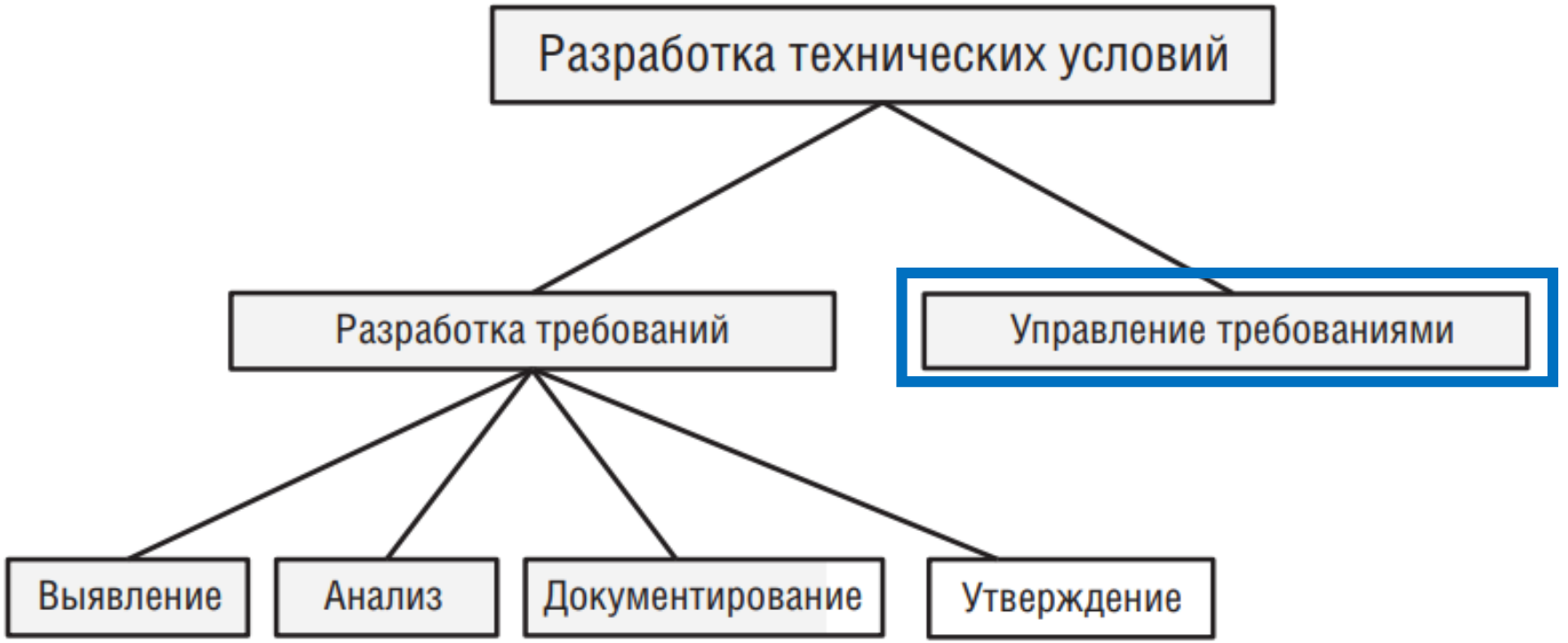
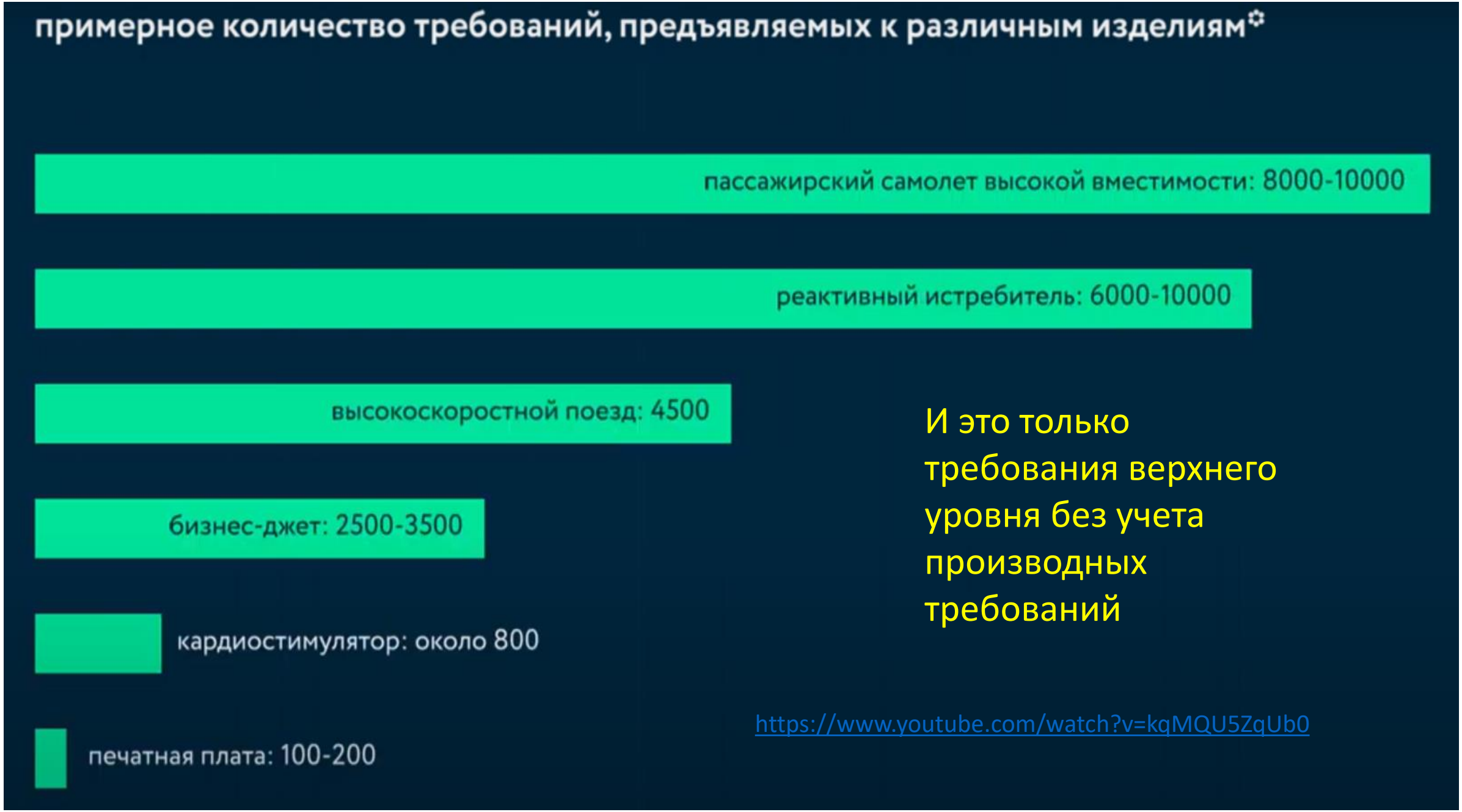
Лекция 6:
Управление требованиями





И это только требования верхнего уровня без учета производных требований

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

каковы основные ошибки, приводящие к переделке уже сделанной работы?

напрямую связаны с процессом управления требованиями

Управление требованиями (Requirements Management, RM) – процесс, включающий идентификацию, выявление, документацию, анализ, отслеживание, приоритизацию требований, соглашений по требованиям и 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**.

По **BABOK** процесс **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)

Процессы и системы управления требованиями для аналитика (babok-school.ru)

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

Состояние требования (жизненный цикл требования)

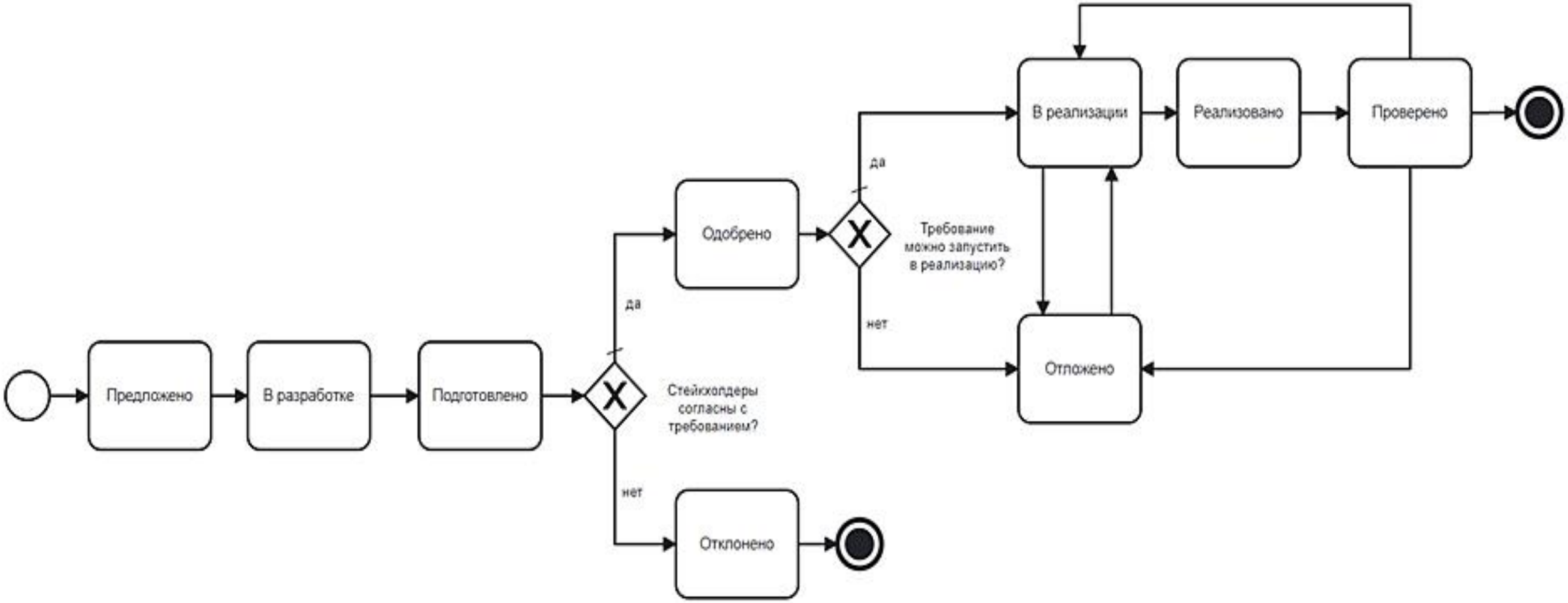
В реальности каждая команда может определить свой набор состояний ЖЦ требования, но можно выделить набор статусов, которые чаще всего используются на практике.

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

Состояние	Определение
Реализовано	Код, реализующий требование, разработан и протестирован
Проверено	Требование удовлетворяет критериям приемки (Acceptance Criteria), что подтверждено соответствующими тест-кейсами. Требование считается завершенным
Отложено	Одобренное требование запланировано для реализации в более позднем выпуске. В системе управления требованиями 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, людей)
Отклонено	Требование предложено, но не запланировано для реализации ни в одном из релизов. В системе управления требованиями 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***

Процессы в ЖЦ требования :

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

Разграничить ответственность участников, вовлеченных во все эти процессы, поможет **матрица ответственности** или **RACI матрица**.

* **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-матрица

Термин RACI является аббревиатурой и показывает степень ответственности и вовлечения :

- ***Исполнитель (Responsible, R)***—непосредственно выполняющий задачу, например, бизнес-аналитик, тестировщик;
- ***Ответственный (Accountable, A)*** — отвечающий за успешное выполнение задачи и принимающий решения. По сути, это владелец бизнес-процесса. В каждом процессе эту роль может выполнять занимать только один человек, двоевластие не допускается.
- ***Консультирующий (Consulted, C)*** – обладающий специальными знаниями или опытом, которыми он может поделиться, например, эксперт предметной области;
- ***Информируемый (Informed, I)*** — которого следует держать в курсе о ходе выполнения процесса и/или его результатах. В отличие от консультирующего, коммуникация с информируемым 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

Рекомендации по составлению матрицы

1.В RACI матрицу включают не фичи продукта, а этапы проекта, хотя их еще называют таски/задачи. Это может быть задача по разработке, проектированию, подготовке отчетности по завершению проекта, но не задача «передвинуть кнопочку.

2.Матрица лучше работает, когда рассматривают от 10 до 25 этапов проекта.

3.Этап проекта должен выражаться через конкретный глагол, вроде «провести ревью», «опубликовать отчет», «оценить сроки», «составить расписание», «собрать данные», «утвердить концепцию».

4.В матрице лучше использовать роли, а не имена. Если завтра Васю сбивает автобус, пусть это цинично, но его роль будет исполнять другой человек.

5.Разрабатывать RACI матрицу лучше группой от 4 до 10 человек. Из одной головы можно неверно оценить ситуацию, а слишком большое количество человек — просто долго и сложно для обсуждения. Но перед утверждением RACI матрицы, каждый, кто будет в ней участвовать, должен увидеть свою роль и согласиться с ней.

6.Лучше устанавливать A и R на минимальный соответствующий уровень.

7.Назначать на A — Accountable, т.е. наделять ответственностью можно только человека с полномочиями.

8.Сведите к минимум позиции I и C.

9.Если вы составили матрицу, то оповестите всех участников проекта о матрице, об их роли в проекте и уточните, уточните, согласны ли они с этой ролью. Если нет, матрицу придется дорабатывать.

10.RACI матрицу иногда приходится обновлять по ходу проекта, т.к. она может устареть.

Проверка матрицы по горизонтали – по этапам

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

- Нет задач без **R** .
- Слишком много **R** (Работа может замедлиться).
- Нет задач без **A** (Нет ответственных, нет выполненной работы).
- Больше, чем одна **A**
- Слишком много **I** (лишние созвоны и совещания)
- Нет избыточных **C** (слишком много консультаций замедляет процесс).

Проверка матрицы по вертикали – по ролям

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

- Слишком много **R**.
- Нет **R** или **A.** Вопрос- нужна ли эта роль, можно ли перераспределить
- Нет пустых мест Человеку в этой роли действительно необходимо участвовать во всех этапах проекта?»
- Слишком много **A** Можно ли снизить количество процессов, в которых человек в этой роли принимает решение и отчитывается за них?

Проверка матрицы (подробнее)

Проверяют матрицу по двум направлениям: по вертикали и по горизонтали.

Вертикальная (по ролям) т.е. оценивается насколько сбалансированы роли в проекте

- ✓ **Слишком много R**

  Проверочный вопрос: «Вывезет человек на этой роли так много задач?»
- ✓ **Нет пустых мест**

  Проверочный вопрос: «Человеку в этой роли действительно необходимо участвовать во всех этапах проекта?»
- ✓ **Слишком много A**

  Проверочный вопрос: «Можно ли снизить количество процессов, в которых человек в этой роли принимает решение и отчитывается за них?»
- ✓ **Нет R или A**

  Проверочный вопрос: «Это нужная роль? Можно ли перераспределить часть ответственности на другие роли или передать часть ответственности с других ролей на эту?»
- ✓ **Двойные позиции A/R, или A/I**

  A/R, A/C, R/C — возможная картина для очень маленькой компании, но лучше избегать.
  A/I, R/I, C/I- свидетельствует о непонимании RACI матрицы как инструмента.
- ✓ **Общая картина**

  Проверочный вопрос: «Соответствует ли количество ответственности на проверяемой роли уровню подготовки человека / его вовлеченности в проект?

Горизонтальная (по этапам)

По сути просматриваем, есть на каждом этапе человек, который будет принимать решение, будет это выполнять, и т.д.

- ✓ **Нет R** Нет исполнителя данного этапа. Проверочный вопрос: «Кто будет это делать?»
- ✓ **Слишком много R** Работа может замедлиться. Проверочный вопрос: «Можно ли разбить этап на более конкретные задачи, чтобы снизить количество исполнителей в этапе?»
- ✓ **Нет A** Нет ответственных, нет выполненной работы. Проверочный вопрос: «У кого есть полномочия из участников проекта, чтобы принять окончательное решение и иметь дело с последствиями данного этапа?». Т.е. кто отдает команду и кому снесут голову с плеч, если решение оказалось неверным
- ✓ **Больше, чем одна A** Первое правило RACI матрицы — только одна A для каждого этапа проекта.
- ✓ **Все ячейки заполнены на каждом этапе** Проверочный вопрос: «Действительно ли нужны все участники проекта на каждом этапе? Есть при преимущества от их участия?»
- ✓ **Нет C/I** Возможно вы кого-то забыли. Проверочный вопрос: «Вы учли всех участников проекта? Коммуникация между «отделами» налажена? Могут ли отделы начать выполнять какой-то этап параллельно или без учета последних изменений? »
- ✓ **Слишком много C** Замедляет проект. Проверочные вопросы: «Действительно нужно консультироваться со всеми C? Затраты часов на эти консультации окупаются? Можно просто проинформировать эти роли?»
- ✓ **Слишком много I** Проверочные вопросы: «Приносит ли это информирование пользу проекту? Или просто создает лишние созвоны и совещания? Можно информировать людей на этих ролях только в исключительных обстоятельствах? Можно ли отказаться от информирования людей на этих ролях?»
- ✓ **Много двойных позиций** Да, в маленькой команде могут возникать «двойные позиции», к примеру, когда на одном и том этапе один человек и выполняет задачу R и принимает по ней конечное решение A. Но такая формулировка может показывать непонимание матрицы, задач, команды, ответственности.

Ось ролей

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

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)

Трассировка требований

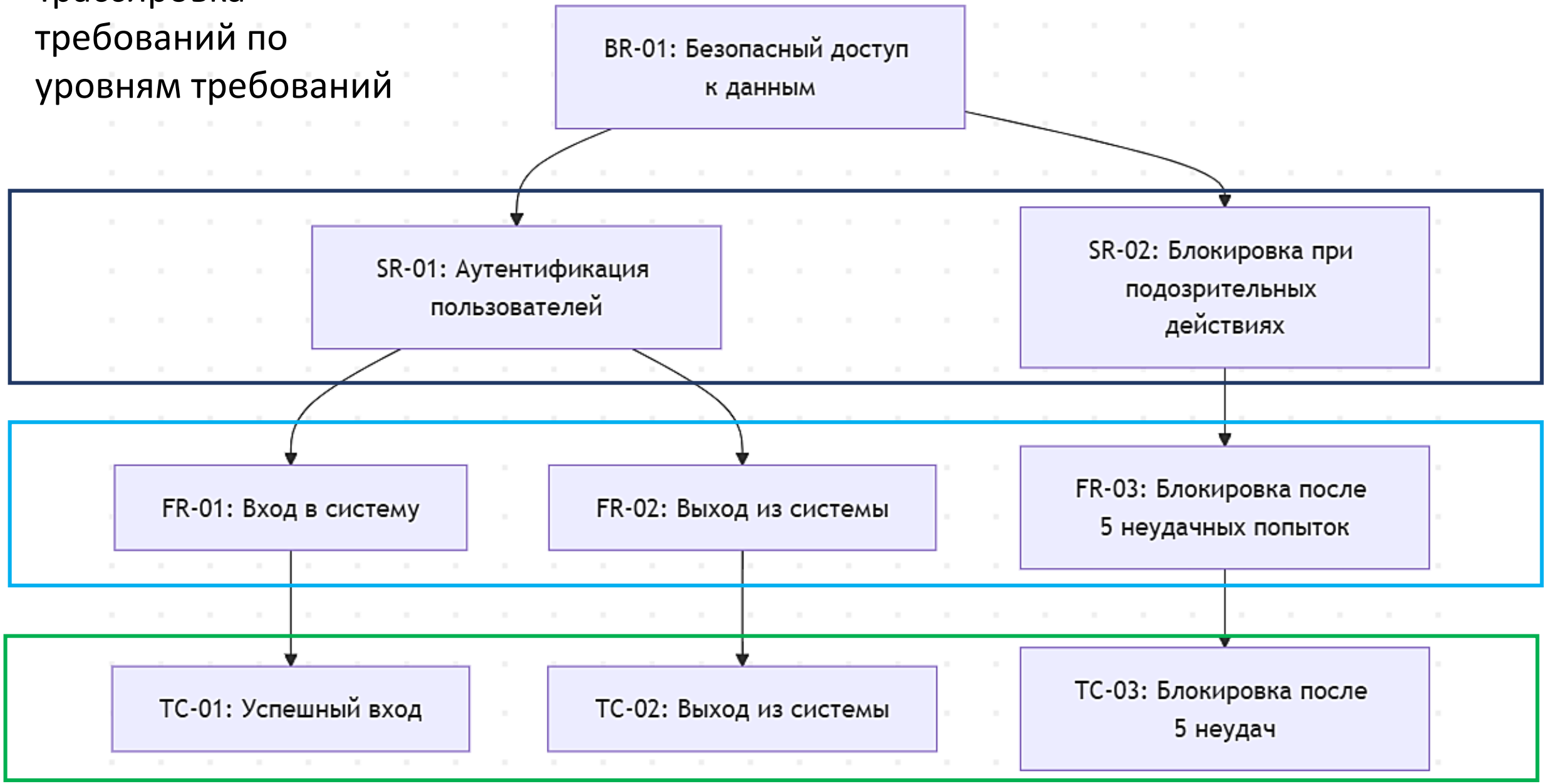
Трассировка требований (Requirements Traceability) — это процесс установления, документирования и поддержания прослеживаемых связей между требованиями и другими артефактами разработки ПО на протяжении всего жизненного цикла системы.

- ✓ Обеспечивает отслеживание взаимосвязей между:
  - исходными требованиями (бизнес-цели, нормативные документы).
  - детализированными требованиями (SRS, пользовательские истории).
  - дизайном (архитектура, UML-диаграммы).
  - кодом (классы, модули).
  - тестами (тест-кейсы, сценарии проверки).
- ✓ Поддерживает как прямую (от требований к коду/тестам), так и обратную (от тестов к требованиям) трассировку.

Таблица трассировки требований по уровням требований

Уровень	ID	Тип Требования	Описание	Связи вниз (Child)	Связи вверх (Parent)
1	BR-01	Бизнес-требование	Система должна обеспечивать безопасный доступ к данным	SR-01, SR-02	—
2	SR-01	Системное требование	Поддержка аутентификации пользователей	FR-01, FR-02	BR-01
2	SR-02	Системное требование	Реализация блокировки при подозрительных действиях	FR-03	BR-01
3	FR-01	Функциональное	Пользователь должен иметь возможность войти в систему	TC-01	SR-01
3	FR-02	Функциональное	Пользователь должен иметь возможность выйти из системы	TC-02	SR-01
3	FR-03	Функциональное	Система должна блокировать аккаунт при 5 ошибках входа	TC-03	SR-02
4	TC-01	Тест-кейс	Проверка успешного входа с корректными данными	—	FR-01
4	TC-02	Тест-кейс	Проверка выхода из системы	—	FR-02
4	TC-03	Тест-кейс	Проверка блокировки аккаунта после 5 неудачных входов	—	FR-03

Трассировка требований по уровням требований



Матрица трассировки требований

Документ табличного вида, который связывает требования с их реализацией на разных стадиях разработки: проектированием, реализацией, тестированием и т. д.

Данный термин — абсолютный рекордсмен по количеству русскоязычных названий. Их не меньше десяти:

- Матрица отслеживания требований
- Матрица прослеживаемости требований
- Матрица сверки требований
- Матрица слежения за требованиями
- Матрица соответствия требований
- и просто матрица требований

Также встречаются названия:

- Матрица трассировки требований
- Матрица трассируемости (и именно это название одобрено ISTQB)
- и матрица трассабилити.
- Также называют просто RTM-матрицей.

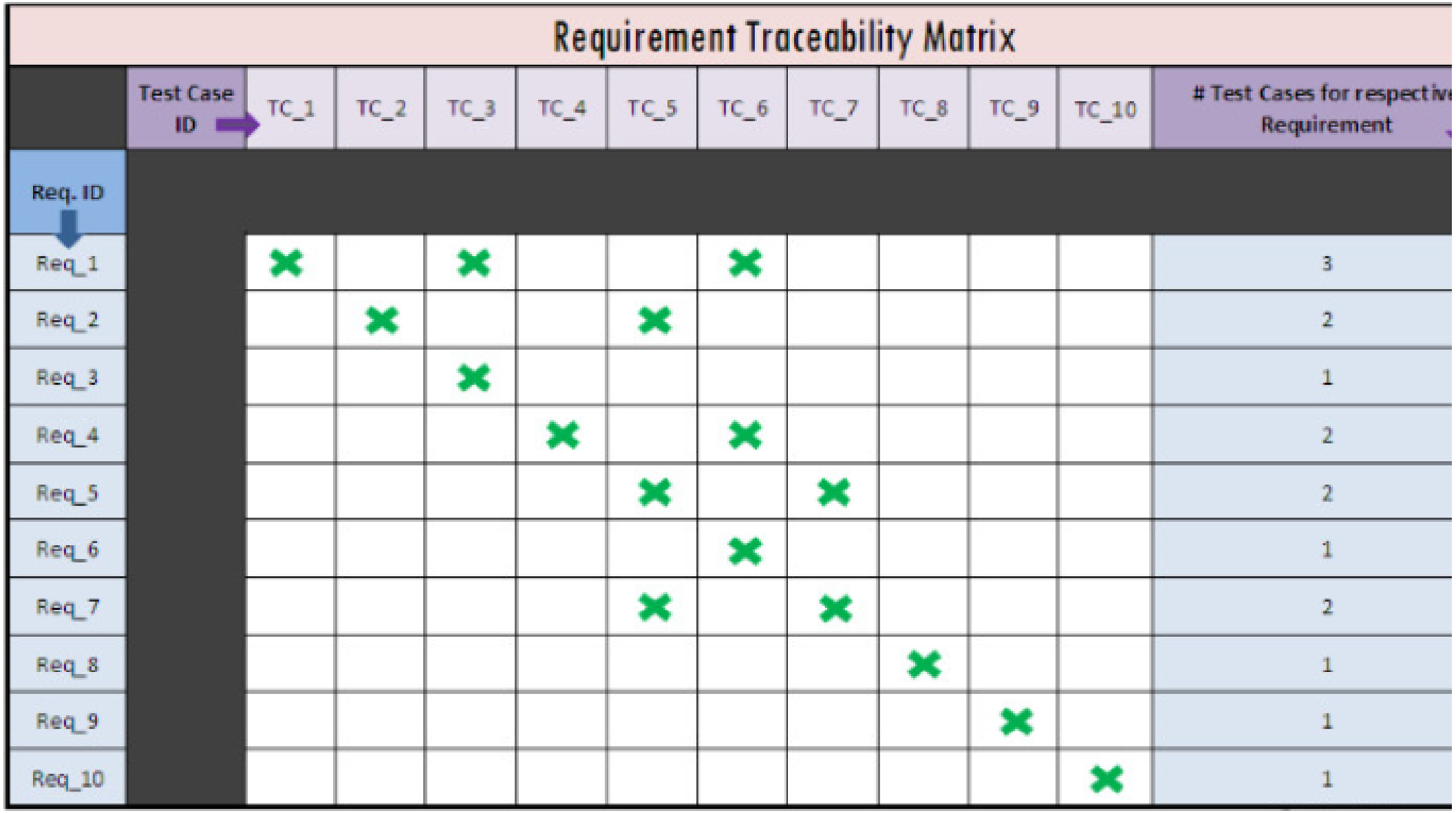
Пример матрицы отслеживания требований

	A	B	C	D	E
1	RTM Template				
2	**Requirement number**	**Module number**	**High level requirement**	**Low level requirement**	**Test case name**
3	2	Loan	**2.1 Personal loan**	2.1.1--> personal loan for private employee	beta-2.0-personal loan
4				2.1.2--> personal loan for government employee	
5				2.1.3--> personal loan for jobless people	
6			**2.2 Car loan**	2.2.1--> car loan for private employee	
7					
8			**2.3 Home loan**		
9					
10					
11					

В этом примере матрицы указана связь с архитектурой

ID Требования	Описание Требования	Связанные Бизнес-Требования	Архитектура/ Дизайн	Код/Модуль	Тест-Кейсы	Статус
FR-01	Пользователь может войти в систему	BR-01	AuthModule	login.py	TC-01, TC-02	Выполнено
FR-02	Пользователь может выйти	BR-01	AuthModule	logout.py	TC-03	В процессе
FR-03	Пользователь может изменить пароль	BR-02	UserProfile	change_pw.py	TC-04, TC-05	Выполнено
FR-04	Система блокирует аккаунт после 5 неудачных попыток	BR-03	SecurityService	lockout.py	TC-06	Не начато

Можно проследить покрытие тестами всех функциональных требований



Requirement Traceability Matrix

Req. ID	Test Case ID →	TC_1	TC_2	TC_3	TC_4	TC_5	TC_6	TC_7	TC_8	TC_9	TC_10	# Test Cases for respective Requirement
Req_1		✖		✖			✖					3
Req_2			✖			✖						2
Req_3				✖								1
Req_4					✖		✖					2
Req_5						✖		✖				2
Req_6							✖					1
Req_7						✖		✖				2
Req_8									✖			1
Req_9										✖		1
Req_10											✖	1

Использование матрица трассирования предоставляет массу плюсов:

- ✓ Обеспечивает **прослеживаемость**: от бизнес-целей до тестов.
- ✓ Помогает найти **пробелы в реализации** (например, нет тестов для требования).
- ✓ **Упрощает управление изменениями**: можно быстро отследить, что повлияет при изменении BR или SR.
- ✓ **Повышает уверенность** в полноте реализации и тестирования.
- ✓ Удобна для **аудита и валидации** требований.
- ✓ Используется в **QA, валидации, сертификации, управлении проектами**.

Отзыв с реального проекта

На проекте матрицы трассируемости стали использоваться не только нами, но и product-owner со стороны заказчика. Так они убеждались, что все требования есть и они корректны, и отслеживали с помощью матрицы, что уже реализовано. Матрицы позволили нам сделать процесс разработки и тестирования в какой-то степени более прозрачным.

Как отслеживать изменения в требованиях?

1. Контроль версий (Version Control)

- Каждое требование должно иметь версию (например, v1.0, v1.1).
- При любом изменении создается новая версия, а предыдущая сохраняется для истории.
- В системах вроде **Jama Connect**, **DOORS**, **Jira + Confluence**, можно видеть историю изменений.

2. Трассировка требований (Traceability)

- Если изменяется требование, то через матрицу трассировки можно отследить, что еще затронуто:
    - код,
    - тест-кейсы,
    - другие требования,
    - документация.

Как отслеживать изменения в требованиях?

3. Журнал изменений (Change Log)

- Ведение журнала: кто изменил, когда, зачем, что именно.
- Часто используется в виде комментариев или встроенных журналов в системах управления требованиями.

Пример записи:

```
Изменено: FR-01 (Добавлена проверка капчи при входе)
Дата: 23.04.2025
Автор: А.Петров
Причина: Требование по безопасности (ISO 27001)
```

Как отслеживать изменения в требованиях?

4. Процедура управления изменениями (Change Control Process)

Обычно включает:

- Инициирование изменения (запрос на изменение).
- Анализ влияния (Impact Analysis).
- Одобрение изменения (Change Approval Board).
- Внедрение и отслеживание.

5. Оповещения и Подписка

- Многие инструменты (Jira, Confluence, Jama) позволяют подписываться на изменения в требованиях.
- При любом обновлении пользователь получает email или уведомление в системе.

Пример журнала изменений требования

Требование: FR-05

Описание: *Пользователь должен иметь возможность сброса пароля через email*

Версия	Дата	Автор	Изменение	Причина
1.0	10.04.2025	Иванова М.	Первоначальное требование сформулировано.	Исходный список требований
1.1	18.04.2025	Петров А.	Добавлено: email должен быть подтверждён перед сбросом.	Соответствие стандарту безопасности ISO 27001
1.2	22.04.2025	Смирнов Д.	Уточнение: ссылка для сброса должна быть одноразовой и действовать 15 мин.	Обратная связь от отдела безопасности

Пример **Impact Analysis** (анализ влияния)

Изменение: *В версии 1.2 добавлено ограничение на срок действия ссылки (15 минут) и одноразовость.*

ВЫВОД: Изменение относительно небольшое, но влияет на:

- код (логика токена),
- тестирование (проверки на срок действия),
- интерфейс (уведомления),
- документацию.

Цель: Определить, какие элементы проекта затронуты этим изменением.

Объект проекта	Влияние	Действие
Функц. требование FR-05	Требует обновления текста	Изменить описание требования
Модуль backend: password_reset.py	Нужно добавить проверку срока действия и токена	Изменить реализацию
Тест-кейсы: TC-11, TC-12	Добавить новые кейсы на истечение времени, повтор	Обновить и добавить тест-кейсы
UI/UX дизайн: Reset form	Добавить визуальное сообщение о сроке действия	Обновить макет
Документация: User Guide	Обновить инструкцию по сбросу пароля	Внести правки в Confluence

Объект проекта	Тип влияния	Действие
Функциональное требование (FR-XX)	Изменение описания/логики	Обновить требование в системе управления
Backend-модуль (например, auth.py)	Необходима доработка логики обработки	Внести изменения в обработку сброса пароля
Frontend-компонент (например, ResetPassword.vue)	Обновление отображения или формы	Обновить компонент сброса пароля на клиенте
Тест-кейсы (например, TC-XX)	Создание/обновление проверок	Добавить тесты на одноразовость и срок действия
UI/UX-дизайн (макет формы)	Корректировка элементов интерфейса	Обновить макеты в Figma
Документация (в Confluence или PDF)	Обновление инструкций и справки	Внести правки в руководство пользователя
База данных (например, таблица токенов)	Возможные изменения структуры или хранения данных	Добавить или изменить поле expiry_time
DevOps/CI pipeline	Добавление/обновление автоматических проверок	Обновить тестовые скрипты и пайплайны

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

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.

5. 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 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:

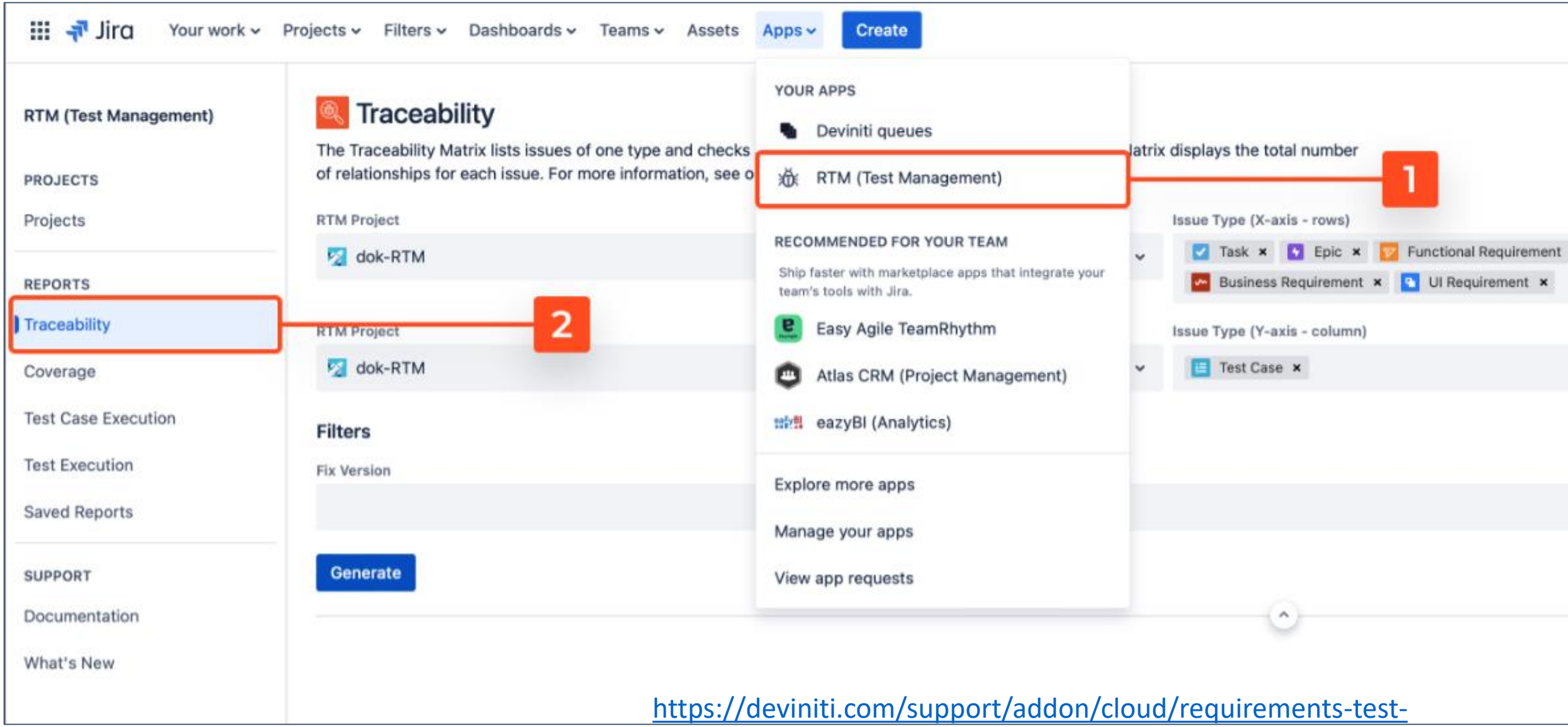
- дата создания;
- номер текущей версии;
- автор (создатель требования в СУТ);
- приоритет, который показывает его относительную важность для стейкхолдеров;
- статус, т.е. состояние жизненного цикла;
- происхождение или источник;
- логическое обоснование;
- контактное лицо, ответственное за принятие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

Отслеживание требований в Jira

Чтобы установить прослеживаемость между требованиями, Jira предоставляет несколько функций:

- ✓ **Связывание задач**: Jira позволяет пользователям создавать связи между разными задачами, обеспечивая прослеживаемость между связанными требованиями. Например, одно требование может быть связано с другим требованием, зависящим от него.
- ✓ **Иерархия задач**: Jira поддерживает иерархию задач, позволяя пользователям устанавливать отношения «родитель-потомок» между задачами. Эта функция особенно полезна, когда требования разбиваются на более мелкие и более управляемые части.
- ✓ **Зависимости:** Используя подключаемые модули или надстройки, команды могут определять зависимости между требованиями и управлять ими, гарантируя, что изменения одного требования должным образом отражаются в других требованиях

Процесс создания матрицы трассировки в Jira



https://deviniti.com/support/addon/cloud/requirements-test-management/latest/traceability/

RTM (Test Management)

PROJECTS
Projects

REPORTS
Traceability
Coverage
Test Case Execution
Test Execution
Saved Reports

SUPPORT
Documentation
What's New

Traceability

Settings

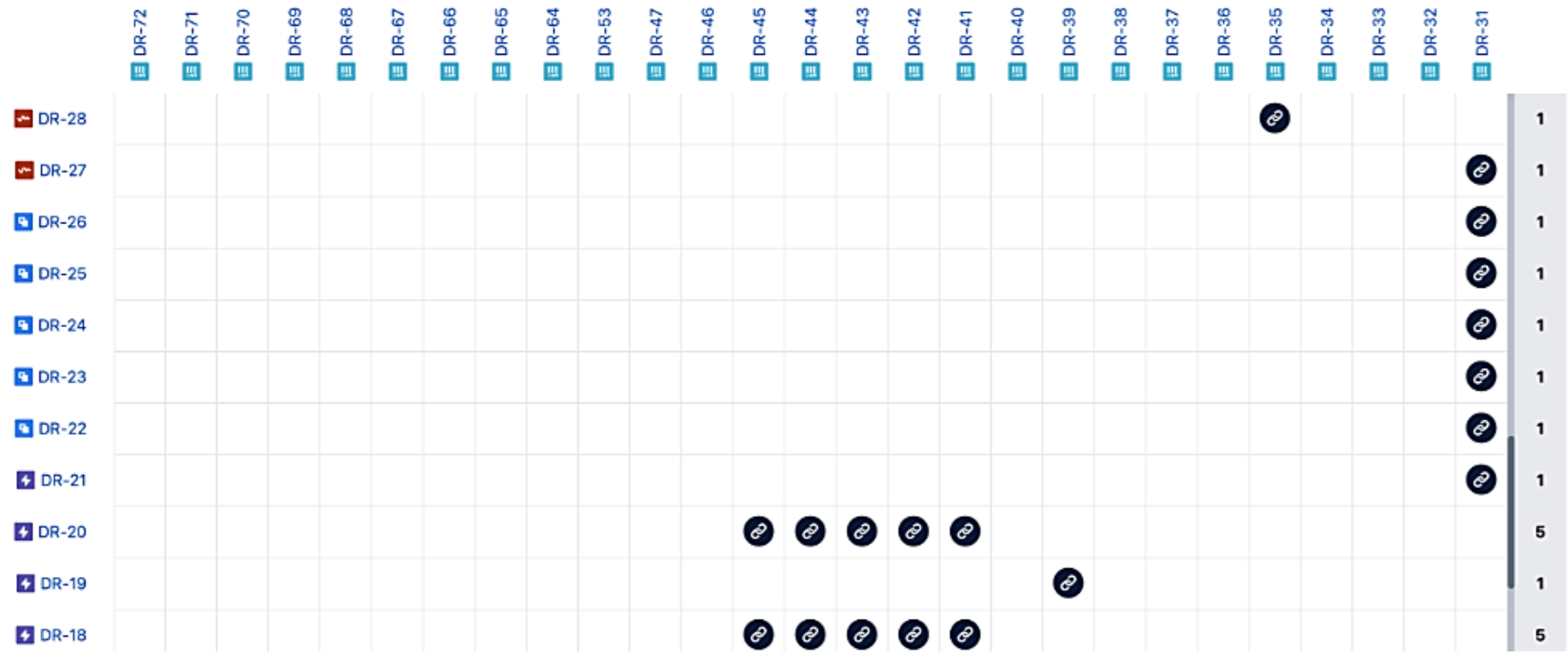
Traceability for DR

Creation date: Sep 5, 2024

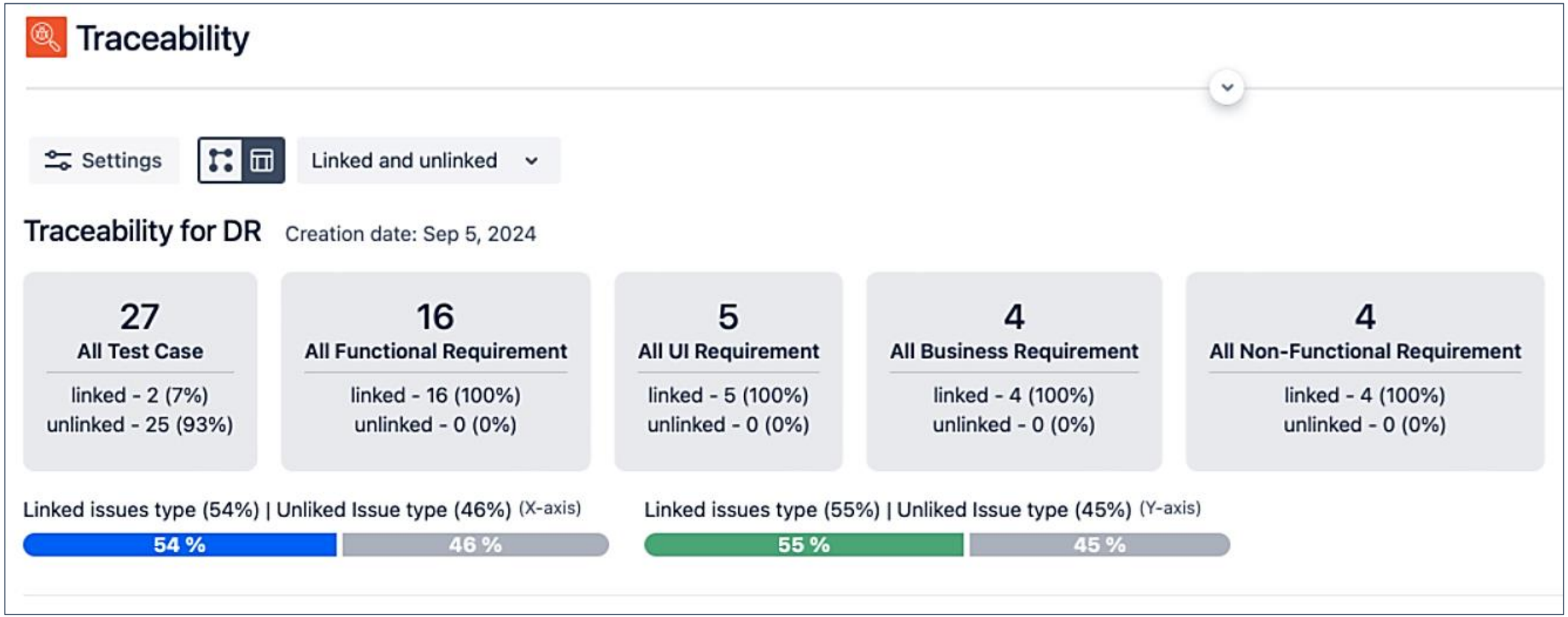
27	16	5	4	4	1
All Test Case	All Functional Requirement	All UI Requirement	All Business Requirement	All Non-Functional Requirement	All Task
linked - 2 (7%) unlinked - 25 (93%)	linked - 16 (100%) unlinked - 0 (0%)	linked - 5 (100%) unlinked - 0 (0%)	linked - 4 (100%) unlinked - 0 (0%)	linked - 4 (100%) unlinked - 0 (0%)	linked - 0 (0%) unlinked - 1 (100%)

Linked issues type (54%) | Unliked Issue type (46%) (X-axis)

Linked issues type (55%) | Unliked Issue type (45%) (Y-axis)



Отчет о прослеживаемости (трассировке) в Jira



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/