Сегодня поговорим об основных документах тестировщика. Разберемся, чем тест план отличается от тестовой стратегии, что такое чек лист, тест кейс и как их составлять, где взять тест свит и зачем он нам.

ТЕСТ ПЛАН и ТЕСТ СТРАТЕГИЯ

Начнем с тест планов и тестовых стратегий именно потому, что эти два документа пишутся на этапе старта проекта.

Тестовая стратегия (по версии ISTQB) это документация, который описывает общие требования к тестированию и подробно описывает как проводить тестирование на уровне организации.

Тестовая стратегия это руководство, в котором рассматриваются цель тестирования, среда, подход к тестированию, инструменты, стратегии автоматизации, анализ рисков.

Кто составляет тестовую стратегию? QA + PM

Что включает:

- список лиц кто пишет, правит, проверяет и подписывает документы;

- описание тестовых активностей в разные отрезки времени

- Описание процесса тестирования

- порядок работы с задачами и ошибками (статусы, какой конечный, какой начальный, что понимаем под улучшением)

- типы тестирования (Load, Performance, Security, etc)

- Работа тестировщика и его зоны ответственности (какая степень компетенции требуется от тестировщика, и какая нагрузка по времени ожидается)

- Приоритеты в тестировании

- Описание тестовых окружений и тестовой базы данных

- тестовая документация на проекте? Что пишем, где храним, что поставляем

- testing tools: инструменты для работы авто и мануальных КУА,

- Review and Approvals подписи всех ключевых людей, согласования, обязательно храним историю изменений

ТЕСТ ПЛАН,

Определение в ISTQB говорит, что это подробный документ, описывающий объем, подход, ресурсы, график тестовых действий.

Виды:

- Мастер – высокоуровневый документ на проект, часто путают с тестовой стратегией
- Детальный Тест план – подробный документ, который пишется на весь проект или его отдельную итерацию, может меняться со временем

- План приемочных испытаний – документ, описывающий набор действия связанных с приемочным тестированием (стратегия, когда кто и как будет проводить работы)

Как создать?

Составление плана тестирования является важной задачей процесса управления тестирования, стандарт тестовой документации дает следующие рекомендации что надо сделать:

- Начните с анализа проекта (продукта) – Невозможно тестировать продукт без информации о нем. Кто будет пользоваться продуктом? Как? Для чего?
- Разработайте тестовую стратегию – высокоуровневый документ, в котором определите цели тестирования, усилия и затраты на тестирование; объем тестирования (что входит и что нет), типы тестирования (Юнит, Апи, Интеграционное, Системное и так далее), пропишите риски и пути их решений, кто будет тестировать и когда
- Определите цели тестирования: чтобы это сделать надо перечислить все функции ПО (функциональность, производительность, интерфейс) и определить цели на основе этих функций. Например: работает ли функционал должный образом без ошибок, интерфейс соответствует потребностям клиента, удобно ли пользоваться функциями, выдерживает ли система заявленные нагрузки.
- Определите критерии тестирования – когда можно начинать тестирование, когда приостанавливать (например 40 % тестов валятся) и когда заканчивать тестирование. Критерии выхода из тестирования нам особенно важны, потому что это дает понимание какое тестирование считать успешным и когда мы готовы перейти к следующему этапу разработки или сдачи проекта. Например – 95% всех критических тест кейзов прошли успешно, нет блокирующих багов, количество мажоров не больше 10.
- Запланируйте ресурсы – описание всех типов ресурсов необходимых для выполнения задач проекта. Сюда входят человеческие ресурсы с описанием зоны ответственности (например тест менеджер управляет проектом, определяет направления проекта; мануальный тестировщик пишет тест кейзы, прогоняет их, заполняет баг репорты, автоматизатор реализует автотесты на основе ручных, прогоняет их)
- Продумайте тестовые стенды (среды): что, сколько и когда надо?
- Проанализируйте расписание проекта и определите дедлайны каждого этапа: работа вместе с ПМ проекта, разбить проект на задачи, составить график что за чем выполняется, оценить сколько усилий потребуется для каждой задачи. Например: когда и в какой срок писать тест кейзы, когда, в какой срок делать регресс и сколько времени это займет.
- Определите какие тестовые документы будут поставляться – что мы поставляем до тестирования (тест план, тест кейз) что поставляем во время тестирования (тестовые данные, логи ошибок, отчеты о тест экзекьюте) и что поставляем после завершения тестирования (отчеты о тестах, дефектах, релиз ноутсы, гайдлайны)

После того как вы выполните всю эту работу, на руках у вас будет документ, включающий в себя все необходимые секции.

Есть 2 стандарта тест плана: ISTQB , RUP но оба они отвечают на одни и те же вопросы:

- Что тестируем (описание объекта, системы, продукта)
- Какие функции входят в тестирование и какие нет

- Как будем тестировать (стратегия, виды тестирования и их применение)
- Кто будет тестировать и какие зоны ответственности у каждого участника. Тест дизайнеры, тест экзекьютеры,
- Когда: критерии начала тестирования (разработка функционала закончена, готовность тестового стенда), приостановки и завершения: (требования к количеству открытых багов выполнены, определение периода без изменений кода, без открытия новых багов и прочее)
- Риски: документируем потенциальные риски в проекте и их решение, например слишком плотный график проекта, трудно вовремя завершить, надо прописать приоритеты фич, тестов; или недостаточная документация, нужно определить как кто будет прояснять возникающие в ходе разработки вопросы.
- Project Milestones: даты начала и окончания каждой фазы
- Рецензии и подписи всех ключевых участников (куа лид, пм, продукт оунер, заказчик ..)

В помощь есть 2 принятых шаблона тест плана - **IEEE 829** и **RUP** ссылки дам

http://www.protesting.ru/documentation/test_plan_template_rup.zip

http://www.protesting.ru/documentation/test_plan_template_ieee_829.zip

Давайте разберемся, в чем же основные отличия Тест Плана от Тестовой стратегии:

1. Тест план это подробный документ, в то время как стратегия высокоуровневая
2. Тест план существуем сам по себе, стратегия может быть частью тест плана
3. Тест план меняется вместе с проектом, стратегия остается постоянной
4. Тест план рассказывает про проект основываясь на спецификации, а стратегия рассказывает про основной подход в компании
5. Тест план пишется на уровне проекта, спринта, вида тестирования, стратегия на уровне организации и может переиспользоваться от проекта к проекту.

Можно ли обойтись без Тест плана, тестовой стратегии?

На эту тему много споров. Одни проекты обходятся без формального тест плана, а другие пишут подробнейшие документы на каждый спринт . В любом случае невозможно отрицать плюсы наличия тест плана:

1. Во первых, он помогает сделать процесс тестирования прозрачным и понятным для всей команды, включая заказчиков и бизнесс менеджеров.
2. Во вторых, он направляет наше мышление и помогает структурировать все знания о проекте и процессе его тестирования.
3. Основные важные моменты задокументированы и согласованы со всеми членами команды и заказчиком. А это значит, что на этапе сдачи проекта не возникнет вопросов почему чтото не написали, не протестировали, пропустили.

ТЕСТ КЕЙЗЫ

Что это такое?

Тест кейз – это **документ** с набором прекондишенов, действий , ожидаемым результатом и постусловиями созданный для **проверки** реализации тестируемой функции или ее части.

Иными словами, это проверка! Оформленная в виде структуры: действие – ожидаемый результат – тестовый (фактический) результат.

Структура тест кейса зависит от договоренностей внутри группы, которая их пишет (это может быть организация в целом или отдельно взятый проект).

У тест кейзов есть статусы – это результат проверки, может быть ПАСС, Фейл или блокед

По Ожидаемому результату Тест кейсы принято делить на **позитивные и негативные**.

Позитивные тест кейсы используют только корректные данные и проверяет что вызванная функция работает так, как и ожидается, что система в целом работает так как и ожидается. Негативные кейсы оперирует невалидными данными чтобы проверить исключительные ситуации и срабатывание валидаторов. Негативные кейзы изначально построены так чтобы проверить что система не работает там где и не должна, да еще и показывает пользователю понятные сообщения.

Но вообще четкой границы между позитивными и негативными нет. К негативным тест кейзам часто относятся нетипичное использование приложения, работа в невозможных конфигурациях, потерю коннекта, что будет если будут эксепшены в процессе выполнения запросов и прочее.

Важно запомнить, что сначала выполняем позитивные проверки а потом уже негативные.

Нам будет не важно, как обрабатывается SQL инъекция если не работают покупки в интернет магазине.

Например:

Тестируем работу с корзиной в интеренет магазине.

Позитивные кейзы:

- добавить товар, удалить товар, добавить товар снова.

Негативные:

- удалить удаленный товар, открыть 2 вкладки и удалить товар дважды. Насколько можно назвать такие ситуации негативными?

Основные поля тест кейса.

Обязательные поля, которые есть во всех тест кейсах, не важно где вы их храните:

- ID (авто) – уникальный номер кейса

- Тайтл – заголовок, старайтесь делать его коротким но максимально отражающим суть проверки

- Приоритет – выставляем в ручную, основываясь на приоритете функциональности, и вероятности использования сценария

- Тест скоуп – краткое описание что это за тест кейс, что он проверяет. Назначение проверки. Например: В этом тест кейзе мы проверяем добавление товара в корзину.

- Прекондишены: описание всех действий которые надо сделать до начала выполнения теста. Например, привести систему в какоето состояние, включить-выключить какието настройки, создать определенные тестовые данные и прочее

- Тест степы – шаги по выполнению самой проверки

- Тестовые данные – примеры данных, которые могут использоваться для прохождения теста, юзеры с паролями, файлы для загрузки, зип коды, запросы в базу данных, и прочее

- ЕР – что мы ожидаем в конце нашей проверки, например что письмо отправилось тестовым юзерам, продукт попал в корзину и прочее

Дополнительные поля: если используется тула, то большая часть будет запоняться автоматически

- Автор – кто создал
- Лейблы, tags, категории - помогают потом искать тесты по ключевым словам
- Авто статус – если есть автоматизация в проекте, позволяет отследить был ли тест заавтомейчен или нет
- Информация по прогонам теста, заполняется автоматически, позволяет отслеживать когда кем гоняли, какой был статус, били ли падения и на каком степе
- Статус текущий
- Ревью – было ли проведено ревью теста, очень полезная кстати говоря активность на проекте. Ревью тестов помогает отследить ошибки копипасты, ошибки в понимании самого тестировщика и плюс позволяет команде быть в курсе над какими задачами работает коллега.
- Посткондишены – приведение системы в то же состояние, которое было до начала прогона теста. Это особенно актуально если в прекондишенах вы включали или меняли какие то сеттинги на общих ресурсах. Иными словами это правило хорошего тона, убрать за собой все следы.
- Комменты – тут может быть все что угодно, история изменений, что и на основании чего менялось.

Как написать хороший тест кейс.

- На каждый кейс только одна цель проверки. **Не надо** писать кейс, который проверяет всю фичу, всю страницу, или содержит в себе позитивные и негативные проверки одной формы.

- Каждый кейс должен быть завершенным
- Кейс должен быть воспроизводимым и при каждом прогоне приводить к одному и тому же определенному результату
- Кейз должен быть простым, понятным любому члену команды, понятным новому человеку без опыта, и не длинным. Прогонять кейз из 40 шагов это мука, ребята.
- Кейз не должен содержать в себе предположения о том, как поведет себя система, все ожидаемые результаты должны быть обоснованы не вашим опытом и предположением, а спецификацией или любой другой докой. Если вы в процессе написания тест кейса понимаете, что не знаете точно, какой должен быть ожидаемый результат то надо идти выяснять.
- Каждый тест кейс должен содержать в себе всю необходимую информацию для прогона, включая тестовые данные, скрипты, файлы загрузки, тулы (мало ли)
- Тесты кейзы не должны содержать в своих шагах линки на другие тест кейсы.
- Избегайте повелительного наклонения, все должно быть написано в нейтральном стиле. Это чисто негласный хороший тон написания тест кейсов.

Примеры

Плюсы и Минусы использования тест кейсов

Плюсы

- Хорошее тестовое покрытие
- При написании используются все возможные тестовые техники, и это дает нам больше уверенности в оптимальном покрытии функционала
- Снижение затрат на поддержку продукта, так как все поведение описано тест кейсами.
- Переиспользование тестов, каждый тест кейс может гоняться множество раз, быть в разных наборах от смоука до регресса.
- Наличие тест кейсов и результатов их прогонов подтверждает, что ПО отвечает потребностям пользователя.
- Легко автоматизировать
- Легко прогонять любому члену команды или новичку, незнакомому с продуктом.
- Использование темплита, стандарта понятного всей команде от которого нельзя отступать

Но есть и минусы:

- Не спасают тестировщиков от парадокса пестицида
- Не гибкие, потому что проверки прописаны по шагам и отступать нельзя
- Увеличивают расходы на поддержку самих тестов
- Могут содержать в себе ошибки копипаста, все мы люди. Чтобы минимизировать этот риск, надо обязательно проводить ревью тест кейсов
- Большое количество документации, иногда это плохо, потому что в ней можно потонуть
- Если нет темплита, или он не соблюдается, то тесты могут отличаться друг от друга и требовать больше времени при прогоне.

Коллекции тест кейзов называются тест Свитами.

Иногда их путают с тест планом, но это совершенно разные вещи.

Тестовый набор можно создавать для проверки одной функции, задачи, бага, всей системы или одного элемента, для смоука, сайнити и регресса. Кейсы можно объединять по приоритетам или позитивности – негативности.
Иными словами, тест план включает в себя тест свиты, которые включают в себя тест кейсы + тестовые данные + результаты прогонов + тестовые скрипты (если есть авто)

Тулы для работы с тест кейзами

Тест кейзы надо гдето хранить, использовать, поддерживать (обновлять), переиспользовать, автоматизировать, хранить статистику о результатах прохождения (кто, когда гонял и что получил), знать были ли найдены баги в результате прогона и в каком они сейчас состоянии. Для облегчения жизни тестировщиков изобретаются все новые и новые тест тулы. Благодаря использованию тулов вместе с каждым тест кейзом хранится тонны информации, которая заполняется автоматически.

Большинство тулов позволяют создать и использовать единый темплит для написания кейсов, объединять тест кейзы в наборы (Сайклы, тест свиты), автоматически строят графики прохождения и собирают различные метирики для анализа (от времени прохождения каждого кейза да пасс-фейл графиков). Так же важной является возможность легко заводить баг в режиме прохождения теста и привязывать тесты, баги, сторики между собой.

ЧЕК ЛИСТЫ

Что это

Это список проверок для любого вида работы. Обычно он не содержит в себе деталей и отвечает на вопрос что должно быть протестировано без пояснений Как это сделать.

Это удобный инструмент в тестировании, потому что помогает экономить время на создании и поддерживании, адаптируется под любые нужды проекта и может содержать в себе огромное количество информации.

Очень часто на проектах пишут расширенные чек листы, которые заменяют тест кейсы, в них добавляют степы, ожидаемые результаты и историю всех прогонов. Выполненые строчки помечают Пасс, Фейл или Нот Ран.

Отдельно надо сказать про существование чеклистов по тестированию типов полей. В сети можно найти чек лист по тестированию емейла, текстового поля и так далее. Это очень полезные списки, пригодятся в работе, даже когда ты думаешь что уже все знаешь.
В целом никто не запрещает создавать и хранить такие типы чеклистов, которые могут применяться в каждом проекте.

Плюсы и минусы

- Нет эффекта пестицида, так как есть проверка но нет шагов и каждый тестировщик будет выполнять ее по разному
- Чек листы составлять и поддерживать быстрее чем тест кейсы. Поэтому очень часто в небольших проектах или проектах с простой логикой выбирают именно чек листы.
- Гибкие
- Все возможные проверки на одном листе, в пределах поля зрения, меньше вероятности что чтото будет пропущено
- Могут использоваться несколько раз, даже в разных проектах (помним про чек лист тестирования емейла)
- В листах можно хранить всю историю прогонов, багов, оставлять комментарии, это делает процесс тестирования прозрачным.

Минусы

- Невозможно автоматизировать
- Для тестирования по чек листу нужны знания продукта, функциональности и это не подходит для новых людей в команде и людей без опыта.
- Неопределенность тестовых данных оставляет больше вероятности сделать проверку неверно, или недопроверить чтото. Плюс тратится время на то что бы понять, как подготовить тестовые данные, а иногда хочется взять и сделать.
- Нет дератизации, часто это так же минус, потому что есть вероятность что твое понимание не совпадает с тем, что задумывалось, так нельзя гарантировать качество тестирования.

Как же создать хороший чек лист

- Основное правило все то же что и для тест кейсов: одна проверка на одну строчку
- При написании чек листа опираться на требования
- Комбинировать проверки в блоки по фичам, категориям
- Внутри каждой категории сначала позитивные проверки, потом негативные. Не забывайте о негативных проверках, они так же очень важны для качественного продукта.
- Договориться о стандартах внутри команды, чтобы использовать одинаковый язык, инструмент, набор полей и степень детализации.
- Детализация в зависимости от задачи, иногда достаточно просто перечислить проверки, а когда то стоит расписать шаги, ЕР, тестовые данные.
- Собирайте чек листы в таблицы, где могут храниться платформы, версии, результаты проверок, и прочее. Это уже не совсем чек лист, это компромисс между листом и тест кейзом.
- Если нужно, добавляете к проверкам замечания, информацию по тестовым данным, это облегчит прохождение проверки в следующий раз.

Ну и когда же используются чек листы

Чек листы не являются заменой тесткейзам, практически нет проектов, где можно обойтись без чек листов, потому что это простой и удобный инструмент как для самопроверки так и для проверок проекта.

Вот некоторые варианты когда мы можем их внедрить и использовать:

- Небольшие проекты, недолгие по времени проекты. В таком случае лучше использовать чек-листы вместо тест кейсов, но делать их расширенными со степами и ЭР
- Кросс браузерное и Кросс платформенное тестирование
- Различные смоуки, возможно набора тест кейсов не хватит, потому что надо проверять чтото что не покрыто кейсами, настройки энвайромента, логи в базе, и прочее. Чек лист поможет не только определить список проверок, но и упорядочить его и разделить по специалистам, оценить мануал и авто покрытие.
- Эд-Хок тестирование – это когда мы тестируем основываясь на свой опыт, тут подойдут как раз общие чек листы для тестирования поля со всевозможными списком проверок
- Тестирование интерфейса и прочие нефункциональные проверки.

Шаблоны документов

http://www.protesting.ru/testing/templates.html