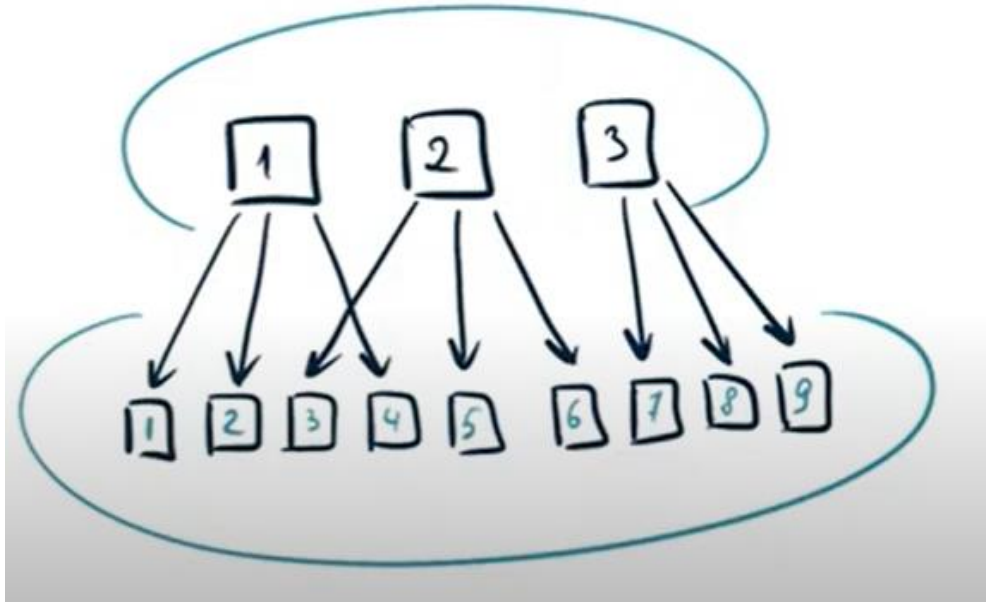
Трассировка требований

Трассировка — это способ представления отношений между требованиями различного уровня в системе, помогающий определить источник любого требования.

Трассируемость – это связь требования с уровнем выше и уровнем ниже.

Бизнес требования Заказчика



Требования нижнего уровня (спецификация, ТЗ)

Трассировка требований позволяет:

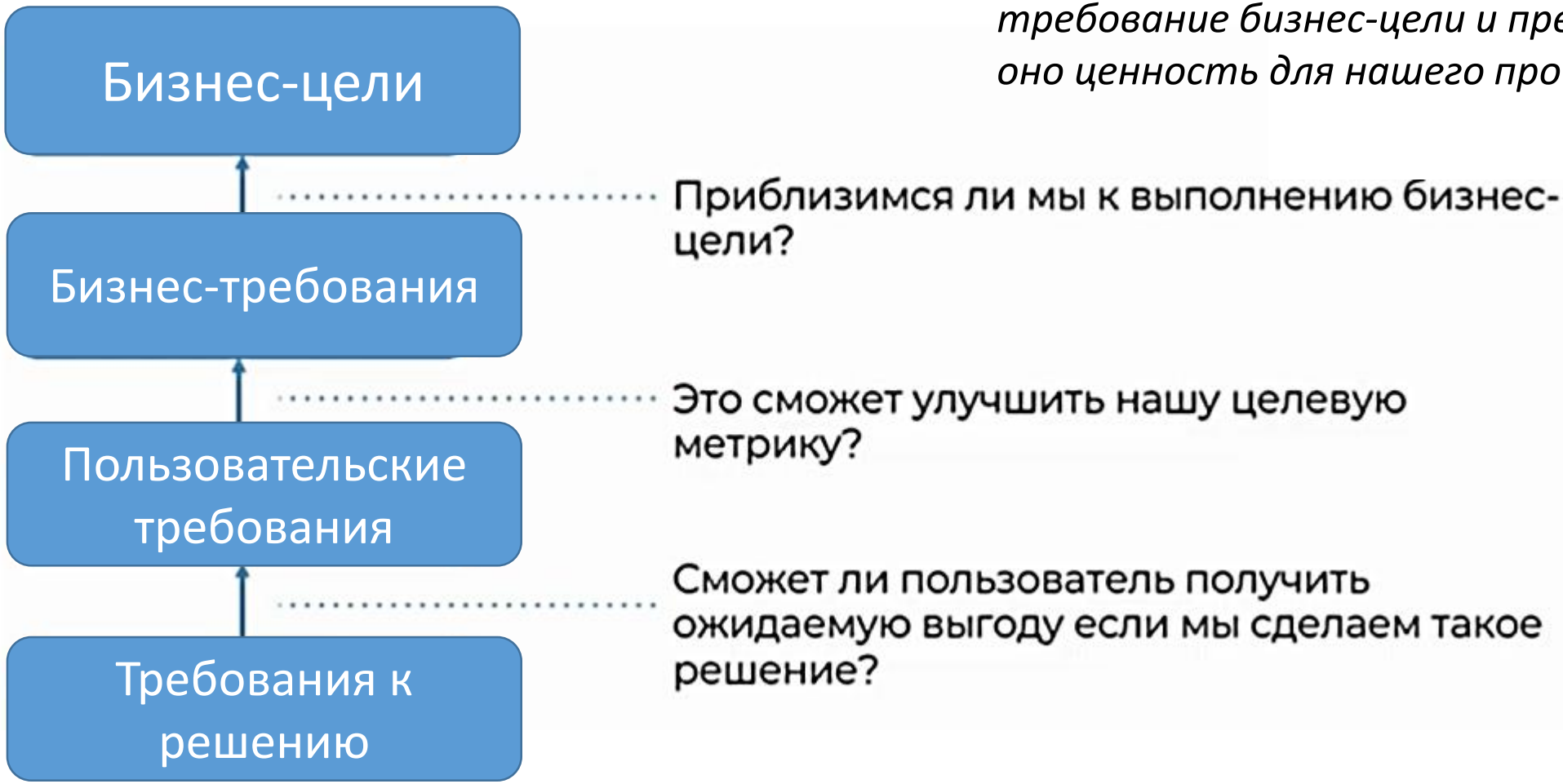
- подтвердить, что реализация удовлетворяет всем требованиям: все, что требовал заказчик, было реализовано;
- подтвердить, что приложение делает только то, что было заказано: не реализовано то, что заказчик никогда не просил;

Зачем нужна трассировка требований?

https://www.youtube.com/watch?v=lU-2r89pgvA

- Анализ важности новых требований

При появлении новых требований можно проанализировать удовлетворяет ли новое требование бизнес-цели и представляет ли оно ценность для нашего продукта



Зачем нужна трассировка требований?

- Анализ изменений

При изменении какого-либо требования верхнего уровня по трассировке можно проанализировать какие требования нижнего уровня изменятся и какие изменения нужно в них внести, что бы обеспечить изменение требования верхнего уровня

- Управление приоритетом

Пример:

У проекта 3 бизнес-цели

У каждой бизнес-цели по 4 бизнес требования

У каждого бизнес-требования по 8 пользовательских историй

96 пользовательских историй

Как приоритизировать такое количество требований ?

Пользовательские истории можно приоритизировать по приоритету связанных с ними бизнес-целей

Матрица трассировки требований (Traceability matrix)

Является документом, зачастую в форме таблицы, в котором визуально показано соотношения между двумя утвержденными базовыми документами, для определения полноты взаимосвязей между их отдельными элементами.

Пример составления TRACEABILITY MATRIX (для части функционала Калькулятора)

ЭТАП 1: Составляем нумерованный список требований к приложению (для примера возьмём несколько существующих функций калькулятора)

ID требования	*Формулировка требования*
Rq1	Ввод цифр и действий. Можно производить вычисления, нажимая на кнопки калькулятора или вводя символы с клавиатуры. Кроме того, доступен ввод цифр и действий с цифровой клавиатуры, когда нажата клавиша NUM LOCK. Также можно вставлять математические выражения из буфера обмена и получать результат (например, набрать в Блокноте «5*3=», скопировать и вставить в Калькулятор, на «экране» которого появится ответ «15»).
Rq2	Очистка экрана / Удаление символов. Полная очистка экрана осуществляется при нажатии на кнопку «C» калькулятора или клавишу Delete на клавиатуре. Последний введённый числовой символ можно удалить, используя клавишу Backspace на клавиатуре.
Rq3	Калькулятор можно использовать для выполнения операций сложения.

ЭТАП 2: Формируем список тестов – это может быть список названий детально расписанных пошаговых тест-кейсов или же чек-лист проверок

ID теста	***Проверка***
Th1	Ввод значений и операций при нажатии на кнопки калькулятора
Th2	Ввод цифр и действий с цифровой клавиатуры, когда нажата клавиша NUM LOCK
Th3	Ввод цифровых значений с клавиатуры
Th4	Вставка математических выражений из буфера обмена (на одно, два, максимально возможное количество действий)
Th5	Очистка экрана нажатием на кнопку «C» калькулятора
Th6	Очистка экрана нажатием на клавишу Delete калькулятора
Th7	Удаление последнего введённого символа нажатием на клавишу Backspace на клавиатуре
Th8	Сложение однозначных, двузначных, многозначных чисел
Th9	Сложение десятичных чисел
Th10	Сложение отрицательных чисел

ЭТАП 3: Составляем TRACEABILITY MATRIX, которая может быть представлена разными способами

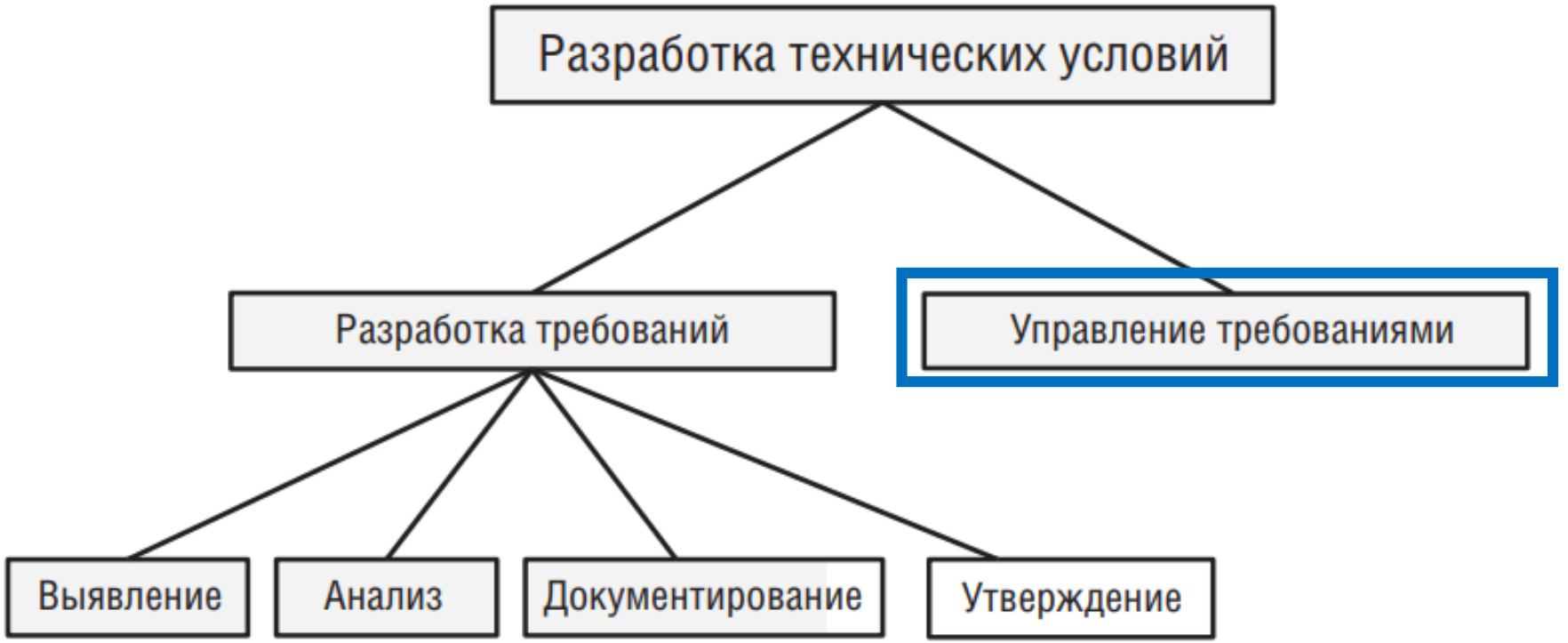
TRACEABILITY MATRIX	
ID требования	*ID теста*
Rq1	Th1
	Th2
	Th3
	Th4
Rq2	Th5
	Th6
	Th7
Rq3	Th8
	Th9
	Th10

TRACEABILITY MATRIX												
	ID теста	Th1	Th2	Th3	Th4	Th5	Th6	Th7	Th8	Th9	Th10	*Кол-во тестов, которые покрывают требование*
ID требования												
Rq1		X	X	X	X							4
Rq2						X	X	X				3
Rq3									X	X	X	3

Таким образом матрица трассируемости:

- позволяет контролировать реализацию требований, отслеживать, что все требования разработаны и протестированы и ничего не пропущено.
- помогает отслеживать какие именно требования еще не покрыты тест-кейсами.
- используется для контроля измененных требований.
- могут использоваться и заказчиком, что позволяет сделать процесс разработки и тестирования более прозрачным.

Лекция 6:
Управление требованиями



Это только требования верхнего уровня без учета производных требований

почему так важно управлять требованиями?

https://www.youtube.com/watch?v=kqMQU5ZqUb0

каковы основные ошибки, приводящие к переделке уже сделанной работы?

напрямую связаны с процессом управления требованиями

По BABOK область знаний «Анализ требований и определение дизайнов» носит инструментальный характер и сосредоточен на моделировании – именно здесь разрабатываются различные виды диаграмм, например, UML, BPMN, IDEF0, EPC, DFD или ERD, для описания поведения и строения проектируемого решения, а также процедуры работы с ним через решение следующих задач бизнес-анализа:

- Спецификация и моделирование требований (Specify and Model Requirements)
- Верификация требований (Verify Requirements)
- Валидация требований (Validate Requirements)
- Определение архитектуры требований (Define Requirements Architecture)
- Определение вариантов решения (Define Design Options)
- Анализ потенциальной ценности и рекомендация решения (Analyze Potential Value and Recommend Solution)

А управление требованиями выполняется в рамках области знаний «Управление жизненным циклом требований» и включает следующие задачи работы с требованиями:

- Трассировка (Trace Requirements)
- Поддержание (Maintain Requirements)
- Приоритизация (Prioritize Requirements)
- Оценка изменений (Assess Requirements Changes)
- Утверждение (Approve Requirements)

Управление требованиями (Requirements Management, RM) – процесс, включающий идентификацию, выявление, документацию, анализ, отслеживание, приоретизацию требований, достижение соглашений по требованиям и затем управление изменениями и уведомление заинтересованных лиц. Управление требованиями — непрерывный процесс на протяжении всего жизненного цикла продукта.

Основная цель управления требованиями — обеспечить четкое и безошибочное выполнение требований для инженерной группы, чтобы они могли убедиться в обнаружении ошибок в системе и потенциально снизить стоимость проекта, а также риск.

Разница между управлением и разработкой требований — управление требованиями — это сбор, анализ, уточнение и определение приоритетов всех продуктов или требований **на этапе разработки программного продукта**.

Примерное распределение рабочего времени бизнес аналитика на разработку и управлением требованиями:

60%

40%

Основные операции на этапе разработки требований

Основные операции на этапе управления требованиями

Базовые операции управления требованиями по Вигерсу

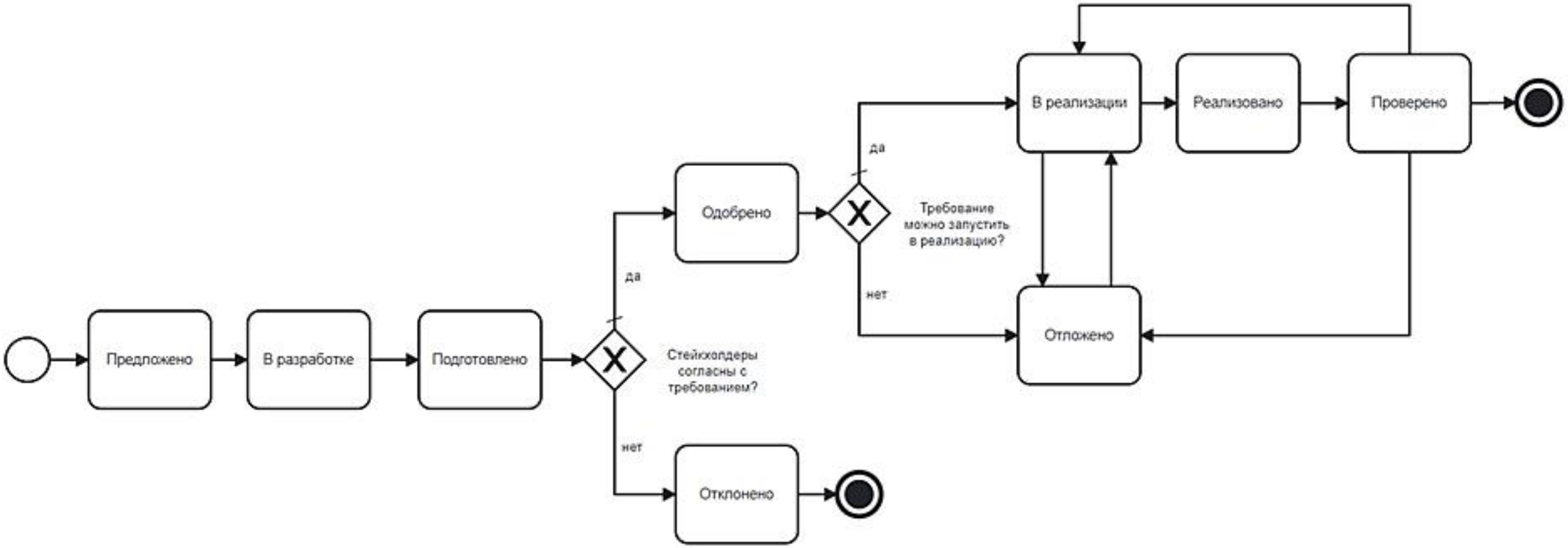
Жизненный цикл требования: *состояния*

В реальности каждая команда может определить свой набор состояний ЖЦ требования, но можно выделить набор статусов, которые чаще всего используются на практике.

Состояние	Определение
Предложено	Требование запрошено уполномоченным лицом
В разработке	Бизнес-аналитик работает над требованием
Подготовлено	Разработана начальная версия требования
Одобрено	Требование проанализировано, его влияние на проект просчитано, и оно размещено в базовой версии спецификации (SRS, ТЗ). Ключевые бизнес-стейкхолдеры согласны с этим требованием, а разработчики обязались реализовать его
В реализации	Разработчики работают над реализацией требования, т.е. пишут программный код
Реализовано	Код, реализующий требование, разработан и протестирован
Проверено	Требование удовлетворяет критериям приемки, что подтверждено соответствующими тест-кейсами. Требование считается завершенным
Отложено	Одобренное требование запланировано для реализации в более позднем выпуске. В СУТ следует указать причину переноса, например, отсутствие окончательного согласия по требованию или невозможность вписать требование с низким приоритетом в текущую итерацию из-за ограниченных ресурсов (времени, денег или емкости команды)
Отклонено	Требование предложено, но не запланировано для реализации ни в одном из релизов. В СУТ следует указать причины отклонения требования и ЛПР – стейкхолдера, кто принял такое решение

Жизненный цикл требования: *состояния*

Переходы между этими состояниями можно представить в виде UML-диаграммы состояний, которая представляет собой направленный граф от начальной точки к конечной через вершины. Каждая вершина представляет собой конкретное состояние ЖЦ требования из вышеприведенной таблицы.



Жизненный цикл требования: *процессы*

К процессам ЖЦ требования относятся :

- ✓ Сбор (Выявление);
- ✓ Спецификация;
- ✓ Верификация;
- ✓ Валидация;
- ✓ Трассировка;
- ✓ Приоритизация;
- ✓ Поддерживание *;
- ✓ Запрос изменений;
- ✓ Оценка изменений;
- ✓ Одобрение (Утверждение).

Разграничить ответственность участников, вовлеченных во все эти процессы, поможет **матрица ответственности**, которую также часто называют **RACI**.

* **Поддерживание требования:** *После того, как вы определили и отследили свои Требования, вам необходимо убедиться, что они поддерживаются на протяжении всего их жизненного цикла. Это включает в себя поддержание их в актуальном состоянии, обеспечение их актуальности и обеспечение того, чтобы они не были устаревшими из-за других требований.*

Матрица ответственности или RACI-таблица

Эта таблица показывает степень ответственности и вовлечения стейкхолдеров относительно изменения в следующих терминах:

- ✓ ***Исполнитель (Responsible, R)***— стейкхолдер, непосредственно выполняющий задачу, например, бизнес-аналитик, тестировщик;
- ✓ ***Ответственный (Accountable, A)*** — стейкхолдер, отвечающий за успешное выполнение задачи и принимающий решения. По сути, это владелец бизнес-процесса. В каждом процессе эту роль может выполнять занимать только 1 стейкхолдер, двоевластие не допускается.
- ✓ ***Консультирующий (Consulted, C)*** – стейкхолдер, обладающий специальными знаниями или опытом, которыми он может поделиться, например, эксперт предметной области;
- ✓ ***Информируемый (Informed, I)*** — стейкхолдер, которого следует держать в курсе о ходе выполнения процесса и/или его результатах. В отличие от консультирующего, коммуникация с информируемым стейкхолдерам направлена в одну сторону: от аналитика к заинтересованной стороне. Эту роль чаще всего выполняют регулятор и клиент (Заказчик).

Процесс ЖЦ требований	Аналитик	Заказчик	Эксперт домена	Руководитель проекта	Конечный пользователь	ИТ-архитектор	Разработчик	Тестировщик
Сбор	R	I	C	A	C	C	C	C

Матрица ответственности (RACI-таблица) по процессам ЖЦ требований к проекту

Процесс ЖЦ требований	Аналитик	Заказчик	Эксперт домена	Руководитель проекта	Конечный пользователь	ИТ-архитектор	Разработчик	Тестировщик
Сбор	R	I	C	A	C	C	C	C
Спецификация	A	I	C	I	C	C	C	C
Верификация	A	I	C	I	C	C	C	C
Валидация	R	A	C	I	C	C	C	C
Трассировка	R	I	C	I	C	C	C	C
Приоритизация	R	A	C	I	C	C	C	C
Поддержание	A, R		C	I	C	C	C	C
Запрос изменений	A	R	R	I	R	C	C	C
Оценка изменений	A, R	C	C	I	C	C	C	C
Одобрение (Утверждение)	R	A	C	I	C	C	C	C

Ось ролей

Ось задач

Работа	Папа	Мама	Сын	Бабушка	Кот
Подготовить машину	A	I	I	-	-
Купить еду	R	A	R	C	-
Взять игрушки	C	A	R	-	-
Взять одежду	R	A	R	R	-
Взять напитки	A	-	-	-	-
Замариновать шашлык	A	-	R	-	-
Взять рассаду	R	C	R	A	-
Взять инструмент	A	I	-	-	-
Оценить погоду	I	I	A	I	-

Исполнитель (Responsible, R)
Ответственный (Accountable, A)
Консультирующий (Consulted, C)
Информируемый (Informed, I)

https://habr.com/ru/companies/timeweb/articles/597281/

Системы управления требованиями

1. Что такое Системы управления требованиями

Системы управления требованиями (СУТ, англ. **Requirements Management Systems, RMS**) помогают аналитикам, проектировщикам и руководителям проводить сбор, фиксирование требований, их систематизацию, приоритизацию, построение взаимосвязей. Такие программные продукты применяются на протяжении всего жизненного цикла продукта.

2. Зачем бизнесу Системы управления требованиями

Суть процесса управления требованиями заключается в том, чтобы обеспечить понимание и согласование всех заинтересованных сторон по поводу того, что должен представлять проект или продукт, и чтобы эти требования были документально зафиксированы.

3. Назначение Системы управления требованиями

Основное назначение программных Систем управления требованиями - облегчение и улучшение процесса управления требованиями к проекту с целью повышения его эффективности и качества.

Системы управления требованиями

4. Основные цели системы управления требованиями включают:

- ✓ **Управление сложными требованиями**. Системы управления помогают организовать сбор и управление множеством требований к проекту, включая их анализ и приоритизацию.
- ✓ **Улучшение коммуникации**. Использование единой системы управления требованиями позволяет уменьшить ошибки при передаче требований от команды к команде, а также улучшает общение между пользователями, заказчиком и командой разработки.
- ✓ **Снижение затрат.** Управление требованиями позволяет снизить риски разработки и время на внесение изменений в продукт, за счет эффективного управления требованиями на всех этапах проекта.
- ✓ **Улучшение качества продукта**. Системы управления помогают гарантировать, что все требования были учтены и реализованы в продукте, что улучшает качество и доверие к продукту.
- ✓ **Управление изменениями и различными версиями требований**. Системы управления требованиями позволяют отслеживать изменения в требованиях и решать конфликты между различными требованиями и версиями документов.

Системы управления требованиями

5. Основные функции и возможности Системы управления требованиями

- ✓ **Администрирование**. Возможность администрирования позволяет осуществлять настройку и управление функциональностью системы, а также управление учётными записями и правами доступа к системе.
- ✓ **Импорт/экспорт данных**. Возможность импорта и/или экспорта данных в продукте позволяет загрузить данные из наиболее популярных файловых форматов или выгрузить рабочие данные в файл для дальнейшего использования в другом ПО.
- ✓ **Многопользовательский доступ**. Возможность многопользовательской доступа в программную систему обеспечивает одновременную работу нескольких пользователей на одной базе данных под собственными учётными записями. Пользователи в этом случае могут иметь отличающиеся права доступа к данным и функциям программного обеспечения.
- ✓ **Наличие API.** Часто при использовании современного делового программного обеспечения возникает потребность автоматической передачи данных из одного ПО в другое. Например, может быть полезно автоматически передавать данные из Системы управления взаимоотношениями с клиентами (CRM) в Систему бухгалтерского учёта (БУ). Для обеспечения такого и подобных сопряжений программные системы оснащаются специальными Прикладными программными интерфейсами (англ. API, Application Programming Interface). С помощью таких API любые компетентные программисты смогут связать два программных продукта между собой для автоматического обмена информацией.
- ✓ **Отчётность и аналитика.** Наличие у продукта функций подготовки отчётности и/или аналитики позволяют получать систематизированные и визуализированные данные из системы для последующего анализа и принятия решений на основе данных.

Системы управления требованиями

6. Система управления требованиями (СУТ) должна хранить следующую информацию:

- ✓ **все виды требований** (бизнес-требования, требования стейкхолдеров, а также требования к решению функциональные и нефункциональные) во всех формах их представления (Use Case, User Story, Каноническая форма);
- ✓ **трассировки требований разного уровня между собой**, а также их связи с бизнес-правилами, тест-кейсами, компонентами системы и задачами на реализацию в таск-трекере;
- ✓ **метаданные требований**, т.е. их атрибуты, включая версии и состояния жизненного цикла.

Наиболее **важными атрибутами требования**, которые описывают его, считаются следующие:

- дата создания;
- номер текущей версии;
- автор (создатель требования в СУТ);
- приоритет, который показывает его относительную важность для стейкхолдеров;
- статус, т.е. состояние жизненного цикла;
- происхождение или источник;
- логическое обоснование;
- контактное лицо, ответственное за принятие решений по одобрению и внесению изменений;
- номер выпуска или итерации, на которую назначена реализация;
- метод проверки или критерий приемки.

Проблемы в управлении требованиями

1. Бизнес-цели, концепция и границы вашего проекта никогда четко не определялись

Есть только идея, цели не определены, значит требования будут постоянно изменяться

Ссылка на ресурс:
[Проблемы в управлении требованиями к ПО - YouTube]

2. У клиентов никогда не хватало времени на работу над требованиями с аналитиками или разработчиками

3. Ваша команда не может напрямую взаимодействовать с непосредственными пользователями, чтобы разобраться с их потребностями

В качестве ответов на многие вопросы будут приняты допущения, важные детали могут быть упущены из-за непонимания разработчиком бизнес-процесса

Проблемы в управлении требованиями

4. Клиенты считают все требования критически важными, поэтому они не разбивают их по приоритетам

Нет приоритетов, значит все делается одновременно, или разработчик может больше внимания уделить тому, что ему более понятно или показалось наиболее важным, при этом действительно важные моменты могут быть нереализованы или не в полной мере

5. В процессе работы над кодом разработчики сталкиваются с многозначностью и отсутствием информации, поэтому им приходится догадываться о многих вещах

6. Общение между разработчиками и заинтересованными лицами кон центрируется на окнах и функциях интерфейса, а не на задачах, которые пользователи будут решать с использованием программного продукта

Упор на интерфейс пользователя, а функционал отодвигается на второй план, это ведет к тому, что проект может не уложиться в сроки

Проблемы в управлении требованиями

7. Ваши клиенты никогда не одобряли требования

Стоит договориться о процедуре согласования требований, а не довольствоваться устными или неформальными рассуждениями

8. Ваши клиенты одобрили требования к релизу (версии), после чего продолжили вносить в них изменения.

9. Область действия проекта увеличилась вместе с принятием изменений в требования, но сроки были нарушены из-за отсутствия дополнительных ресурсов и отказа от удаления части функциональности

10. Затребованные изменения требований были утеряны, и никто не знает состояние запроса на указанные изменения

11. Клиенты запросили определенную функциональность и разработчики реализовали ее, но никто ее не использует

Нужно постоянно показывать заказчику прототипы или презентации

12. В конце проекта спецификация была выполнена, но бизнес-задачи не были выполнены, и клиент в итоге не удовлетворен результатами проекта

Книга Фредерика Брукса «Мифический человеко-месяц, или Как создаются программные системы» об управлении проектами в области разработки программного обеспечения

Фредерик Брукс сделал вывод: «Время выполнения проекта не обратно пропорционально числу программистов, причины:

- В программировании работа не может быть произвольно разделена на несколько независимых частей. Части проекта зависят друг от друга, и некоторые задачи можно начинать выполнять только после того, как будут закончены другие.
- Программисты должны тратить часть времени на взаимодействие друг с другом.

Если есть N специалистов, то количество пар специалистов (для взаимодействия между собой, в нашем случае «консультант – разработчик») равно N(N–1)/2, то есть с ростом числа участников проекта затраты времени на взаимодействие растут квадратично. Поэтому начиная с какого-то N рост числа участников проектной команды замедляет выполнение проекта».

Если сроки сорваны, наем новых специалистов замедляет выполнение проекта по причине того, что новичкам требуется время на изучение текущего состояния задач, подходов к разработке. В книге сформулирован **закон Брукса**: «Если проект не укладывается в сроки, то добавление рабочей силы задержит его еще больше».

https://www.antonpiskun.pro/pro-upravlenie-razrabotkoj-programmnogo-obespecheniya-prostymi-slovami-sbor-trebovanij-i-ih-oczenka/