**Тестовая документация**

**Создание тестовой документации является вторым этапом жизненного цикла ПО.**

Тестовая документация включает в себя:

* тест план;
* тестовая стратегия;
* чек-лист;
* тестовый сценарий;
* тестовый комплект;
* пользовательская история (User Story);
* отчет о дефекте.

**Тест план (Test Plan) -**это документ, описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.

**Хороший тест план должен описывать следующее:**

1. Что надо тестировать? Описание объекта тестирования: системы, приложения, оборудования.
2. Что будете тестировать? Список функций и описание тестируемой системы и её компонент в отдельности.
3. Как будете тестировать? Стратегия тестирования, а именно: виды тестирования и их применение по отношению к объекту тестирования.
4. Когда будете тестировать? Последовательность проведения работ: подготовка (Test Preparation), тестирование (Testing), анализ результатов (Test Result Analisys) в разрезе запланированных фаз разработки.

Критерии начала тестирования:

* готовность тестовой платформы (тестового стенда);
* законченность разработки требуемого функционала;
* наличие всей необходимой документации.

Критерии окончания тестирования - результаты тестирования удовлетворяют критериям качества продукта:

* требования к количеству открытых багов выполнены;
* выдержка определенного периода без изменения исходного кода приложения Code Freeze (CF);
* выдержка определенного периода без открытия новых багов Zero Bug Bounce (ZBB).

**Преимущества тест плана:**

1. Возможность приоритезации задач по тестированию.
2. Построение стратегии тестирования, согласованной со всей командой.
3. Возможность вести учет всех требуемых ресурсов (как технических, так и человеческих).
4. Планирование использования ресурсов на тестирование.
5. Просчет рисков, возможных при проведении тестирования.

Составляющей частью планирования тестирования (как отдельного документа или же процесса планирования в целом) является стратегия тестирования. Стратегия может быть:

1. Частью общего тест-плана.
2. Отдельным документом.

**Тестовая стратегия**определяет то, как мы тестируем продукт. Это набор мыслей и идей, которые направляют процесс тестирования.

В стратегии тестирования описывают:

1. Тестовую среду.
2. Анализ рисков проекта.
3. Инструменты, которые будут использовать, чтобы провести автоматизированное тестирование и для других целей.
4. План действий при непредвиденных обстоятельствах.

Стратегия может быть представлена как в виде традиционно расписанного документа, так и в более наглядном формате, например, используя таблицу:



В общем, план тестирования устанавливает цели процесса тестирования, он определяет, что будет проверяться, а стратегия тестирования описывает, как достичь целей, поставленных в плане тестирования.

**Пользовательские истории**

**Пользовательские истории** (User Story) — способ описания требований к разрабатываемой системе, сформулированных как одно или более предложений на повседневном или деловом языке пользователя. Пользовательские истории используются гибкими методологиями разработки программного обеспечения для спецификации требований.

**User Story**—  это короткая формулировка намерения, описывающая что-то, что система должна делать для пользователя.

Пользовательская история фиксирует короткую формулировку функции на карточке или, возможно, с помощью онлайн-средства.

**Примеры:**

* Залогиниться в мой портал мониторинга энергопотребления.
* Посмотреть ежедневный уровень потребления.
* Проверить мой текущий тариф.

Несмотря на то, что стори играют в огромной степени роль, ранее принадлежавшую спецификациям требований (SRS), сценариям использования и т. п., они все же ощутимо отличаются рядом тонких, но критических нюансов:

* Они не являются детальным описанием требований (то-есть того, что система должна бы делать), а представляют собой скорее обсуждаемое представление намерения (нужно сделать что-то вроде этого).
* Они являются короткими и легко читаемыми, понятными разработчикам, стейкхолдерам и пользователям.
* Они представляют собой небольшие инкременты ценной функциональности, которая может быть реализована в рамках нескольких дней или недель.
* Они относительно легко поддаются эстимированию, таким образом, усилия, необходимые для реализации, могут быть быстро определены.
* Они не занимают огромных, громоздких документов, а скорее организованы в списки, которые легче упорядочить и переупорядочить по ходу поступления новой информации.
* Они не детализированы в самом начале проекта, а уже более детально разрабатываются «точно в срок», избегая таким образом слишком ранней определенности, задержек в разработке, нагромождения требований и чрезмерно ограниченной формулировки решения.
* Они требуют минимум или вовсе не требуют сопровождения и могут быть безопасно отменены после имплементации.

**Структура юзер стори**

Текст самой юзер стори должен объяснять роль/действия юзера в системе, его потребность и профит, который юзер получит после того как история случится.

**К примеру: Как, <роль/персона юзера>, я <что-то хочу получить>, <с такой-то целью> .**

П**равила на написание User Story**

1. Есть один actor.
2. Есть одно действие.
3. Есть одна ценность / value / impact.

**Actor**

Вы выделили персоны, или у вас есть роли, и вы легко их вписываете в начало истории. Есть одна проблема. Убери часть истории про актера. Если история ничего при этом не потеряла - значит эта часть бесполезна.

Джеф Паттон предлагает следующее:

1. Разделите всех актеров на группы. Целевая группа, важная группа, менее важная группа и тп.
2. Дайте уникальные названия актерам в этих группах. Даже если в системе у них будет одинаковые роли "Пользователя системы".
3. Пишите истории с точки зрения этих актеров указывая их уникальные названия.
4. В результате вы сможете визуально увидеть какие истории необходимы для актеров целевой группы, какие - для каждой группы и тп. Вы не просто можете использовать это при разборе истории и выстраивания анализа вокруг указанного актера. Вы сможете более правильно выстроить приоритет, так как истории актеров целевой группы для нас более важны.

**Действие**

Это суть истории, "что нужно сделать". Что можно улучшить. Действие должно быть одно - основное. Нет смысла описывать" авторизуется и выполняется поиск" или "указывает параметры поиска и выполняет поиск". Укажите то действие, что вам действительно нужно.

Важно описывать историю на уровне "ЧТО?" делает, а не "КАК?". Это главное в истории. Опишите проблему, а не ее решение. Лучше вы потом с командой это обсудите и найдете более оптимальное "КАК" - решение.

**Ценность**

Главная проблема с User Story. Вы всегда знаете первую часть истории, но всегда сложно указать для чего это делается. Но это Scrum, все должно быть указано как User story согласно шаблону, и потому вы пишите "чтобы ...".

Уберите эту часть из истории. Если ничего не потеряли - значит формализация ценности в истории была бесполезна. Что же можно сделать?

Отказаться от формулировки "чтобы". Для каких-то историй можно указать ценность истории в таком формате, но не для большинства.

Перейти с понятия ценности (value) на влияние (impact). Ваша история не обязательна должна иметь ценность, но обязательно должна оказывать влияние на кого актера, что указан в истории. А уже это влияние ведет в конечном итоге к цели, которая имеет для вас ценность.

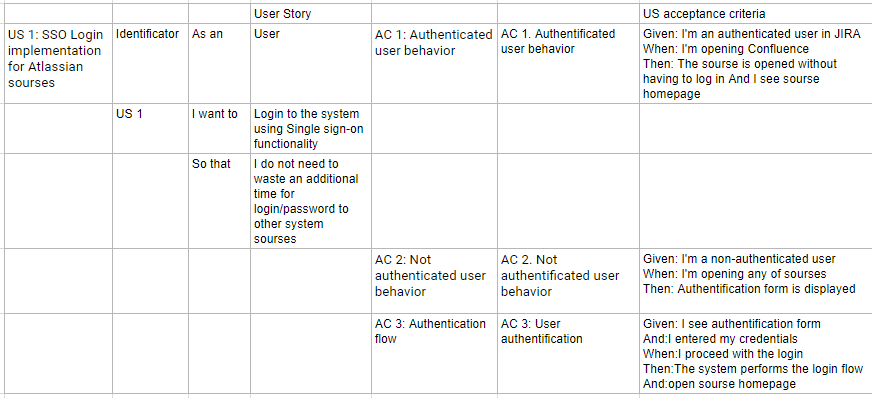
Представим что вы создали историю - "Как инвестиционный аналитик я получаю отчет №17 об инвестициях чтобы БЫСТРЕЕ принять решение".

У меня Аcceptance Сriteria - это метрика на value в US. Как померить такой value? Как понять что аналитик принял решение быстрее? Как вы поймете в конце что история выполнена?

Переделаем историю на влияние - "Как инвестиционный аналитик я получаю отчет №17 об инвестициях БЫСТРЕЕ". То есть сейчас этот отчет формируется за 60 сек. Вы указываете в АС что отчет должен формироваться за 15 сек. В конце понятно выполнено ли АС, понятно какие влияние вы оказали на работу аналитика.

**Практические советы по написанию пользовательских историй**

* Лучше написать много историй поменьше, чем несколько громоздких.
* Каждая история в идеале должна быть написана избегая технического жаргона — чтобы клиент мог приоритезировать истории и включать их в итерации.
* Истории должны быть написаны таким образом, чтобы их можно было протестировать
* Тесты должны быть написаны до кода.
* Как можно дольше стоит избегать UI. История должна выполняться без привязки к конкретным элементам.
* Каждая история должна содержать оценку.
* История должна иметь концовку — т.е. приводить к конкретному результату.
* История должна вмещаться в итерацию.

**Пример юзер стори:**продолжение:

