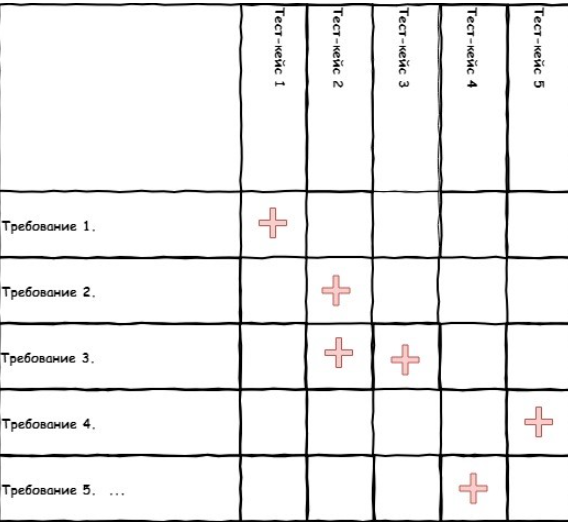
Когда требования на проекте меняются “на лету” и у вас нет под рукой средства контроля за реализацией каждого отдельного требования по фиче или модулю, перед вами встает вопрос: как проводить анализ покрытия?

**матрица трассируемости** — двумерная таблица, содержащая соответствие функциональных требований продукта (functional requirements) и подготовленных тестовых сценариев (test cases).  
  
На пересечении соответствующих строки и столбца ставится отметка, обозначающая, что данное требование покрывается данным тест-кейсом.  
  
Таким образом, таблица дает визуальное отображение двух параметров:

* наличие в системе требований, которые еще не покрыты (если у требования нет ни одного пересечения с тест-кейсами (достаточное условие);
* есть ли в системе избыточное тестирование — если требования имеет несколько пересечений (необходимое условие).



**Варианты связей в матрице трассируемости**

Привязка требования и тест-кейса может быть:

* 1 к 1 (атомарное требование, которое покрывается одним тест-кейсом, данный тест-кейс покрывает только это требование);
* 1 к n (требование, которое покрывается несколькими тест-кейсами, данные тест-кейсы покрывают только это требование);
* n к n (требование, которое покрывается несколькими тест-кейсами, данные тест-кейсы покрывают это и другие требования).

По поводу последнего пункта хочется отметить, что

* когда одно требование в матрице трассируемости покрывается несколькими тестами, это может говорить об избыточности тестирования. В таком случае надо проанализировать, насколько требование атомарно.
* если выполнением всех тест-кейсов мы обеспечиваем полноту покрытия, а сами тест-кейсы не дублируют друг друга — это не будет избыточным тестированием.

### Специфика оценки покрытия с помощью матриц трассируемости

Если для оценки покрытия мы используем метрику “отношение количества требований к количеству тестовых артефактов”, то связи в матрице должны быть “1 к 1”, а требования максимально декомпозирован

**Пример.** Имеем неатомарное требование: “Пользователь должен иметь возможность изменять и форматировать письмо в текстовом редакторе”. Одного тест-кейса явно будет недостаточно, но если в матрице будет прилинкован только один артефакт, визуально будет представление, что требование покрыто.

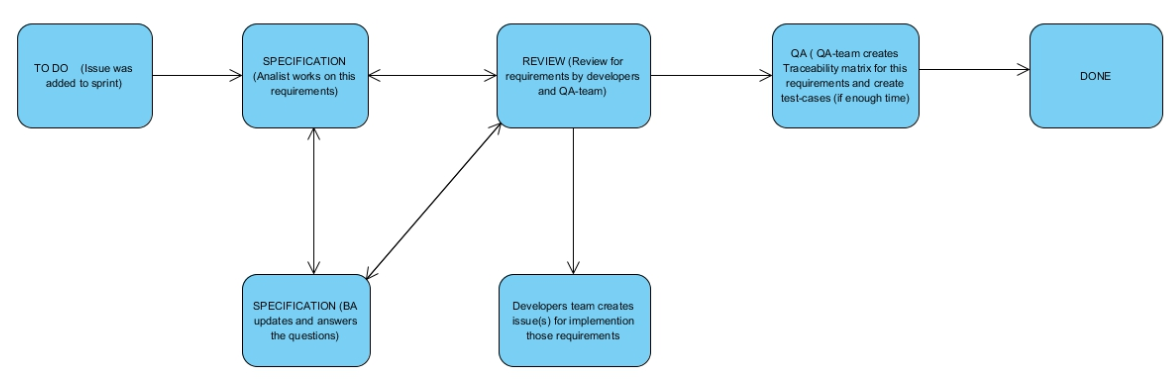
Поэтому лучше:

* разделить это требование в матрице на отдельные атомарные функции текстового редактора;
* для каждой функции написать приемочный критерий;
* для каждого критерия создать тестовый артефакт;
* если несколько атомарных требований могут быть покрыты одним чек-листом, можно не делать избыточного дробления, сэкономив ресурсы.

При нехватке ресурсов на максимальную декомпозицию можно использовать неатомарные требования, но на покрытие каждого их них нужно создать несколько тестовых артефактов.  
  
В таком случае, тест-кейсы и чек-листы для каждого неатомарного требования составляются единовременно, то есть каждое требование в матрице или полностью покрыто артефактами или не покрыто совсем.  
  
При составлении матриц желательно придерживаться рекомендации, что декомпозиция каждого требования в отдельно взятой матрице должна быть примерно равной, то есть в одной таблице не должны содержаться требования, часть из которых требует 5 тест-кейсов, а часть — один тест-кейс.  
  
Оценка покрытия в таком случае рассчитывается отдельно для каждой матрицы.  
Так как наша проектная документация может иметь различный вид для каждой фичи и даже описание одной фичи может содержать UML, схемы, диаграммы юз-кейсов и переходов, а проект содержит более 40 объемных функциональностей, мы решили разрабатывать отдельную матрицу для каждого модуля или фичи, чтобы не терять ни один из плюсов данного инструмента.  
  
Оценка покрытия также рассчитывается отдельно для каждого модуля или фичи.  
  
При таком подходе мы можем использовать метрику, описанную выше: “количество требований к количеству тестовых артефактов”. Даже если у нас есть связи 1 к n, n к n, у нас есть несколько компонентов, каждый из которых может быть использован в нескольких модулях. Требования и приемочные критерии описываются в каждой матрице, а тестовый артефакт используется один.  
  
Наши матрицы хранятся также в системе управления требованиями Confluence — каждая матрица расположена с структуре в качестве дочерней страницы фичи, для которой была разработана. Также все матрицы собраны на одной странице для удобства при оценке покрытия всего приложения.

**Создание и ведение матрицы**

Создание матрицы включено в наш воркфлоу работы над задачами по аналитике.

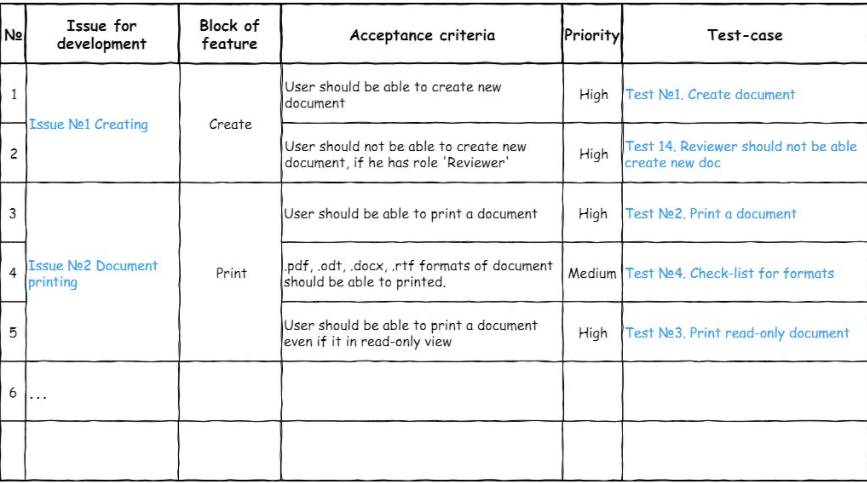


Когда мы получаем информацию о новой фиче, аналитик нашей команды создает задачу в таск трекере и совместно с product-owner со стороны заказчика работает в рамках этой задачи. В процессе сбора и структурирования требований вся команда проводит ревью и задает дополнительные вопросы. Когда требования сформулированы, задокументированы и подтверждены заказчиком, тим-лид разработки создает таски на разработку данной фичи, а команда тестирования может приступать к созданию матрицы трассировки.  
  
И здесь можно выделить следующие этапы составления Traceability Matrix:

1. В начале требования декомпозируются и подлежат приоритезации командой QA и\или product-owner. Результатом этапа становится структурированный и приоритезированный список всех требований по данной функциональности.
2. Вторым этапом будет общение с командой разработки и проставление задач из таск трекера на разработку в матрицу к соответствующим требованиям. В результате мы можем отследить трассируемость требований и задач на разработку.
3. Третий этап — разработка тест-кейсов и чек-листов.  
   Данный этап проводится или перед тестированием или во время тестирования конкретной задачи.  
   Если функциональность новая, и интерфейс будет изменяться, то могут быть кейсы, в которых шаги лучше описывать непосредственно перед началом тестирования задачи.  
   Если же функциональность по реализации схожа с одной из уже существующих фич, то мы может приступить к описанию тест-кейсов с шагами сразу после ревью и декомпозиции требований.
4. 4 этап — заполнение матрицы тест-кейсами.  
   По результатам всего процесса мы получаем задачи на разработку, тест-кейсы на тестирование и матрицу трассируемости, объединяющую их и требования.  
   Задача на разработку требований закрывается.
5. 5 этап — поддержка матрицы в актуальном состоянии. Изменения должны вноситься при любых модификациях требований. Также следует учитывать интеграционные связи между двумя матрицами, которые описывают разные фичи или модули, и при изменении в одной обязательно проверять, нет ли необходимости правки второй.

На нашем проекте мы используем матрицы трассируемости не только для оценки покрытия, но и для определения связи между задачами на разработку, требованиями и тестовыми артефактами.  
  
Поэтому матрица имеет вид таблицы, каждая строка которой содержит:

* номер и описание задачи на разработку из таск трекера;
* логический блок, к которому принадлежит задача (опционально);
* атомарное требование или приемочный критерий (acceptance criteria);
* приоритет;
* номер и описание соответствующего тестового артефакта.



Так как мы используем таск трекер Jira, Zephyr by Jira для тестовой документации и систему управления требованиями Сonfluence, все сущности синхронизируются и такая трассируемость позволяет нам:

* визуализировать актуальное состояние реализации;
* разбивать требования на более атомарные и структурировать их;
* отслеживать, есть ли требования, на которые еще не запланирована разработка (пропуск реализации);
* отслеживать, реализовано ли требование в данный момент;
* отслеживать, покрыто ли требование тест-кейсом (пропуск тестирования);
* наглядно отображать приоритезацию требований.

**Сложности в работе с матрицей трассируемости**

1. Актуализация  
   Матрица будет полезна только при условии, что она будет поддерживаться всегда в актуальном состоянии. На нашем проекте с часто меняющимися требованиями актуализация занимала много времени, но если матрицу не актуализировать, она становится не только бесполезной, но и вносит путаницу.  
     
   **Как решали**:  
     
   Частично решили проблему с частым изменение требований и перенесли этап создания матрицы на момент, когда требования уже просмотрены командой и подтверждены заказчиком.  
   Командой было принято решение, что аналитик будет актуализировать требования не только на самой странице с описанием фичи, но находить и актуализировать их в матрице, выделяя другим цветом. Это помогло всей команде не потерять изменения, а команде QA в частности — видеть какие тест-кейсы требуют актуализации.
2. Временные ресурсы  
   На проекте может быть срочный релиз и работа с новыми требованиями в одно и то же время, и все QA ресурсы направляются на тестирование, а не работу с требованиями. Таким образом, возрастает долг по тестовой документации.  
     
   **Как решали**:  
     
   Если все QA-специалисты заняты тестированием приоритетных задач, мы переносим создание матрицы по конкретной фиче. Максимально он переносится на момент тестирования первой задачи по этой фиче и в таком случае матрица заполняется тест-кейсами по мере тестирования задач, в которых реализована фича.  
     
   QA-специалист закладывает в оценку время не только на написание самих тест-кейсов, но и время на разработку матрицы.
3. Эффективность.  
     
   Если проект небольшой и все требования оформлены в виде структурированного ТЗ, а тест-кейсы создаются на каждое требование сразу, матрица трассируемости в нашем виде будет только дублировать информацию и будет лишней тратой ресурсов.  
     
   Поэтому нужно использовать стандартную матрицу, описанную в определении, для оценки покрытия.

### Приятности

* Матрица позволяет контролировать реализацию требований, отслеживать, что все требования разработаны и протестированы и ничего не пропущено.
* Матрица помогает команде QA отслеживать, есть ли долг по тестовой документации, и какие именно требования еще не покрыты тест-кейсами.
* Инструмент используется аналитиком и QA-командой для контроля измененных требований.
* 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.