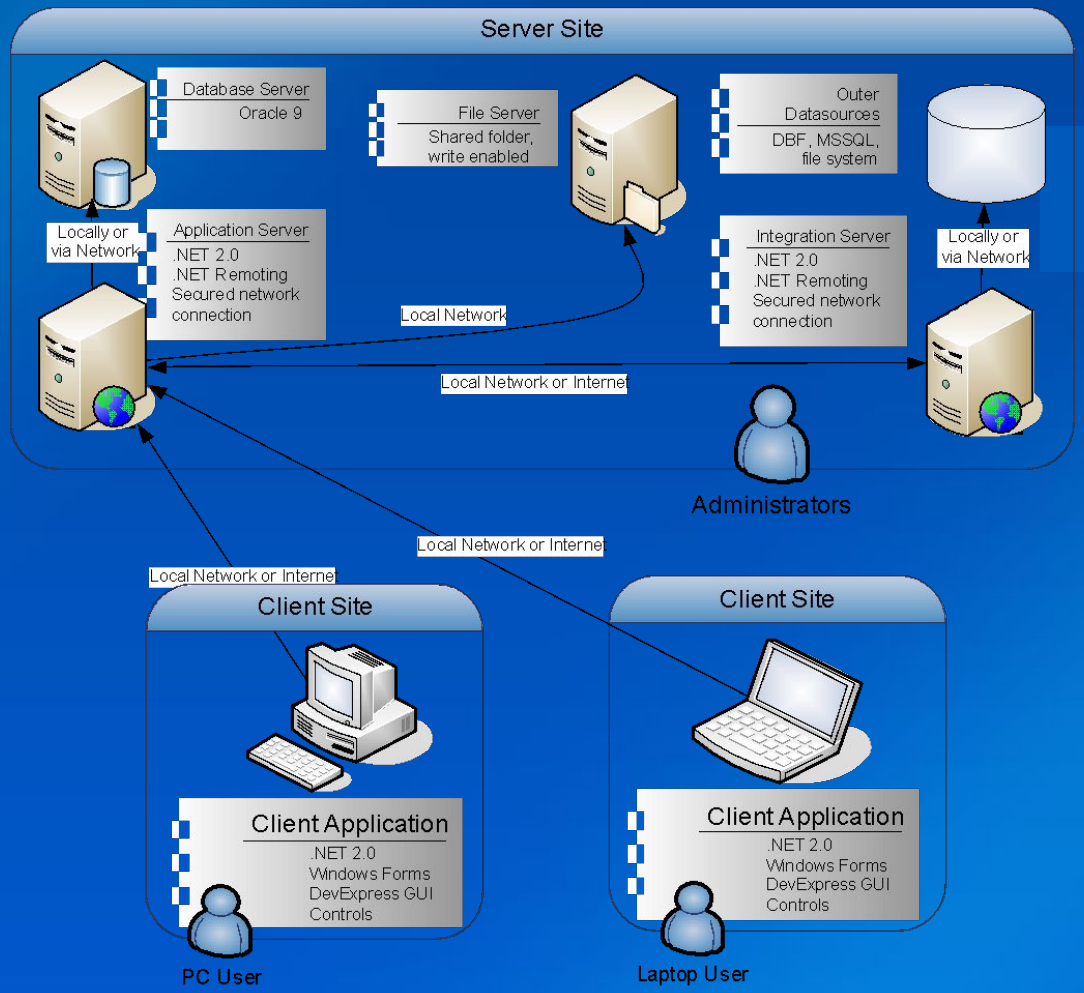
Обработка RFX

- Цель
  - Получить контракт на разработку приложения
- Форма
  - Proposal
    - Technical Vision
    - Functional Vision
    - Cost $
- Технический специалист
- Бизнес аналитик

Technical Vision

- Ориентация - заказчик
- Что должно быть
  - Обзор системы
  - Топология системы
  - Используемая платформа/технология
  - Ответственности отдельных частей
- Чего писать не стоит:
  - Специфических технических вещей *

Пример схемы для Technical Vision



Фаза 0 / Фаза анализа

- Цель
  - Собрать требования по проекту
- Форма
  - SRS
- Бизнес аналитик

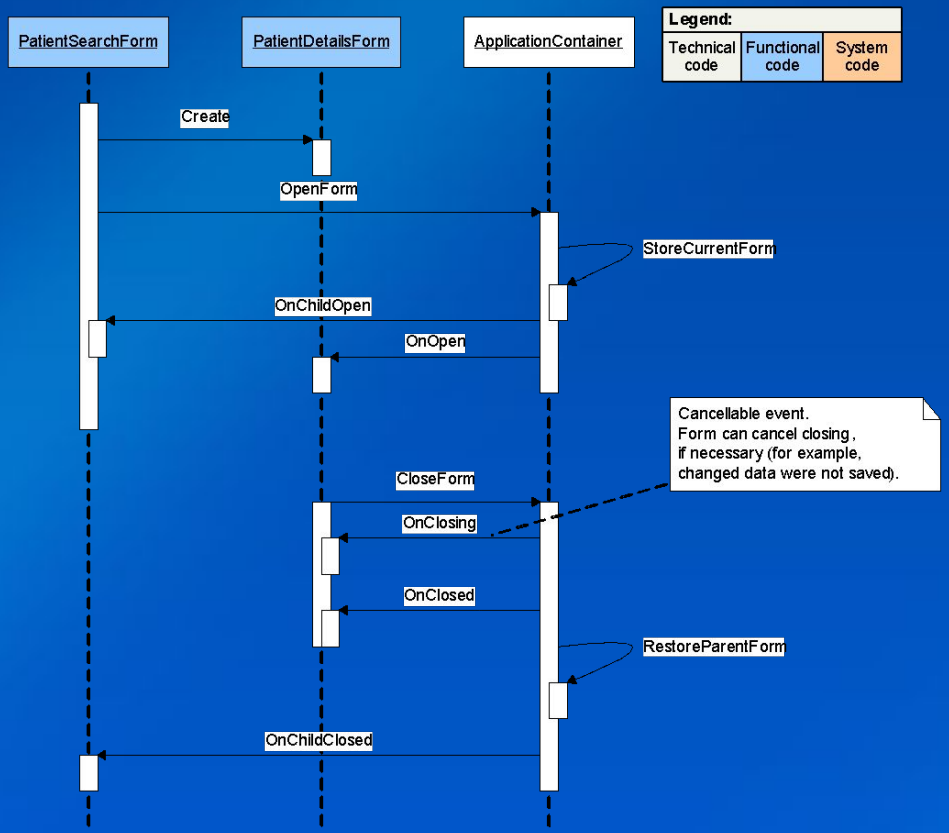
Начало разработки *

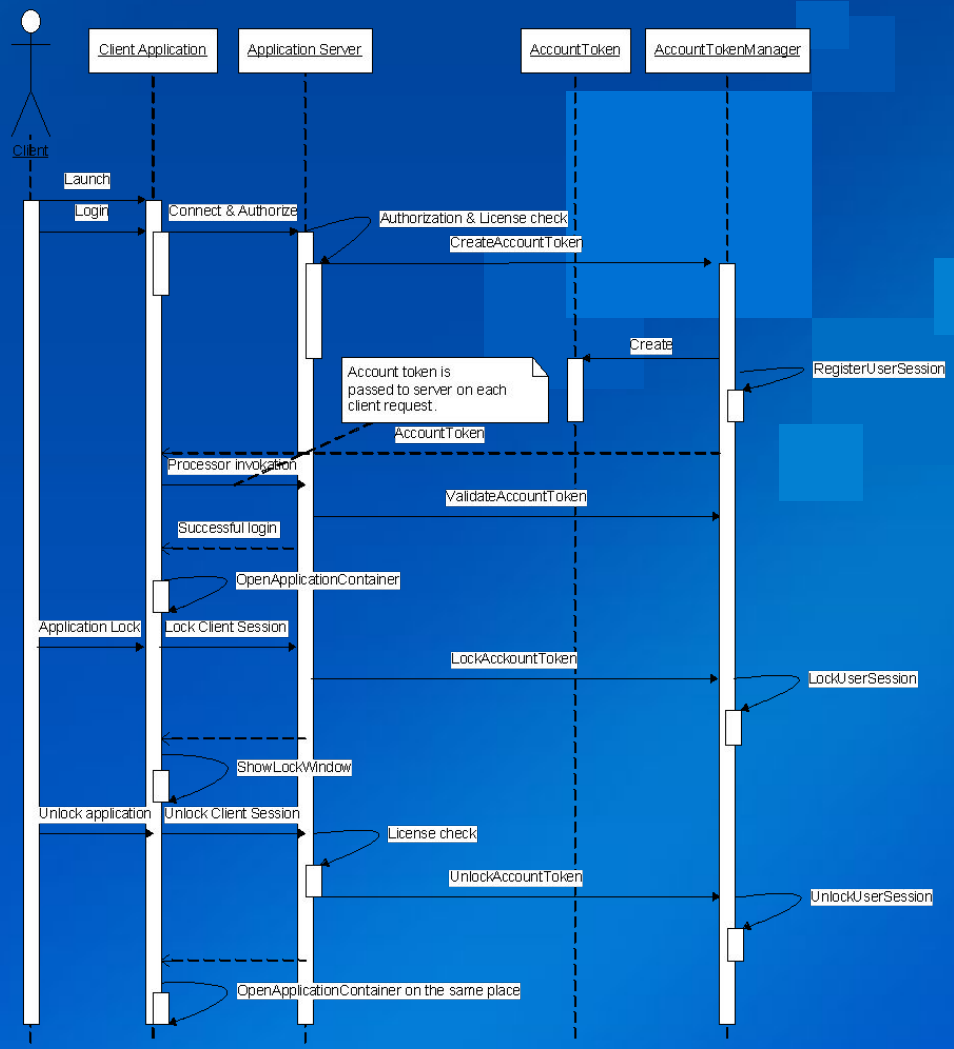
- Цель
  - Разработать архитектуру проекта, получить базу для его дальнейшего развития
- Форма
  - SAO / SDD
  - Code
- Технический специалист

System Architecture Overview

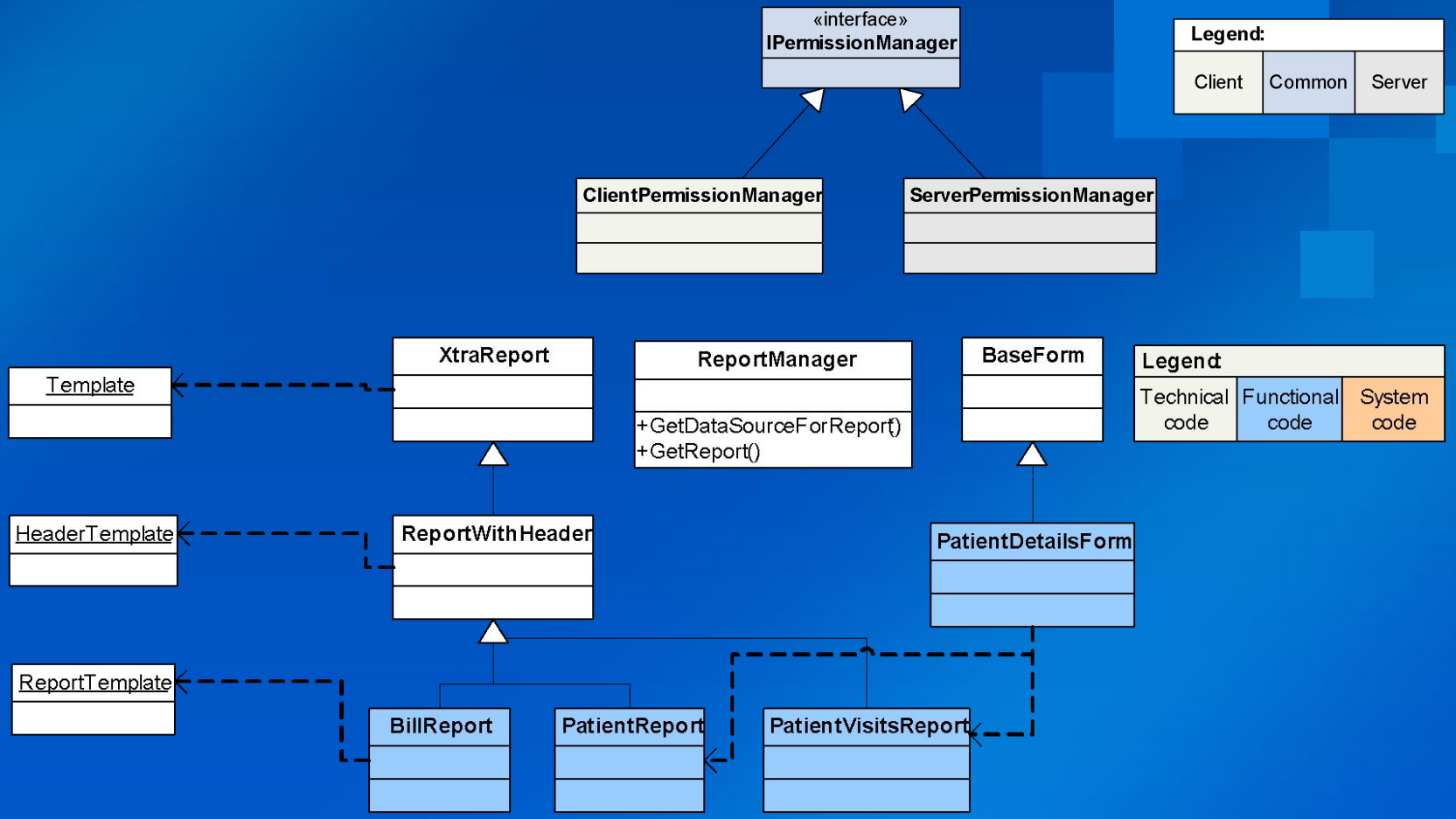
- Ориентация – тех. специалисты, разработчики
- Один из часто пишущихся документов
- Чаще всего пишется до начала либо в начале процесса разработки
- Что должно быть
  - Общее описание архитектуры
  - Описание различных фич/механизмов
  - Различные UML диаграммы

Диаграммы последовательностей





Диаграммы классов



System Design Document

- Ориентация – тех специалисты, разработчики
- Отличается от SAO более детальным и глубоким рассмотрением отдельных частей системы
- Что должно быть
  - Схемы
  - Различные UML диаграммы
  - Возможно примеры/отрывки кода

Что писать в документации

- Введение
- Обзор системы
- Описание дизайна
- Архитектурные решения
- Архитектура системы
- Политики и соглашения
- Детальное описание системы

Введение

- Назначение документа
- Целевая аудитория для данного документа
- Ссылки на другие документы, prerequirements
- Важные понятия и определения
- Соглашения по обозначениям в документе
- Краткое содержимое всего документа

Обзор системы

- Здесь дается общий обзор разрабатываемого продукта, включая следующие пункты:
  - Общая функциональность
  - Базовые архитектурные решения
    - Например 3-tier architecture
  - Назначение и ответственность отдельных частей приложения

Описание дизайна: предположения

- Используемое/требуемое ПО/железо
- Операционная система
- Требования к пользователям системы
- Возможность дальнейшего изменения системы

Описание дизайна: ограничения

- Требования по софту и железу
- Доступность различных ресурсов (Internet, printer и т.п.)
- Совместимость со стандартами
- Требования по предоставляемым интерфейсам/используемым протоколам
- Требования к хранилищам данных (объемы данных и пр.)
- Требования к системе безопасности
- Ограничения по используемой памяти и т.п.
- Требования по производительности
- Требования к качеству

Описание дизайна: принципы

- Описываются основные принципы, которыми следует руководствоваться при разработке и проектировании архитектуры будущего приложения
  - KISS (“keep it simple stupid!”)
  - Упор на скорость/производительность в ущерб использованию памяти
  - Внешний вид/принцип работы как у существующего приложения
  - …
- Желательно пояснение по каждому принципу

Архитектурные решения

- Выбор языка/платформы/библиотеки, повторное использование готовых компонент
- Возможности по расширению системы
- Обнаружение, обработка ошибок и восстановление после них
- Фреймворк для работы с базой данных/внешним хранилищем данных
- Механизмы синхронизации и параллельной обработки
- Механизмы коммуникации
- Использование внешних ресурсов

Политики и соглашения

- Описываются те решения, которые не сильно влияют на высокоуровневую организацию, однако влияют на детали реализации тех или иных механизмов
- Гайдлайны и соглашения
- Внутренний протокол общения между отдельными модулями
- Выбор отдельного алгоритма / паттерна программирования
- Интерфейсы предоставляемые внешним системам
- Организация кода по физической структуре каталогов и файлов

Описание архитектуры системы

- Дается общее представление об архитектурном устройстве приложения
- Декомпозиция на компоненты
- Взаимодействие, роли и обязанности отдельных компонент

- Здесь же можно дать высокоуровневые описания различных подсистем/компонент

Детальное описание системы

- Компонент (модуль, класс, библиотека, ...)
- Назначение, роли и обязанности
- Правила взаимодействия, ограничения
- Ресурсы, которые используются/управляются этим компонентом
- Сервисы/интерфейсы предоставляемые компонентом

Общие рекомендации

- DRY (Don’t Repeat Yourself)
  - Вещи, которые легко читаются из кода, можно оставить без внимания, сделав соответствующую ссылку
  - Следует избегать дублирования, выносить общие вещи в отдельные секции/документы и ставить ссылки
- Желательно использовать больше диаграмм (классов, последовательностей) для демонстрации того как работает и устроена система

[здесь могла бы быть реклама тренинга по UML]

Фаза приёмки ПО

- Происходит валидация разработанного ПО на соответствие исходным требованиям и исправление найденных багов
- Активно участвуют практически все QA, заказчик, бизнес-аналитик и разработчики

Различные гайды

- Ориентация – широкие массы
- Что должно быть
  - Простое понятное изложение
  - Пошаговые инструкции
  - Проработка различных ситуаций/вариантов

Фаза эксплуатации ПО

- Деньги заплатили, все довольны ☺
- Для поддержки (фикса багов в гарантийный период) а также для дальнейшего развития функционала приложения может потребоваться API документация

API Documentation

- Ориентация – разработчик
- Не забывать писать понятные комментарии
- Различные утилиты для генерации
  - NDoc
  - Sandcastle
  - Javadoc
  - Rdoc

Спасибо за внимание