

Лекция 6

Процессы управление ИТ-проектом

ФАЗЫ ПРОЕКТНОГО УПРАВЛЕНИЯ

1. Инициация проекта.
2. Планирование проекта.
3. Реализация.
4. Закрытие проекта.

1 фаза - ИНИЦИАЦИЯ проекта

Инициация проекта –

процесс формального санкционирования нового проекта, определение его *целей и содержания*, поставляемых *результатов, ограничений* и др.

Шаги руководителя:

- разработка устава проекта;
- утверждение его у заказчика и спонсора.

В ходе инициации проекта назначается менеджер проекта, заказчик и другие заинтересованные лица проекта

В *сложных и многофазных* проектах процесс инициации повторяется в начале каждой фазы, чтобы подтвердить неизменность целей и содержания проекта, а также его результатов.

Типовая структура устава проекта

Раздел	Содержание раздела
Название проекта	Краткое и полное название проекта, код проекта (если используется)
Команда проекта	ФИО, <u>руководителя</u> , <u>заказчика</u> , <u>участников</u> проекта. Перечень лиц, интересы которых могут быть затронуты в проекте.
Цель	Формулировка цели, т. е. ради чего был инициирован проект, какой эффект компания хочет получить в результате реализации. Цель должна соответствовать критериям SMART – Specific, Measurable, Achievable, Realistic, Timelined – конкретный, измеримый, достижимый, реалистичный, ограниченный временем
Связь со стратегией компании	Описание, какую стратегическую цель поддерживает данный проект
Результаты и их содержание	Описание вещественных и измеримых артефактов, передаваемых заказчику в ходе проекта: <ul style="list-style-type: none">– документы;– оборудование;– программный продукт и его компоненты;– обученный персонал;– здания и сооружения. Содержание должно пояснять, из каких элементов (работ, характеристик) состоит результат
Ограничения	Описываются ограничения: 1) по срокам, 2) по бюджету, 3) организационные
Риски проекта	Описание факторов (условий), которые могут позитивно или негативно повлиять на выполнение проекта.

Необходимость инициации проекта

Руководителю проекта	<ul style="list-style-type: none">➤ Получить полномочия управлять командой.➤ Узнать, кто заказчик проекта и какие у него ожидания от проекта.➤ Получить подтверждение поддержки проекта высшим руководством (спонсором).
Спонсору проекта	<ul style="list-style-type: none">➤ Увидеть взаимосвязь проекта со своей стратегией.➤ Назначить ответственных лиц за формулирование задачи (заказчик) и получение результата (руководитель проекта).➤ Оценить, какие ограничения и риски принимает на себя организация с появлением проекта.
Заказчику	<ul style="list-style-type: none">➤ Сформулировать цели проекта в том виде, которые обеспечат максимальную полезность результатов проекта для бизнеса.➤ Получить полномочия предъявлять требования к результатам проекта.

2 фаза - СБОР ТРЕБОВАНИЙ

Результаты фазы – зафиксированные требования к результатам проекта

Сбор требований *начинается* с выявления заинтересованных лиц проекта.

Заинтересованные стороны проекта (Вигерс, 2004):

- заказчики, которые финансируют проект или приобретают продукт для решения бизнес-задач;
- пользователи, которые взаимодействуют напрямую или не напрямую с приложением;
- аналитики требований, которые пишут требования и передают их разработчикам;
- разработчики, которые создают, разворачивают и поддерживают продукт;
- тестеры, которые определяют соответствие поведения ПО желаемому;
- технические писатели, создающие руководства пользователей, тренировочные материалы и справочную систему;
- менеджер по проекту, который планирует процесс и руководит командой разработчиков вплоть до успешного выпуска продукта;
- сотрудники правового отдела, наблюдающие, чтобы продукт не противоречил действующим законам и постановлениям;
- производственники, которые должны построить продукт, содержащий данное ПО;
- сотрудники отдела продаж и маркетинга, кому придется работать с продуктом и его пользователями.

2. 1 Что есть требование

Требования -

- условия или возможности, необходимые пользователю для решения проблем или достижения целей;
- условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;
- документированное представление условий или возможностей для пунктов 1 и 2.

IEEE Standard Glossary of Software Engineering Terminology (1990)

Требования...

это спецификация того, что должно быть реализовано. В них описано поведение системы, свойства системы или ее атрибуты. Они могут быть ограничены процессом разработки системы.

(Sommerville и Sawyer, 1997)

Ловушка:

лица, заинтересованные в проекте, могут не разделять общего представления о том, что такое требование.

В проекте **необходимо:**

- задать формат требования,
- определить требования к результатам проекта

2.2 Этапы разработки требований

1. Извлечение
2. Анализ
3. Документирование
4. Утверждение

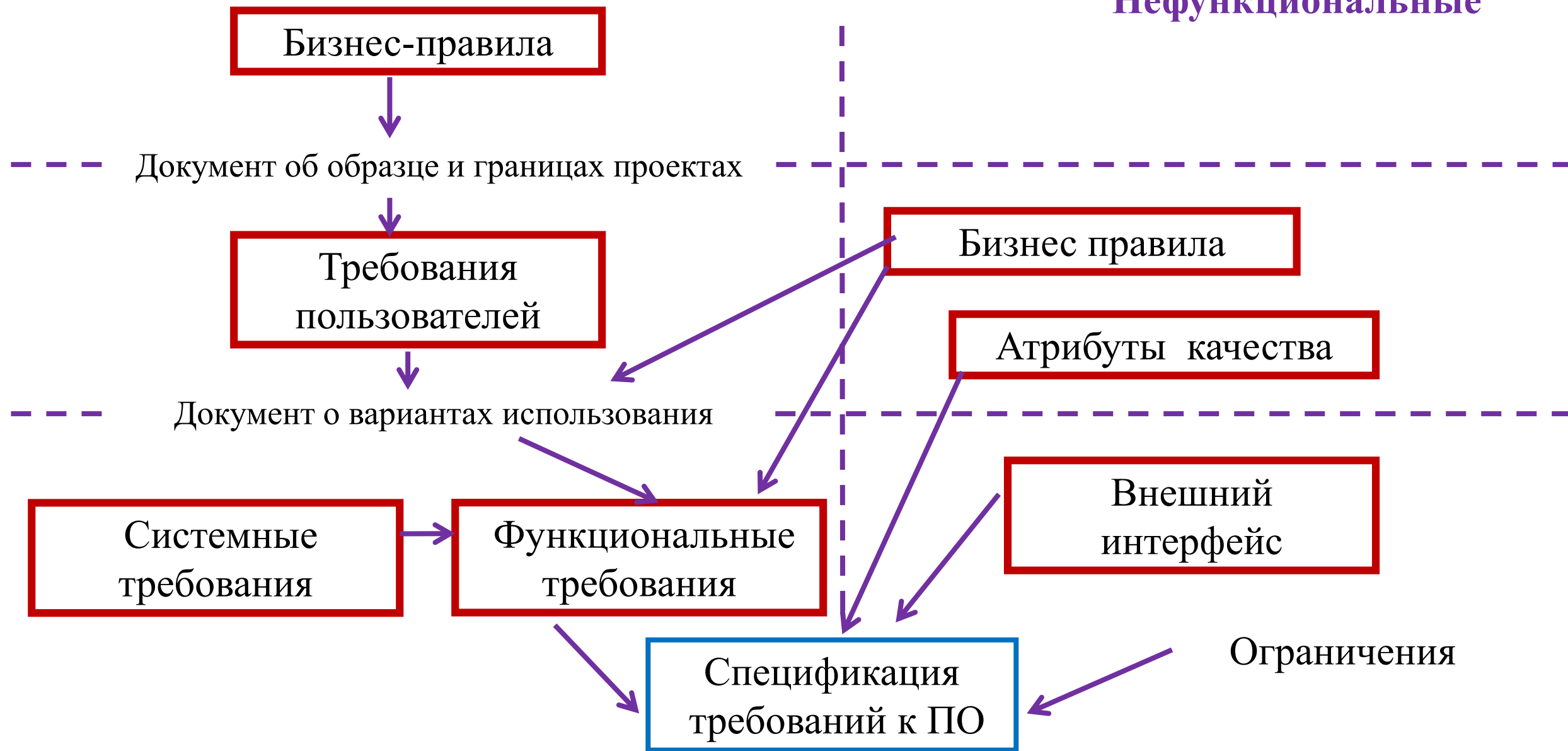
2.3 Характеристики «хороших» требований (Вигерс, 2004)

1. Полнота. Информации должно хватить для реализации требования.
2. Корректность. Для соблюдения корректности необходима связь с источниками требований – например, с пожеланиями пользователей.
3. Осуществимость. Возможность реализовать требование.
4. Необходимость.
5. Назначенный приоритет. Требуется для определения, чем можно пожертвовать, если в проекте что-то пойдет не так.
6. Недвусмысленность. Читатели требований должны понимать их одинаково.
7. Проверяемость. Требования можно протестировать.

2.4 Виды требований

Функциональные

Нефункциональные



2.5 Выявление требований

Требованиями НЕ являются :

- идеи;
- предложения;
- детали дизайна системы;
- замечания о способе реализации (кроме известных ограничений);
- данные о планировании проекта;
- сведения о тестировании.

Требования
отвечают
на вопрос «ЧТО?»,
но НЕ на вопрос «КАК?».

2.6 Методы фиксации требований

Формат фиксации требований
по типу носителя:

- в классическом бумажном виде
- в электронном виде.

Спецификация требований:

- документ
- wiki-сайт

1 Формальный метод – Техническое задание

SPS – Software Requirement Specification

IEEE Std 830

1. Введение
2. Цели
3. Соглашения о терминах
4. Предполагаемая аудитория и последовательность восприятия
5. Масштаб проекта
6. Ссылки на источники
7. Общее описание
8. Видение продукта
9. Функциональность продукта
10. Классы и характеристики пользователей
11. Среда функционирования продукта (операционная среда)
12. Рамки, ограничения, правила и стандарты
13. Документация для пользователей
14. Допущения и зависимости
15. Функциональность системы
16. Функциональный блок X (может быть несколько)
17. Описание и приоритет
18. Причинно-следственные связи, алгоритмы (движение процессов, workflows)
19. Функциональные требования
20. Требования к внешним интерфейсам
21. Интерфейсы пользователя (UX)
22. Программные интерфейсы
23. Интерфейсы оборудования
24. Интерфейсы связи и коммуникации
25. Нефункциональные требования
26. Требования к производительности
27. Требования к сохранности (данных)
28. Критерии качества программного обеспечения
29. Требования к безопасности системы
30. Прочие требования
31. Приложение А: Глоссарий
32. Приложение Б: Модели процессов и предметной области и другие диаграммы
33. Приложение В: Список ключевых задач

1. Общие сведения
2. Назначение и цели создания системы
 - Назначение системы
 - Цели создания системы
3. Характеристика объектов автоматизации
4. Требования к системе
 - Требования к системе в целом
 - Требования к функциям, выполняемым системой
 - Требования к видам обеспечения
5. Состав и содержание работ по созданию системы
6. Порядок контроля и приёмки системы
7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие
8. Требования к документированию
9. Источники разработки

Фиксирование требований в настоящее время

wiki-сайт

Wiki -

веб-сайт, структуру и содержимое которого пользователи могут самостоятельно изменять с помощью инструментов, предоставляемых самим сайтом

Форматирование текста и вставка различных объектов в текст производится с использованием [вики-разметки](#).

[Wikipedia](#)

Популярный инструмент ведения документации в стиле wiki

Google Sites

- <https://sites.google.com/>.

2 Пользовательские сценарии (UseCases)

Use Case

это сценарная техника описания взаимодействия

С помощью Use Case может быть описано

- пользовательское требование,
- требование к взаимодействию систем,
- описание взаимодействия людей и компаний в реальной жизни

В разработке ПО Use Case применяют для проектирования и описания взаимодействия пользователя и системы:

Use Case = требования пользователя к решению определенной задачи в системе

Пример Use Case

РЕГИСТРАЦИЯ ПАССАЖИРА НА РЕЙС

Система	Система регистрации пассажира на рейс
Основное действующее лицо	Пассажир
Цель	Зарегистрироваться на рейс
Триггер	Пассажир решает зарегистрироваться на рейс и заходит на страницу регистрации сайта
Результат	1) Информация о регистрации пассажира сохранена 2) У пассажира есть посадочный талон

ОСНОВНОЙ ПОТОК СОБЫТИЙ

№	Действующее лицо	Действие	Комментарий
1	Система	Запрашивает фамилию и код бронирования	-
2	Пассажир	Вводит фамилию и код бронирования Пассажира на рейс	7 символов, заглавные латинские и цифры. Например: MTDTС1
3	Система	Проверяет, что приобретен билет на этот рейс на имя этого Пассажира	-
4	Система	Сохраняет информацию о регистрации	
5	Система	Подтверждает Пассажиру, что он зарегистрирован на рейс	
6	Система	Выводит для печати посадочный талон	
7	Пассажир	Отправляет посадочный талон на печать и код бронирования	

Преимущества Use Case

- можно выявить больше функциональных требований: практически каждая строчка Use Case является отдельным функциональным требованием.
- видно, какие функции должны выполняться вместе,
- есть возможность выставлять приоритеты реализации этих требований так, чтобы они были готовы в одно время

Use Case для руководителя проекта

это естественный способ описывать диалог, он понятен человеку без подготовки, Use Case обычно не содержит деталей реализации и пишется на языке целей пользователей.

Use Case удобно согласовывать с Заказчиком как «единицу поставки»

Use Case для тестировщика

является базой для формирования тестовых сценариев - test case, так как описываются в каком контексте должно производиться каждое действие пользователя.

Use Case, по умолчанию, являются тестируемыми требованиями так как в них всегда указана цель, которой нужно достигнуть и какие шаги надо для этого воспроизвести.

Use Case для проектировщика интерфейсов позволяет

- делать специализированный интерфейс для каждой роли пользователя,
- выводить на первый план интерфейса элементы, соответствующие приоритетным целям пользователя,
- делать интерфейс более лаконичным и простым для восприятия, увеличивая скорость обучения.

Use Case для разработчика позволяет

разработчику лучше планировать работу над реализацией отдельных объектов и функций, а также снимет часть вопросов о используемых форматах данных

Use Case для технического писателя позволяет

формировать документацию на основе описанных и согласованных ранее Use Case

Use Case для аналитика и менеджера продукта это инструмент:

- разработки и обеспечения полноты функциональных требований,
- выявления видов информации (нефункциональных требований), которая необходима для работы над проектом:
 - ограничений;
 - атрибутов качества;
 - требований к пользовательскому интерфейсу;
 - внутренних правил работы в предметной области (бизнес-правил);
- обеспечения трассировки требований между исходными требованиями заказчика (бизнес-требованиями) и техническими требованиями (не только функциональными)

Ограничения метода Use Case

Use Case не обеспечивают полноту всех функциональных требований: в систему должна быть заложена сложная бизнес-логика, т.е. обработка информации в системе зависит не только и не столько от действий пользователей, а сколько от *внутренних правил взаимодействия объектов*.

3 Agile-методы – Пользовательские истории

Agile/Scrum – применяется развитие идеи пользовательских сценариев (User Story).

Пользовательские истории собираются в беклог – Backlog

User Stories

Вся функциональность программного продукта разбивается на отдельные (максимально независимые) кусочки.

В Scrum они называются «кусочки требований» - piece of requirement.

Каждый «кусочек» записывается в формате:

Я как [пользователь] хочу [действие], чтобы [польза]

Пользователь – любой пользователь в системе. В несложных системах обычно вводят 3-5 типов пользователей.

Действие – описание функциональности.

Польза – что получает пользователь в результате этого действия. Здесь фиксируется некая бизнес-ценность – ради чего это действие требуется делать.

Пример

Описание функциональности	Пользовательские истории
Электронный магазин. Добавление товаров в корзину товаров	Я как покупатель хочу положить товар в корзину, чтобы оформить заказ на все товары одновременно.
Система управления обучением. Информирование участника тренинга о месте и дате проведения тренинга.	Я как участник тренинга хочу получить письмо с подтверждением моего участия в тренинге, чтобы ничего не пропустить.

Пользовательские истории фиксировались в Scrum
в виде карточек или стикеров - «пользовательские карточки» (User Card)

Удобно –

для команд, которые полностью сидят в одной
комнате

Не удобно

для распределенных команд

online инструменты

Jira & Confluence – от компании Atlassian.
Google Docs + Spreadsheet
MS Excel
Backlog

Все пользовательские истории собираются в «беклог»
(Список пользовательских историй)

Пользовательская история в беклоге должна содержать:

- Описание
- Приоритет – обычно чем выше висит карточка с историей – тем выше приоритет
- Объем – оценка этого требования
- Релиз/итерация – когда по плану предполагается разработка истории
- Последние два требования появляются после планирования проекта.