

Тестовая документация



План тестирования

(test plan) - документация, описывающая цели тестирования, которые должны быть достигнуты, средства и график их достижения, организованная для координации тестовой деятельности.



Test Plan



```
graph TD; A[Test Plan] --> B[Главный план тестирования (master test plan) - план тестирования, используемый для координации нескольких уровней или типов тестирования.]; A --> C[Уровневый план тестирования (level test plan) - план тестирования, обычно относящийся к одному уровню тестирования.];
```

Главный план тестирования (master test plan) - план тестирования, используемый для координации нескольких уровней или типов тестирования.

Базовый план для всего проекта, где хранится общая информация

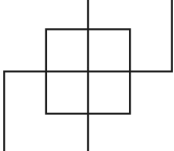
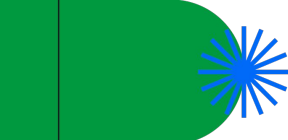
Уровневый план тестирования (level test plan) - план тестирования, обычно относящийся к одному уровню тестирования.

Создается либо для каждой команды, либо для каждой итерации



Структура тест-плана

1. Цель (purpose)
2. Области, подвергаемые тестированию (features to be tested)
3. Области, не подвергаемые тестированию (features not to be tested)
4. Тестовая стратегия (test strategy) и подходы (test approach)
5. Критерии (criteria). Чаще всего это критерии начала и завершения тестирования
6. Ресурсы (resources): программные, аппаратные, человеческие, финансовые
7. Расписание (test schedule)
8. Роли и ответственность (roles and responsibility)
9. Оценка рисков (risk evaluation)
10. Документация (documentation)
11. Метрики (metrics) - числовые характеристики показателей качества



Критерии начала и завершения тестирования

Entry criteria:

1. Выход билда для тестирования
2. 100% требований и мокапов утверждены и проверены

Exit criteria:

1. Выполнение более 80 % запланированных на итерацию тест-кейсов
2. Исправлено 100% критических багов
3. Автоматизированно 80% регрессионных тест-кейсов



Примеры тест-планов

Ознакомиться с примерами тест-планов можно [здесь](#)



Тестирование на основе чек-листов

(checklist-based testing) - техника тестирования на основе опыта, при которой тестовые сценарии разрабатываются для выполнения пунктов чек-листа.





Чек-лист



(checklist) - набор идей: идей по тестированию, идей по разработке, идей по планированию и управлению — любых идей.





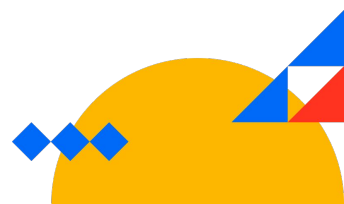
Обязательные части чек-листа

Шапка: содержит информацию о названии приложения, его версии, окружении, на котором проводится тестирование (версия ОС, браузера, эмулятора), ответственного за тестирование, дату тестирования

Тестируемые модули, субмодули: например, регистрация, аутентификация, авторизация

Список проверок: они должны отражать основную суть, без лишней детализации

Статус: информация о статусе прохождения проверки - пройдено/не пройдено (passed/failed)





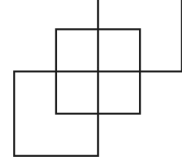
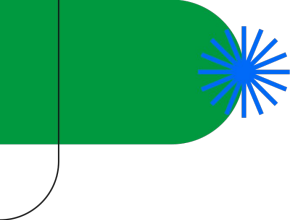
Дополнительные части чек-листа

Ожидаемый результат: то, что мы ожидаем увидеть после запуска проверки согласно требованиям

Типы тестирования: к какому типу относится проверка?

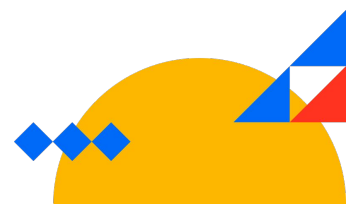
Отчеты о дефекте: ссылки на отчеты о дефектах для прослеживаемости

Заметки: если нужно добавить комментарии



Рекомендации

1. Если чек-лист используется для разных итераций, то можно создавать отдельную вкладку в таблице для каждой итерации.
2. Если необходимо проводить тестирования на разных платформах, то есть смысл добавить дополнительные столбцы.
3. Всегда начинайте проводить тестирование с позитивных кейсов и не объединяйте негативные проверки между собой.




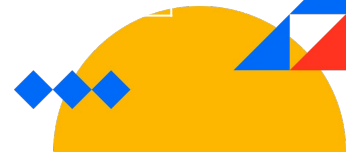


Чит-листы (cheat-sheet)

Существуют так называемые чит-листы. Обычно в них объединяют часто повторяющиеся проверки, например:

1. Общие проверки для веб и мобильных приложений
2. Список проверок для тестирования веб-форм
3. Список проверок для определенного типа тестирования, например, usability или локализация

Также в такие чит-листы иногда относят:

1. Список самых популярных команд в Git
 2. Список самых популярных команд в bash
 3. Список самых популярных операторов в SQL
- 
- 



Примеры чек-листов

Ознакомиться с примерами чек-листов можно [здесь](#)



Тестовый случай [сценарий]

(test case) - набор предусловий, входных данных, действий (где применимо), ожидаемых результатов и постусловий, разработанных на основе тестовых условий



“

”

Тестовый сценарий

(test scenario) - последовательность действий над продуктом, которые связаны единым ограниченным бизнес-процессом использования, и соответствующих им проверок корректности поведения продукта в ходе этих действий





Виды тест-кейсов

Тестовый сценарий высокого уровня (high-level test case) — тестовый сценарий с абстрактными предусловиями, входными данными, ожидаемыми результатами, постусловиями и действиями (там, где применимо)

Тестовый сценарий низкого уровня (low-level test case) — тестовый сценарий с конкретными значениями предварительных условий, входных данных, ожидаемых результатов и постусловий и подробным описанием действий (там, где применимо).



Атрибуты тест-кейса

Идентификатор (ID): уникальный номер, необходимый для прослеживаемости. В системах по управлению кейсами (TMS - test management system) проставляется автоматически

Приоритет (Priority): срочность и важность выполнения задачи

Требование (Requirement): ссылка на требование, для проверки которого служит кейс

Модуль (Module): название структурной части, в которой находится предмет тестирования

Заголовок (Title): Отражает суть проверки



Атрибуты тест-кейса

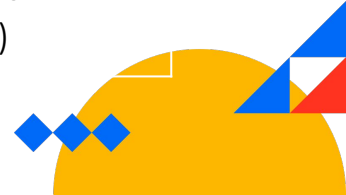

Тестовые данные и предусловия (input data, test data, preconditions):

информация о данных, которые необходимы для тестирования (данные для ввода, файлы с определенным расширением и размером и т.д.) + специальное состояние системы до начала тестирования (пользователь зарегистрирован, созданы объекты в базе данных и т.д.)

Шаги (Steps): последовательность действий для получения ожидаемого результата

Ожидаемые результаты (Expected results): ссылка на требование, для проверки которого служит кейс. Результат должен быть для каждого шага.

Постусловия (Postconditions): возвращение систему в исходное состояние (удаление данных, пользователей, отключение виртуальной машины и т.д.)



Жизненный цикл тест-кейса

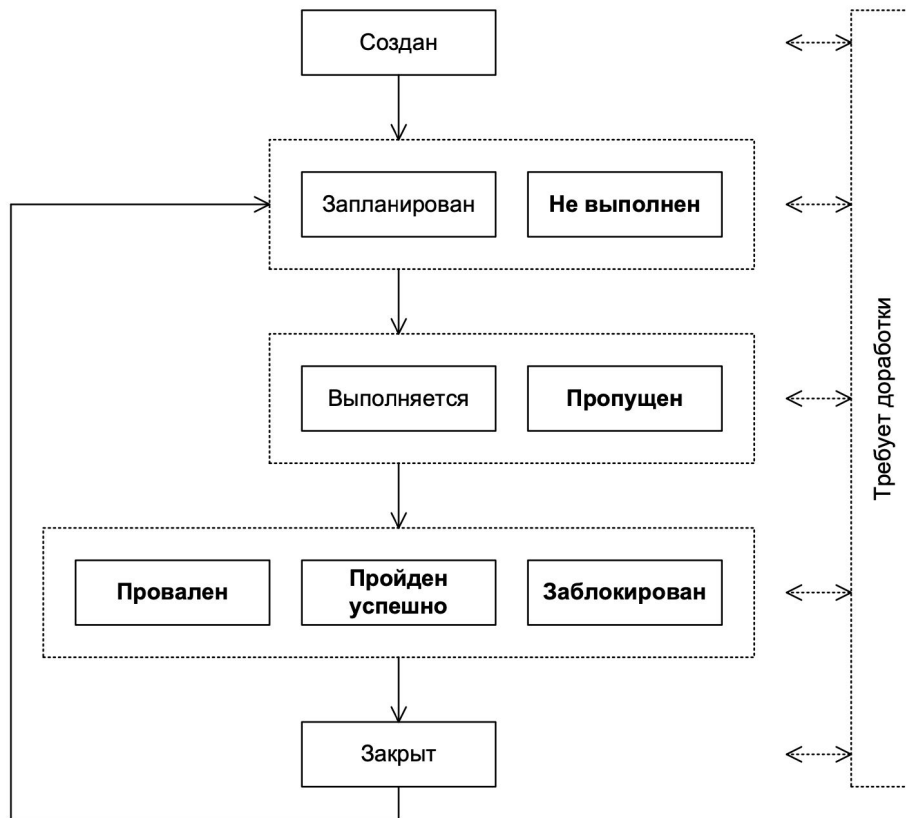


Рисунок 2.4.а — Жизненный цикл (набор состояний) тест-кейса

Святослав Куликов "Тестирование ПО. Базовый курс."



Статусы и примеры

В системах по управлению тестовой документацией есть атрибут **статус (Status)**.

Стоит различать статус кейса и статус проверки при их запуске во время тестирования (**тестовый прогон, test-run**).

Статус кейса: черновик (*draft*), активный (*active*), устарел (*outdated*) и др.

Статус проверки: пройдена (*passed*), провалилась (*failed*), пропущена (*skipped*) и др.
Ориентируйтесь на жизненный цикл тест-кейса на вашем проекте.

С примерами тест-кейсов можно ознакомиться [здесь](#)

Чек-лист vs тест-кейс

CL

1. Начальные этапы
2. Несложные проекты
3. Часто изменяющиеся требования
4. Требуют хорошего понимания системы
5. Легко поддерживать
6. Используются для повторяющихся проверок

ТС

1. Зрелый проект
2. Сложная бизнес-логика
3. Требования меняются не так часто
4. Подходят для обучения персонала
5. Сложно поддерживать
6. Активно используются для формирования регрессии

ВИД ДОКУМЕНТАЦИИ МОЖЕТ ЗАВИСЕТЬ ОТ ЗАКАЗЧИКА И УСЛОВИЙ КОНТРАКТА



Набор тестов

(test suite) - набор тестовых сценариев или тестовых процедур, выполняемых в определенном тестовом прогоне.



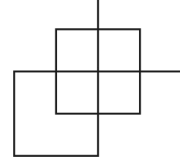


Тестовый сценарий

(test scenario, test procedure specification, test script) - документ, описывающий последовательность действий по выполнению теста

В версии 2023 не упоминается



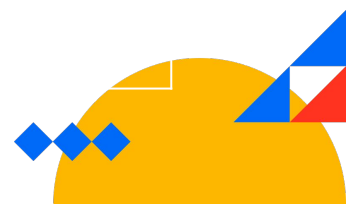


Пользовательский сценарий

Пользовательские сценарии (или сценарии использования), представляющие собой цепочки действий, выполняемых пользователем в определённой ситуации для достижения определённой цели.

Поясним это сначала на примере, не относящемся к «Конвертеру файлов». Допустим, пользователь хочет распечатать табличку на дверь кабинета с текстом «Идёт работа, не стучать!» Для этого ему нужно:

- 1) Запустить текстовый редактор.
- 2) Создать новый документ (если редактор не делает это самостоятельно).
- 3) Набрать в документе текст.
- 4) Отформатировать текст должным образом.
- 5) Отправить документ на печать.
- 6) Сохранить документ (спорно, но допустим).
- 7) Заккрыть текстовый редактор.





Отчет о дефекте

(defect report, bug report) -
документирование возникновения,
характера и состояния дефекта





Ошибка, дефект, отказ

Ошибка (error)

Действие человека, которое приводит к неправильному результату

Дефект (defect)

Несовершенство или недостаток рабочего продукта, проявляющееся в несоответствии требованиям или спецификациям

Отказ (failure)

Событие, при котором компонент или система не выполняют требуемую функцию в соответствии со спецификацией





Атрибуты отчета о дефекте

Идентификатор: уникальный номер отчета. Присваивается автоматически в багтрекинговой системе (BTS, bug tracking system)

Краткое описание (summary): отвечает на три вопроса: что, где, когда?
Что произошло? Где это произошло? При каких условиях?

Подробное описание (description): информация о дефекте в развернутом виде + фактический и ожидаемый результат + ссылка на требование

Шаги по воспроизведению (steps to reproduce, STR)

Окружение (environment) информация о среде, на которой был обнаружен баг (версия ОС, браузера, мобильного устройства)



Атрибуты отчета о дефекте

Важность (severity) - степень ущерба, который наносится проекту существованием дефекта. Серьезность для разрабатываемого ПО.

Критическая (critical) — существование дефекта приводит к масштабным последствиям катастрофического характера

Высокая (major) — существование дефекта приносит ощутимые неудобства многим пользователям в рамках их типичной деятельности

Средняя (medium) — существование дефекта слабо влияет на типичные сценарии работы пользователей, и/или существует обходной путь достижения цели

Низкая (minor) — существование дефекта редко обнаруживается незначительным процентом пользователей и (почти) не влияет на их работу



Атрибуты отчета о дефекте

Срочность (priority) показывает, как быстро дефект должен быть устранён.

Наивысшая (ASAP, as soon as possible) срочность указывает на необходимость устранить дефект настолько быстро, насколько это возможно.

Высокая (high) срочность означает, что дефект следует исправить вне очереди, т. к. его существование или уже объективно мешает работе, или начнёт создавать такие помехи в самом ближайшем будущем.

Обычная (normal) срочность означает, что дефект следует исправить в порядке общей очерёдности. Такое значение срочности получает большинство дефектов.

Низкая (low) срочность означает, что в обозримом будущем исправление данного дефекта не окажет существенного влияния на повышение качества продукта.



Атрибуты отчета о дефекте

Комментарий (comments, additional info) дополнительные данные о дефекте, возможность упомянуть исполнителя

Вложения (attachments) подтверждение дефекта: фото, видео, логи, тестовые данные, архивы и так далее

Также в дефектах обычно есть следующие поля:

1. Статус в жизненном цикле
2. Информация о создателе отчета и исполнителе
3. Ссылки на связанные артефакты
4. Привязка к спринту, отдельным модулям системы, пользовательским историям, тест-кейсам для прослеживаемости.

Жизненный цикл дефекта

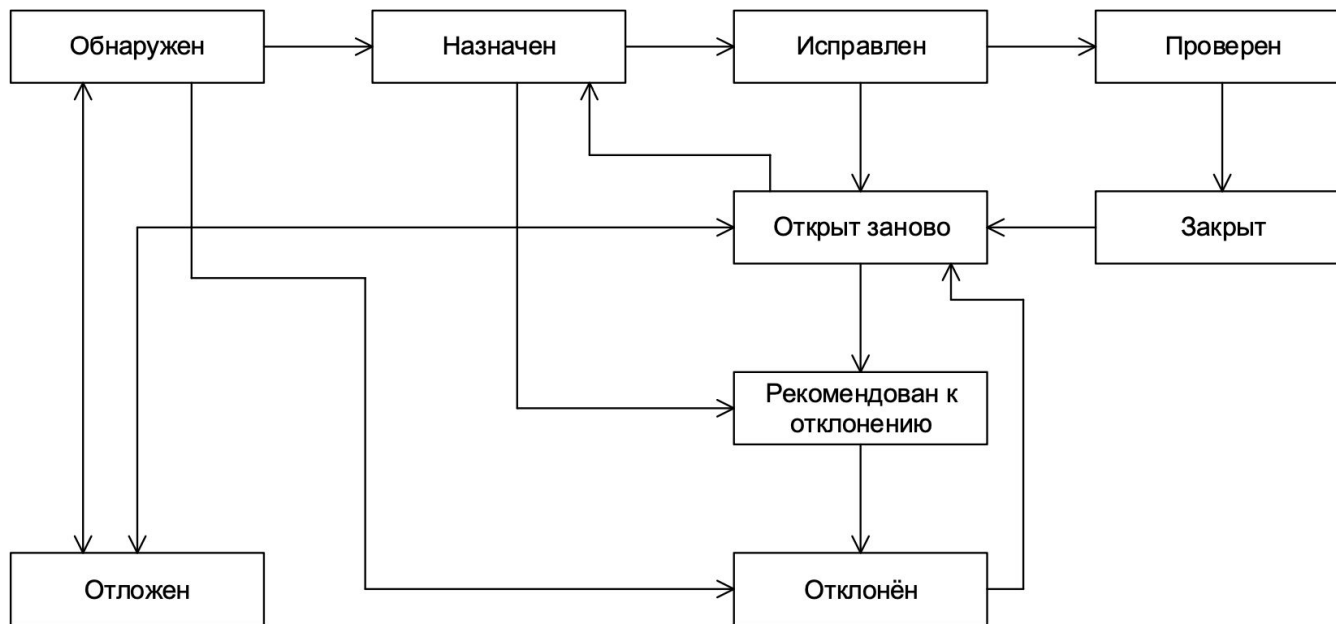
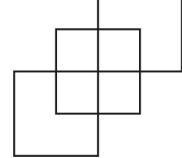
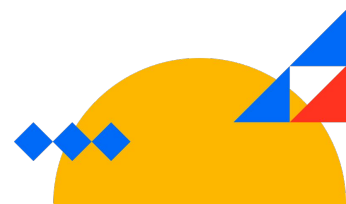
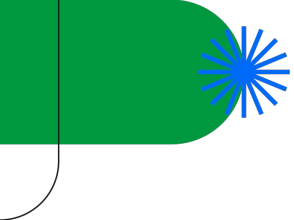
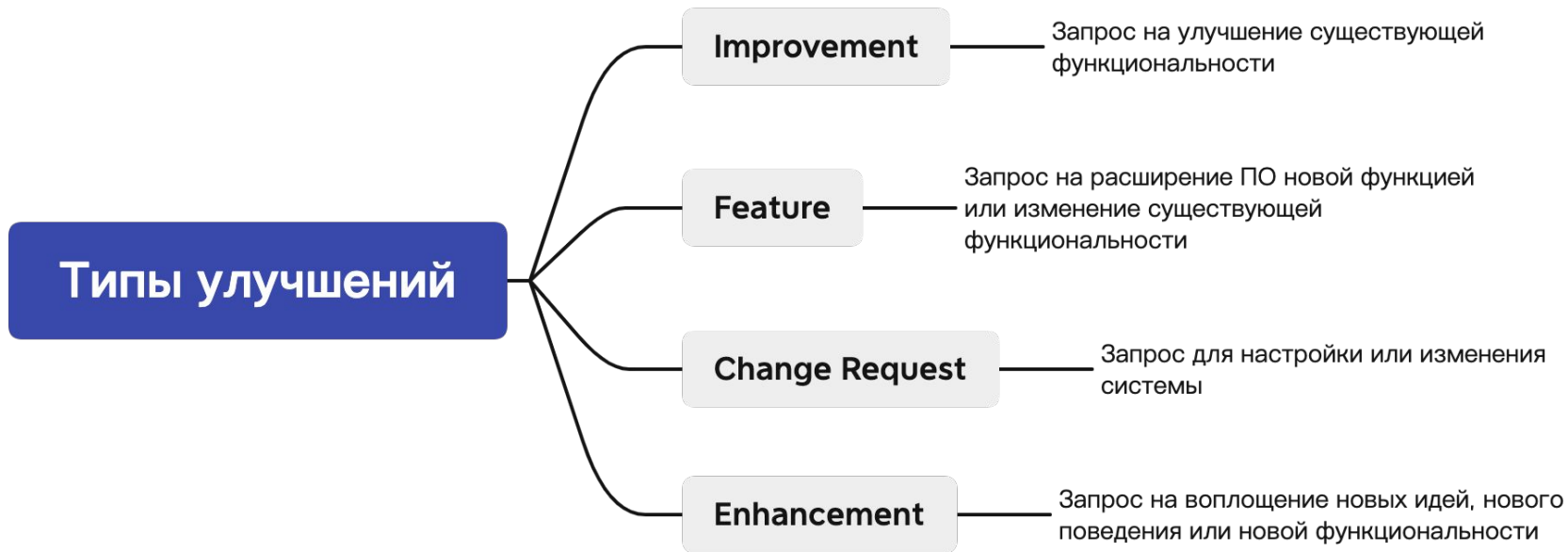


Рисунок 2.5.с — Жизненный цикл отчёта о дефекте с наиболее типичными переходами между состояниями



Типы улучшений





Советы по созданию вложений

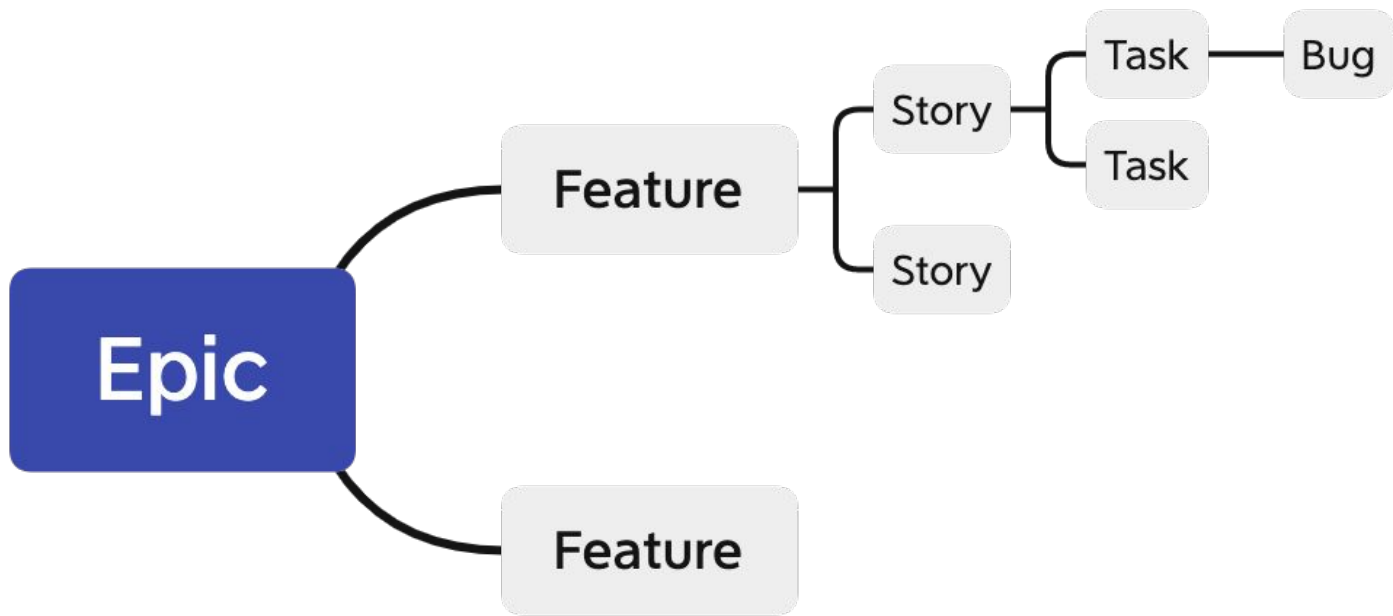
1. Используйте программы для записи экрана и фото: SnagIt, ShareX, Lightshot, Monosnap
2. Не рисуйте от руки, все уже есть в программе
3. Захватывайте часть экрана, по которой без проблем можно найти искомый объект
4. Локализируйте баг на скрине, используя стрелки, красные рамки, надписи
5. Называйте вложения либо заголовком самого отчета, либо его идентификатором
6. Отключайте звук при записи видео (если это только не звуковой баг)
7. Не показывайте на вложении личные данные
8. При тестировании веб на скрине должен присутствовать открытый DevTools
9. При тестировании мобильных приложений нужно прикладывать логи
10. Если для воспроизведения дефекта необходимы определенные тестовые файлы и данные, то их тоже стоит приложить
11. Сокращайте продолжительность видео и оставляйте только ключевые моменты



Реальная жизнь и советы

1. Несколько дефектов можно объединять между собой, если:
 - это визуальные баги, которые находятся в одном модуле
 - дефект воспроизводится одинаково для разных полей (например, одинаковая ошибка при валидации).

В таком случае можно указать тот факт, что дефект воспроизводится в нескольких местах как в описании, так и в комментариях.
2. Не все дефекты будут исправлены. Их приоритет зачастую определяет Project/Product Manager.
3. В случае накопления большого количества дефектов в бэклоге необходимо проводить Bug Triage (сортировка багов). Это собрание со всей командой с целью определения приоритетов по исправлению багов.
4. Обращайте внимание на наличие всех атрибутов. Грамотный отчет = счастливый разработчик :)
5. Все дефекты должны быть задокументированы



“

Отчет о ходе тестирования*

”

(test progress report) - тип периодического отчета о тестировании, который включает информацию о ходе активностей тестирования в сравнении с исходными планами, риски и альтернативы, требующие принятия решения.

**прим. автора: чаще всего его называют отчет по результатам тестирования.*





Атрибуты отчета о тестировании

Краткое описание (summary) отражает основные достижения, проблемы, выводы и рекомендации.

Команда тестировщиков (test team)

Описание процесса тестирования (testing process description) - перечень работ за конкретный период

Расписание (timetable) - детализированное расписание работы команды тестировщиков и/или личные расписания участников команды.

Статистика по новым дефектам (new defects statistics) в виде таблицы



Атрибуты отчета о тестировании

Список новых дефектов (new defects list)

Статистика по всем дефектам (overall defects statistics) - таблица, содержащая данные обо всех дефектах за время проекта

Рекомендации (recommendations) - перечень работ за конкретный период

Приложения (appendixes) - фактические данные (графики и диаграммы)

“

Матрица трассируемости

”

(traceability matrix, RTM) - двумерная таблица, описывающая связь двух сущностей (например, требований и тестовых сценариев). Таблица позволяет производить прямую и обратную трассировку от одной сущности к другой, обеспечивая таким образом возможность определения покрытия и оценки влияния предполагаемых изменений.





Пример матрицы трассируемости

	TC 1	TC 2	TC 3	TC 4
REQ 1		+		
REQ 2	+			
REQ 3		+	+	
REQ 4				+

Оценка трудозатрат

“

Оценка затрат на тестирование
(test estimation) – аппроксимация
объема трудозатрат, связанная с
различными аспектами
тестирования.

”





Популярные техники эстимации

1. Пальцем в небо (метод проб и ошибок)
2. Процентное отношение к разработке (например, на 1 тестировщика приходится 2-3 разработчика; значит на тестирование тратится в 2-3 раза меньше времени, чем на разработку)
3. Процентное распределение, при котором на все фазы SDLC выделяется определенный процент времени. Время выделенное на тестирование, разделяется на отдельные фазы STLC.
4. Эстимация, основанная на предыдущем опыте
5. Структурная декомпозиция, при которой крупные задачи разделяются на более мелкие, которые легко оценить
6. Эстимация по трем точкам

Дополнительную информацию по эстимации можно прочесть в ISTQB CTFL 2023.



Простой пример эстимации

Вводные данные: команда состоит из 3 человек, в регрессионном наборе 30 кейсов, на прохождение одного кейса тратится 10 минут, обычно 20% тестов находят новые баги и нужен ретест. Для простоты подсчета не будем учитывать время на написание документации и уточнение требований.

1. Определяем время на регрессионное тестирование: **$(30 * 10) / 3 = 100$ минут**
2. Определяем время на ретест: **$100 * 0.2 = 20$ минут**
3. Примерное время на тестирование **120 минут**

Обязательно нужно учесть риски: недоступность членов команды, проблемы с ПО, внештатное увеличение количества багов.

4. Заложим на это еще 10% времени. Итого общее время на тестирование: **132 минут**
- 
- 



Эстимация по трем точкам

Рассчитывается по следующей формуле:

$(O + (4 \times M) + E)/6$, где

О - оптимистичная оценка, когда все идет по плану

М - наиболее вероятная оценка, с учетом возникающих проблем

Е - пессимистичная оценка, когда все идет не по плану

Также рассчитывается стандартное отклонение: **$(E - O)/6$**

На нашем примере это будет выглядеть так: **$(100 + (4 \times 120) + 132)/6 = 118.7$ минут**

Финальный вариант: **118.7 ± 5.3 минут**





Реальная жизнь и советы

1. Сохраняйте результаты по времени, которое вы затратили на тестирование. Для этого есть отдельные поля в TMS и/или отдельная таблица, которую нужно завести
2. Эстимация включает в себя не только время, которые вы потратили на само тестирование, но и время на придумывание кейсов, их оформление, анализ требований, коммуникацию с командой
3. Оценка трудозатрат всего проекта складывается из работы всех участников команды, а не только отдела разработки. Об этом нужно напоминать
4. Всегда держите в голове риски и время, которое будет необходимо на повторное тестирование. Возвращаемся к п.1, так как аналитика, статистика и прогнозирование не может существовать без ваших данных
5. Полученную оценку нужно периодически обновлять
6. Будьте готовы к тому, что на незрелых проектах существует только одна оценка: пальцем в небо :)