Тема: Разработка и управление требованиями

Общая концепция требований

Требование - совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации.

Вопросы к требованиям

- Для чего делаем?
- Что делаем?
- Как делаем?
- Какие ограничения есть?
- Насколько срочно?
- Как убедиться, что получили то, что хотели?

Требования к проекту

- Физические ресурсы
- Патенты, товарные знаки
- План (в широком смысле)
- Потребность в обучении персонала
- Тип и количество документации
- Внешнее ПО и лицензии

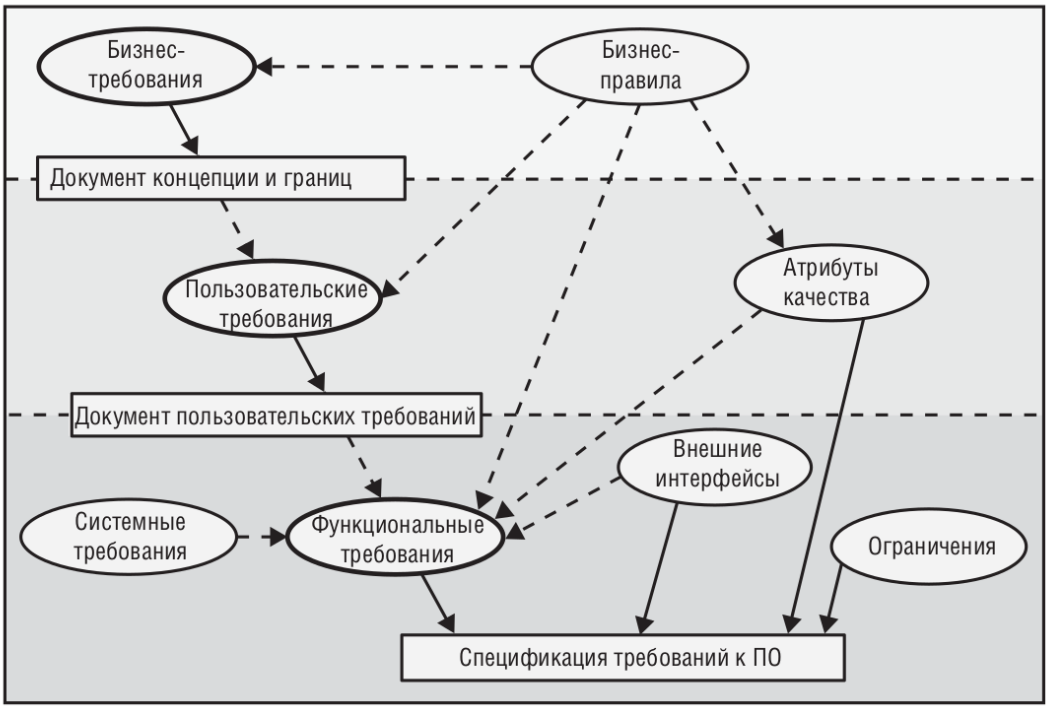
- Регламенты
- Требования по сертификации
- Бюджет
- Сроки
- Риск-менеджмент

Требования к продукту

Две большие группы - функциональные (ф) и нефункциональные (н)

- Бизнес-требование
- Бизнес-правило
- Ограничение (н)
- Внешнее требование к интерфейсу (н)
- Фича (ф)
- Атрибут качества (н)
- Системное требование (н)
- Пользовательское требование (ф)

Уровни требований к продукту



Нефункциональные требования (архитектурные характеристики)

- Доступность (Availability)
- Надежность (Durability)
- Время хранения данных
- Масштабируемость (Scalability)
- Удобство пользователя (Usability)
- Переиспользование (Reusability)
- Расширяемость (Extensibility)
- Переносимость (Portability)
- Взаимодействие между компонентами (Interoperability)

- Модульность (Modularity)
- Тестируемость (Testability)
- Требования к безопасности
- Сложность поддержки
- Возможность локализации
- Производительность
- Ограничения

Свойства качественных требований

- Завершенность
- Однозначность
- Непротиворечивость
- Необходимость
- Актуальность
- Выполнимость
- Проверяемость

Работа с требованиями

Разработка требований — непосредственно выявление, формализация и документация

Управление требованиями— изменение требований во времени, переработка, оценка влияния на проект

Разработка требований

- Выявление

- Анализ
- Документация
- Утверждение

Матрица трассировки

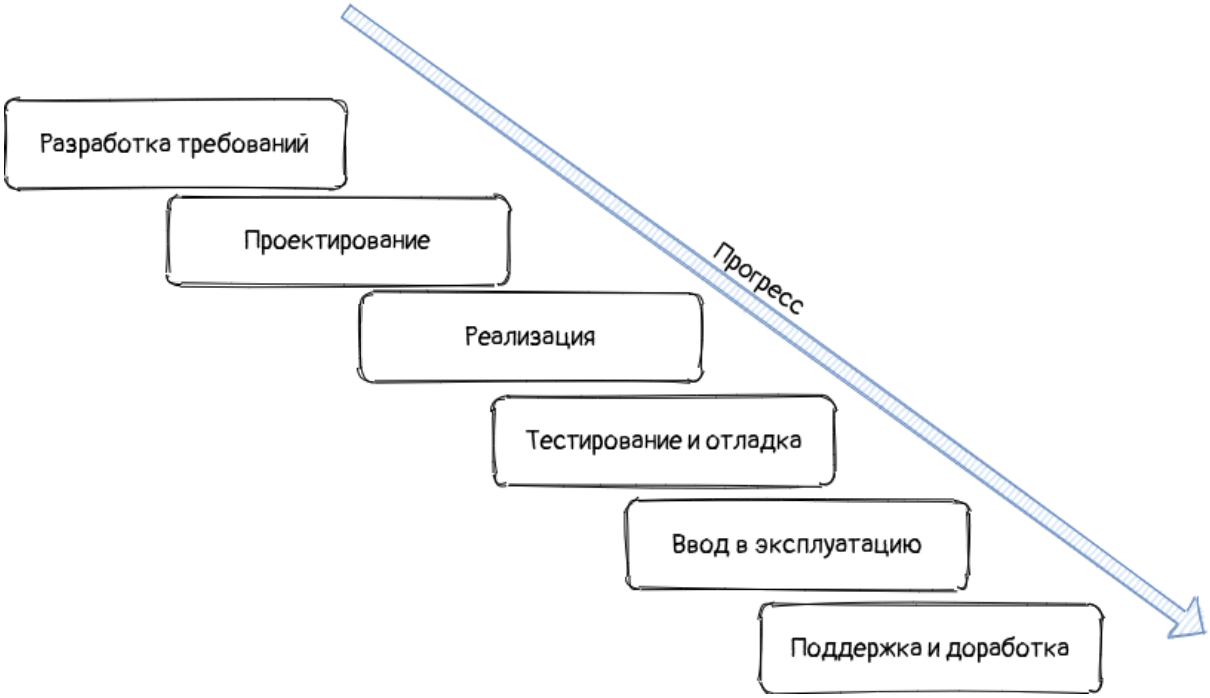
Нужна для отслеживания требований от бизнесовых к деталям и наоборот

Пользовательс-кое требование	Функциональное требование	Элемент дизайна	Элемент кода	Тест
UC-28	catalog.query.sort	Каталог классов	CatalogSort()	search.7
				search.8
UC-29	catalog.query.import	Каталог классов	CatalogImport()	search.12
			CatalogValidate()	search.13
				search.14

	Вариант использования			
Функциональное требование	**UC-1**	**UC-2**	**UC-3**	**UC-4**
FR-1	↵			
FR-2	↵			
FR-3			↵	
FR-4			↵	
FR-5		↵		↵
FR-6			↵	

Каскадная (водопадная) модель

Разработка ведется этапами, этапы не пересекаются.



Гибкие подходы (Agile)

Облегченные модели разработки, более гибкие и приспособленные к изменениям.
Манифест — https://agilemanifesto.org/iso/ru/manifesto.html

Пользовательские истории

Как **<Роль>** я хочу **<Что-то сделать>** чтобы **<Достичь какой-то цели>**

*Как **пассажир**, я хочу **зарегистрироваться на рейс** чтобы **добраться до пункта назначения**.*

Обладают свойством **INVEST**.

Сценарий использования.

Состав:

- Название
- Краткое описание/цели которые преследуются
- Пользователь или его роль
- Предусловия
- Основной сценарий работы (успешный сценарий)
- Альтернативный сценарии выполнения или же негативные кейсы
- Постусловия

Пример:

Название: Регистрация нового пользователя

Описание: Пользователь из интернета регистрируется для того, чтобы оставлять сообщения на форуме

Пользователь: Случайный пользователь интернета, который хочет ругаться в комментариях

Предусловия: Пользователь не зарегистрирован на форуме

Основной сценарий:

1. Пользователь переходит на форму регистрации по ссылке
2. Пользователь заполняет поля email и пароль
3. Пользователь нажимает кнопку зарегистрироваться
4. Система сообщает что ему отправлено письмо с подтверждением
5. Пользователь получает письмо на почту с ссылкой подтверждения регистрации

6. Пользователь переходит по ссылке
7. Система сообщает, что пользователь успешно зарегистрирован

Альтернативные сценарии: Если пользователь ввел уже существующий email, то система после нажатия кнопки Зарегистрироваться должна сообщить, что такой пользователь уже есть

Постусловия: Зарегистрированный пользователь с подтвержденным email может аутентифицироваться в системе

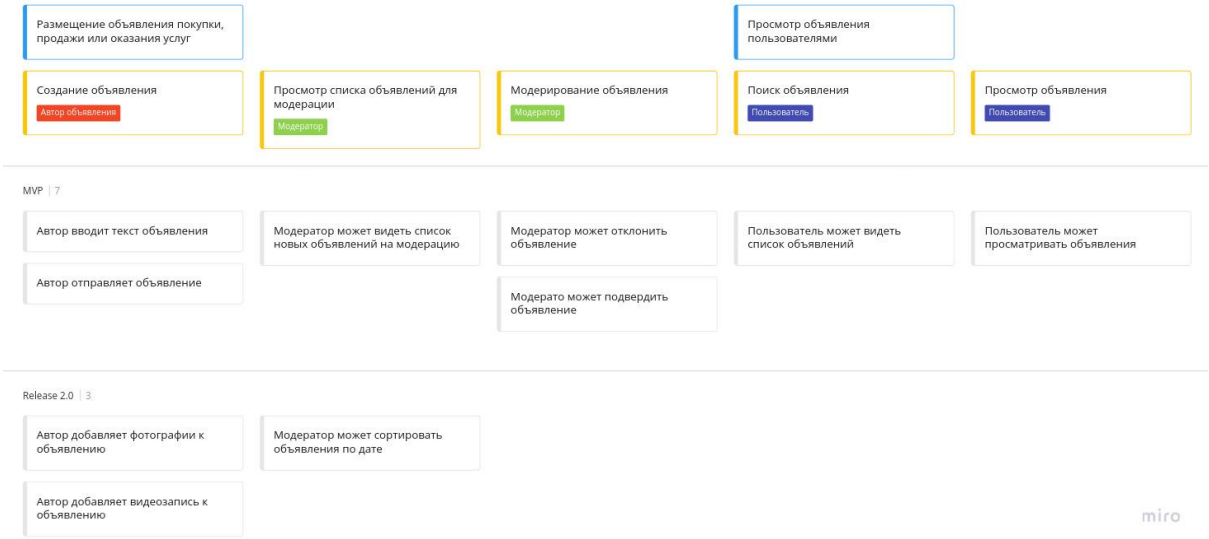
Что использовать?

Смотри лекцию :-)

Карты пользовательских историй

Позволяют увидеть лес за деревьями, сформировать общее видение, разглядеть процесс и запланировать.

Сверху вниз растет растет детализация.
Слева направо изменяется процесс во времени.



Можно использовать для описания и планирования систем, которые не содержат пользовательских интерфейсов.

Технические истории

Тот же подход как и в пользовательских, но рассматриваем с точки зрения разработчиков, аналитиков, владельцев продукта и так далее.

Было: Настроить CI-сервер
Стало: Как разработчик я хочу получить протестированный докер-образ, который заливается и деплоится на dev-стенд после того как я сделал коммит

Было: Провести нагрузочные тесты на виджет
Стало: Как владелец продукта, я хочу узнать какая пропускная способность и задержка ответа у нашего виджета, чтобы убедиться, что мы готовы для участия в распродажах

howto.stringconcat.com — Разработка и эксплуатация Enterprise-приложений на Java и Kotlin без боли и сожалений

Общий план

- Выявляем цели
- Исследуем
- Проектируем
- Планируем
- Реализуем
- Демонстрируем результат
- Оцениваем свою работу
- Собираем обратную связь
- GOTO 1

Антипаттерны

Анемичный разработчик — аналитик лезет на техническую поляну

Прямой доступ — заказчик ходит напрямую к разработчикам

ALL INCLUSIVE — подписываемся под всем, потом разберемся

Свободный художник — я художник, я так вижу

Копирайтер — пишем документацию вместо того, чтобы её сгенерировать

Фичезадача — задача в трекере выглядит как фича

Литература

1. Разработка требований к программному обеспечению, Карл Вигерс, Джой Битти
2. Пользовательские истории. Искусство гибкой разработки ПО, Джефф Паттон
3. Пользовательские истории. Гибкая разработка программного обеспечения, Майк Кон