Технология разработки программного обеспечения

Этапы процесса разработки

Этапы процесса разработки ПО

Жизненный цикл ПО - это последовательность этапов, которые проходит программа в ходе своего существования

Классическими считаются:

- Анализ
- Проектирование
- Разработка
- Тестирование
- Развертывание
- Использование

Этапы процесса разработки ПО

Это цикл не единственный , его можно расширить или сузить

- Бизнес-моделирование
- Анализ требований
- Планирование
- Разработка архитектуры
- Кодирование
- Тестирование и отладка
- Документирование
- Сертификация
- Внедрение
- Сопровождение

Распределение времени в проекте

Существует правило по которому распределяется время работы в проекте 30-40-30

- 30% - анализ и проектирование
- 40% - реализация проекта
- 30% - отладка и тестирование

30-20-10 хорошая команда

30-60-90 плохая команда

Анализ проекта, списки требований

- Разработка требований
  - Выявление требований
  - Анализ требований
  - Спецификация требований
  - Проверка требований
- Управление требованиями
  - После составления списка требований его необходимо от ранжировать
  - Реализация первоочередных требований (базовые)
  - Выявление несовместимых требований
  - Требования, которыми можно пожертвовать в кризисной ситуации (90% этим пользоваться не будет)

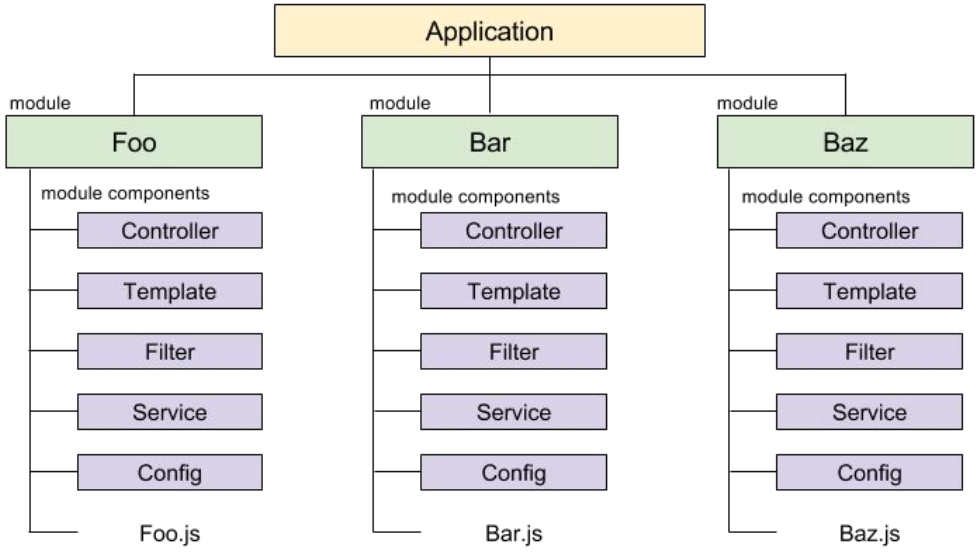
Анализ проекта, списки требований

Виды требований

- Функционал системы
  - Что должна делать система? (на общем уровне)
  - Как она должна это делать
  - В какой среде
- Качество реализации
  - Производительность
  - Время бесперебойной работы
  - Варианты обработки внештатных ситуаций
  - Качество тестов
  - Состав и объем документации (Техническое описание, базовые алгоритмы, видео уроки, сервис помощи, руководство по развертыванию и т.д.)
- Сроки реализации
- Стоимость разработки

Проектирование

- Разбиение системы на отдельные модули
- Определение функциональности модулей



Сверху вниз

Плюсы: разработка именно необходимой нам системы

Минусы: увеличенная стоимость с учетом выполнения полного цикла работ и создания уникальных технологий

Снизу вверх

Плюсы: стоимость разработки ниже за счет использования готовых модулей

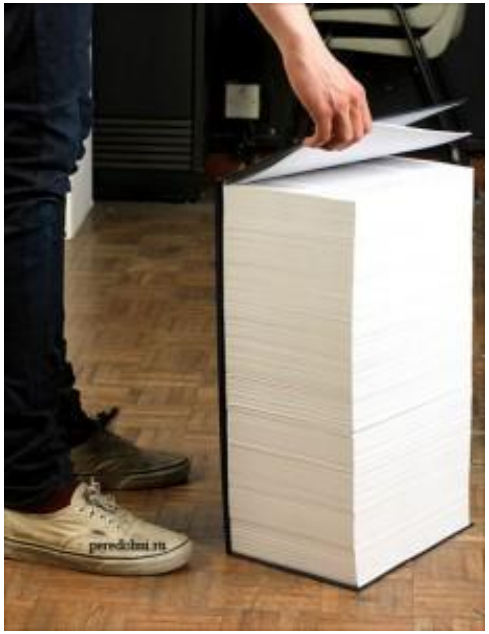
Минусы: возможны отклонения от начальной постановки задачи

Списки требований

Преимущества

- Обеспечивает контрольный список требований.
- Обеспечивает договор между заказчиками и разработчиками.
- Для большой системы может обеспечить описание высокого уровня.

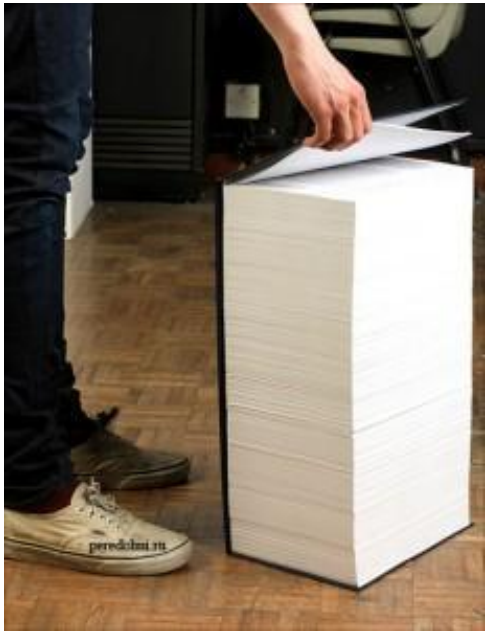
Списки требований



Недостатки

- Такие списки могут занимать сотни страниц. Фактически невозможно прочитать такие документы в целом и получить чёткое понимание системы.
- Такие списки требований перечисляют отдельные требования абстрактно, оторваны друг от друга и от контекста использования (Абстракция лишает возможности видеть, как требования связываются между собой или работают вместе).

Списки требований



- Такие списки могут занимать сотни страниц. Фактически невозможно прочитать такие документы в целом и получить чёткое понимание системы.
- Такие списки требований перечисляют отдельные требования абстрактно, оторваны друг от друга и от контекста использования (Абстракция лишает возможности видеть, как требования связываются между собой или работают вместе).
- Эти списки создают ложное чувство взаимопонимания между заинтересованными лицами и разработчиками.
- Эти списки дают заинтересованным лицам ложное чувство защищённости, что разработчики должны достигнуть определённых вещей.

Какая же есть альтернатива?

Пользовательские истории (User stories)

Что это такое?

Пользовательские истории (англ. *User Story*) — способ описания требований к разрабатываемой системе, сформулированных как одно или более предложений на ежедневном или деловом языке пользователя.

Служат не для документации требований, а скорее, напоминанием заказчику о наличии таковых для дальнейших обсуждений продукта с командой

Почему же обсуждение требований так важно?

- Заказчик не может учесть всех аспектов продукта самостоятельно, так как требования должны «лечь» на технологии. Только совместные усилия заказчика и команды в поиске решений и компромиссов дают оптимальные результаты
- Детализация требований посредством обсуждений дает возможность всем участникам понять суть требований и, так образом, обзавестись базой для принятия оптимальных решений
- Изначально упуская некоторые аспекты историй и оставляя тем самым место для дискуссий, заказчик расширяет область поиска решений. Это приводит к расширению кругозора участников проекта и, как следствие, к предпосылкам нахождения более приемлемых решений;
- Свобода поиска решений является отличным мотивирующим механизмом, что делает повседневную работу команды более интересной.

Почему же обсуждение требований так важно?

- видение (vision) системы
- пользовательские роли

Actor, роль – одна из обобщенных пользовательских ролей;

действие – действие, выполняемое пользователем посредством взаимодействия с системой;

цель, ценность – конечная цель текущей задачи, выполняемой пользователем посредством взаимодействия с системой.

Пример

Видение(Vision)

У нас (заказчиков) есть потребность в реализации системы, которая бы позволила пользователям хранить и обмениваться фотографиями. Ожидается, что прибыль от системы будет достигаться за счет процента с продаж пользователями своих фотографий, также, возможно, за счет рекламы третьих компаний.

Итак, истории:

1. Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать или продать их другим пользователям.
2. Как рекламодатель я могу помещать свою рекламу в системе, ориентированную на пользователей.
3. Как администратор я могу управлять фотографиями пользователей, так чтобы контент сайта был легальным.

Во время обсуждения первой истории

1. Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать или продать их другим пользователям.
2. Как рекламодатель я могу помещать свою рекламу в системе, ориентированную на пользователей.
3. Как администратор я могу управлять фотографиями пользователей, так чтобы контент сайта был легальным.
4. Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.
5. Как гость я могу войти в систему под ранее созданной учетной записью, для последующей работы.

Пользуясь принципом симметричности требований

1. Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать или продать их другим пользователям.
2. Как рекламодатель я могу помещать свою рекламу в системе, ориентированную на пользователей.
3. Как администратор я могу управлять фотографиями пользователей, так чтобы контент сайта был легальным.
4. Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.
5. Как гость я могу войти в систему под ранее созданной учетной записью, для последующей работы.
6. Как пользователь я могу удалить свою учетную запись и перестать быть пользователем системы.

Обсуждая концепцию учетных записей, рождаются также следующие истории

1. Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать или продать их другим пользователям.
2. Как рекламодатель я могу помещать свою рекламу в системе, ориентированную на пользователей.
3. Как администратор я могу управлять фотографиями пользователей, так чтобы контент сайта был легальным.
4. Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.
5. Как гость я могу войти в систему под ранее созданной учетной записью, для последующей работы.
6. Как пользователь я могу удалить свою учетную запись и перестать быть пользователем системы.
7. Как пользователь я могу изменить данные своей учетной записи.
8. Как пользователь я могу сделать некоторые поля своей учетной записи видимыми для других пользователей.

Куда подевались детали?

Как без понимания деталей программист может написать адекватный код, а тестировщик его принять?

Детали необходимы.

Куда подевались детали?

Как без понимания деталей программист может написать адекватный код, а тестировщик его принять?

Детали необходимы.

Детали историй – это больше не неизменная часть требований, которые продумываются заказчиками во время написания требований и предъявляются команде в готовом виде. Вместо этого заказчик и команда во время обсуждений историй совместно приходят к понимаю уровня детализации, который необходим на текущей фазе, и принимают совместные решения, пополняя истории все большим количеством информации.

Пример детализации истории

Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.

Что нужно, чтобы гость смог зарегистрироваться в системе?

Пример детализации истории

Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.

Нужен проверенный email и выбранные пользователем имя и пароль.

Пример детализации истории

Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.

Нужен проверенный email и выбранные пользователем имя и пароль.

- Тест 1: пользователь не может ввести пароль меньше 6 символов
- Тест 2: пользователь не может ввести имя меньше 3 и больше 20 символов
- Тест 3: пользователь должен иметь уникальное имя в системе
- Тест 4: после регистрации пользователь должен получить имейл для активизации своей учетной записи
- Тест 5: пользователь не может войти в систему, если учетная запись не была активизирована
- Тест 6: при успешном входе система приветствует пользователя текстом «Добро пожаловать, имя пользователя»

Инструменты работы с историями

Упорядочивание

Пользуясь знанием рынка, а также здравым смыслом (к сожалению на сегодняшний день оба этих критерия не поддаются численной оценке), заказчик выстраивает список историй таким образом, чтобы максимизировать возврат вложений от проекта.

Разбиение

Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать или продать их другим пользователям.

- Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать их другим пользователям.
- Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность продать их другим пользователям.

Группировка

Преимущества

Гибкие методологии предпочитают общение лицом к лицу вместо всесторонней документации; быструю адаптацию к изменениям вместо фиксации на проблеме. Это достигается следующим:

- Истории короткие. Они представляют маленькие кусочки деловой ценности, которые можно реализовать в период от дней до нескольких недель.
- Позволяют разработчикам и клиентам обсуждать требования на протяжении всей жизни проекта
- Нуждаются в очень небольшом обслуживании
- Рассматриваются только в момент использования
- Поддерживают близкий контакт с клиентом
- Позволяют разбить проект на небольшие этапы
- Подходят для проектов, где требования изменчивы или плохо поняты.
- Облегчают оценку заданий

Недостатки

- Без определенных приемочных испытаний, истории являются открытыми для различных интерпретаций, что усложняет их использование как основу для соглашения
- Требуется близкий контакт с клиентом на протяжении всего проекта, что в некоторых случаях может быть сложно либо приводить к накладным затратам
- Истории могут плохо масштабироваться на больших проектах
- Истории полагаются на компетентность разработчиков
- Истории используются для начала дискуссии. К сожалению, они могут не фиксировать окончание дискуссии и таким образом не в состоянии служить надежным методом документации системы.