**Разработка тестовой документации. Поиск, анализ и документирование дефектов**

На некоторых проектах нет задокументированных требований, и тогда зачастую поддержка тестовой документации является единственным разумным способом хранения и передачи знаний о продукте.

Создание тестовой документации значительно улучшает качество продукта за счет более тесного сотрудничества, уточнению деталей при разработке плана тестирования и документации. После завершения тестирования наличие тестовой документации позволяет проверить, насколько успешно были проведены все этапы тестирования. В зависимости от конкретного проекта, тестовая документация может отличаться по формату, степени детализации и охвату. Главная задача тестовой документации — сделать процесс выполнения работы прозрачным для заказчика и улучшить качество тестирования ПО.

**Чек-листы vs Тест-кейсы**

Чек-лист отличается от тест-кейса степенью подробности. В чек-листе нет подробных шагов кейса, для использования чек-листа при тестировании очень много информации нужно держать в голове в момент прогона тестов и знать логику работы приложения на отлично.

|  |  |
| --- | --- |
| **+**   * непрерывное выполнение его тестировщиком: все кейсы расположены в порядке, удобном для прогона, и времени на переход к следующему кейсу не тратится * покрывает большое количество пользовательских сценариев * проверяет восстановление после ошибок и прерываний, что в случае мобильных приложений очень важно * длинные сессии, что повышает вероятность обнаружения утечек памяти и навигационных проблем * информация о требованиях поступает тестировщику последовательно, что дает лучшее понимание логики работы приложения | +   * может хранить историю тестирования и отображать, что именно и как тестировалось: что та или иная функциональность обязательно была или будет проверена и затронута при тестировании. * помогает быстро включить в работу новых сотрудников: достаточно открыть сохраненный ТК и пройти его по шагам так же, как проходил другой опытный специалист, работавший в компании. * помогает увидеть как предмет разработки (программа, сайт) должен выглядеть. С имеющимися скриншотами экранов, если они есть, можно будет не забыть о том, что “вон та” кнопка должна быть серой, а не красной. |
| -   * подразумевает декомпозицию по экранам приложения и сильно завязан на дизайн * трудности с поддержкой: кейсов в одном чек-листе может быть до двух сотен * невозможность заменить требования для разработчиков | -   * тестировщик "проверяет", а не тестирует. * следуя сценарию, упускаются важные проблемы * бездумная прогонка * большие затраты времени * часто устаревают, и их надо постоянно обновлять и поддерживать |

**Чек-листы можно записать:**

* в виде таблиц (удобно в [Google Sheets](https://www.google.com/sheets/about/))
* в виде интеллект-карт (удобно в [XMind](http://www.xmind.net/))
* в виде списка проверок в специально предназначенной системе (удобно в [Sitechco](http://sitechco.ru/))
* в виде списка в текстовом документе, который привычен.

**Тест-кейсы можно отобразить:**

* в виде таблицы с текстовыми данными
* в специальном сервисе для ведения тест-кейсов (например, в [TestLink](http://testlink.org/)).

**Дефект (баг bug, ошибка)** – отклонение фактического результата от ожиданий пользователя, сформированных на основе требований, спецификаций, иной документации или опыта и здравого смысла.

**Любой дефект** независимо от используемой системы управления дефектами имеет 5 основных характеристик: 1. Headline/Summary — суть проблемы. 2. Severity — степень критичности (серьезности). 3. Description — алгоритм воспроизведения. 4. Expected result — ожидаемое поведение. 5. Attachment — прикреплённые файлы.

Headline/Summary — суть проблемы. Цель составления заголовка дефекта — предоставить максимально понятную информацию о том, что, где и в результате чего произошло и насколько эта проблема критична для проекта.

Severity — степень критичности (серьезности). Показывает степень ущерба, который наносится проекту существованием дефекта. Градации важности: критическая, высокая, средняя, низкая.

Description — алгоритм воспроизведения дефекта. Описывает шаги для повторения дефекта.

Expected result — ожидаемое поведение. Показывает разработчикам, как именно должен быть исправлен дефект.

Attachment — вспомогательное средство передачи информации о проблеме. Это прикрепленный к дефекту файл, дополняющий описание: скриншот, файлы, необходимые для воспроизведения дефекта, логи программы, видео ошибки.

**Жизненный цикл дефектов**

Жизненный цикл дефекта (Defect Lifecycle) – это последовательность этапов, которые проходит дефект на своём пути с момента его создания до окончательного закрытия. Для простоты восприятия изображается в виде схемы с возможными статусами и действиями, которые приводят к смене этих статусов.

Любой этап работы с дефектом обозначается атрибутом «Статус». Статус дефекта показывает, кто на данный момент работает с дефектом и что следует делать с ним. После того, как работу с багом закончил один из участников проекта, он меняет статус дефекта и «перенаправляет» дефект на того, кто должен продолжить работу.

Зачастую жизненный цикл бага связан с выбранной системой баг-трекинга, в которой по умолчанию могут быть настроены статусы и переходы.

Новый (New) ‒ отчет о дефекте заводится в баг-трекинговую систему в первый раз.

Назначен (Assigned) ‒ отчет о дефекте назначается на соответствующего разработчика.

Открыт (Open) ‒ разработчик берет отчет о дефекте в работу для анализа и исправления.

Исправлен (Fixed) ‒ разработчик сделал необходимые изменения в коде и проверил эти изменения сам. Отчет о дефекте с этим статусом возвращается обратно тестировщику.

Повторное тестирование (Re-testing) ‒  тестировщик выполняет повторное тестирование измененного кода, который был предоставлен разработчиком, для проверки, исправлен ли дефект или нет.

Проверен (Verified) ‒ если дефект не воспроизводится, тестировщик подтверждает, что этот дефект исправлен.

Переоткрыт (Reopened) ‒ если дефект все же воспроизводится, даже после его исправления разработчиком, тестировщик переоткрывает его и назначает на разработчика. Этот дефект проходит через жизненный цикл дефекта еще раз.

Закрыт (Closed) ‒ если тестировщик уверен, что дефект больше не воспроизводится, то он его закрывает. Этот статус означает, что дефект исправлен, протестирован и одобрен.

Дубликат (Duplicate) ‒ если дефект повторяется дважды или есть два бага, которые являются следствием одной причины, то одному из них присваивается данный статус.

Отклонен (Rejected) ‒ если разработчик считает, что этот дефект не является обоснованным или веским, и дефект не будет рассматриваться для исправления или реализации, он его отклоняет.

Отсрочен (Deferred) ‒ ожидается, что дефект, которому присвоили такой статус, будет исправлен в следующих версиях. Причин для присвоения этого статуса может быть несколько: приоритет дефекта низкий, данный дефект не повлечет больших сбоев в программном продукте.

Не баг (Not a bug) ‒ этот статус присваивается, если в функционал приложения не будет внесено никаких изменений. Например, если заказчик просит изменить цвет или размер кнопок, или текста – это не дефект, а просто изменения в дизайне приложения.

**У каждого дефекта** (несоответствие между реальным и ожидаемым поведением системы) есть атрибуты: «Серьезность» и «Приоритет» с указанием цифрового или буквенного значения. Серьезность относится к технической стороне вопроса, а приоритет – к менеджерской. На сегодняшний день, приоритет принято разделять на три уровня, а серьезность – на пять:

Приоритет (Priority) – это атрибут, указывающий на очередность выполнения задачи или устранения дефекта. Проставляется руководителем или менеджером проекта.

* P1 – Высокий (High) – требуется исправить в первую очередь;
* P2 – Средний (Medium) – требуется исправить во вторую очередь, когда нет дефектов с высоким приоритетом;
* P3 – Низкий (Low) – исправляется в последнюю очередь, когда все дефекты с более высоким приоритетом уже исправлены.

**Серьезность (Severity)** – это атрибут, характеризующий влияние дефекта на работоспособность приложения. Проставляется тестировщиком или техническим специалистом, который может оценить степень влияния дефекта на работу системы.

* S1 – Блокирующий (Blocker) – дефект полностью блокирует выполнение функционала, нет никакого способа его обойти. Если провести аналогию с закрытым помещением и дверью – то дверь закрыта, у вас нет никакой возможности её открыть и покинуть помещение. Окон нет, ключ к двери не подходит.
* S2 – Критический (Critical) – дефект блокирует часть функциональности, но есть альтернативный путь для его обхода. По аналогии с помещением и дверью: вы можете покинуть помещение через окно, хотя дверь по-прежнему закрыта и ключ к ней не подходит.
* S3 – Значительный (Major) – дефект, указывающий на некорректную работу части функциональности. Зачастую связан не с тем, что функция не работает, а с тем, что она работает неправильно. В любом случае, существует более одной точки входа для инициации нужной функциональности. Так, вы можете покинуть помещение без использования ключа (дыра в безопасности), через вентиляцию (другая точка входа) или дверь открывается не в ту сторону (как следствие, упирается в угол и открывается только частично – некорректная реализация).
* S4 – Незначительный (Minor) – дефект, не относящийся к функциональности системы. Обычно серьезность Minor проставляется для тех дефектов, которые относятся к удобству использования или интерфейсу. По аналогии с помещением и дверью – на двери написано «От себя», хотя она открывается на себя, неудобное расположение замочной скважины и т.д.
* S5 – Тривиальный (Trivial) – дефект, не затрагивающий функциональность системы, а также оказывающий минимальное влияние на общее качество системы. Часто неотличим от уровня «minor». Обычно это грамматические дефекты в сопроводительной документации к системе. Иногда дефект относится к «невидимым» проблемам с точки зрения пользователя или пользовательского интерфейса и рассматривает сторонние библиотеки или сервисы, не относящиеся к самой разработанной системе. По аналогии с помещением и дверью – замок и ключ не одного производителя, в помещении слышится шум сверху (не относится к самому помещению) и т.д.

**Документировать дефекты** необходимо для:

* официального описания дефектов;
* назначения ответственных за их исправление;
* мониторинга дефектов;
* сбора отчетности по работе над проектом или отдельной его составляющей.

**Документирование ошибки**

● Цель составления отчета об ошибке — ее исправление

● Отчет об ошибке (участие нескольких заинтересованных лиц, одновременная работа с несколькими ошибками, исторический анализ)

● Качество исправления зависит от качества документирования ошибки

● Отчет об ошибке надо составлять сразу, иначе — проблема воспроизведения

**Структура отчета**

● программа/модуль

● версия

● тип отчета (ошибки)

● важность ошибки

● описание

● повторяемость

● алгоритм действий (воспроизведение)

● предлагаемое исправление

● автор отчета и ответственный за исправление

● Комментарии

● Состояние

● Приоритет (отмечает руководитель, может сильно расходиться с важностью)

● Резолюция (текущее состояние)

**Критерии отчета об ошибке**

● реальный документ

– проблемы с проверкой, воспроизведением

● нумерация/индексация

● простота (1 ошибка — 1 отчет)

● понятность

● воспроизводимость

● беспристрастность

Анализ воспроизводимости:

● записать последовательность действий приводящих к ошибке

● не торопиться с завершением написания отчета

● выявить наиболее серьезные проблемы (повышение важности)

● найти кратчайший путь воспроизведения (облегчение отладки)

● найти альтернативные действия, приводящие к этому результату

● выявить связанные проблемы

● проверка более ранних версий (поиск модификаций в коде)

● проверка на нескольких конфигурациях

● не воспроизводить на исходных данных,

использовать резервное копирование

● ошибка происходит до появления признаков

**Ошибки при описании ошибок**

* Сокращение инструкции по воспроизведению ошибки
* Отсутствие описания ошибочного и обоснования ожидаемого поведения
* Использование нечётких или неоднозначных формулировок

**Распространенные ошибки при описании дефектов**: 1. Сокращение инструкции по воспроизведению ошибки: − Использование сокращений. − Частое применение аббревиатур. − Опускание «маловажных» подробностей. 2. Отсутствие описания ошибочного поведения. 3. Отсутствие обоснования ожидаемого поведения. 4. Использование нечётких или неоднозначных формулировок

**Рекомендации по хорошему описанию дефектов:**

1. Подробное описание результатов

2. Понятное описание дефекта (точные названия программного средства)

3. Предоставление ссылки на соответствующее требование,

к нарушению, которого приводит фактический результат работы программного средства

4. Четкое указание окружения (ОС, браузер, настройки), под которым возникла ошибка

5. Своевременное создание дефекта

6. Поиск наиболее серьезных последствий ошибок

7. Проверка на заполненность полей и правильное написание

**Критерии определения критичности**

В процессе работы разработчики исправляют баги и реализуют новый функционал. Разные баги и функции имеют разную важность для проекта, разный приоритет (то есть порядок выполнения).

Но у разных людей – разные взгляды на то, что более важно, а что – менее. То, что тестировщик или разработчик сочтёт неважным, может оказаться существенным для заказчика. И наоборот.

Ну а ситуации, когда тестировщик утверждает, что ошибку обязательно нужно исправить, а разработчик с этим никак не соглашается, знакомы каждому из нас.

Кроме того. В отсутствие какого-либо регулирующего документа, при определении важности и приоритета нам остаётся полагаться только на собственное представление о продуктах. Учитывая масштаб и долгую историю продуктов, нет никакой гарантии, что оно а) будет правильным б) совпадёт у разных людей.

По этим причинам создается регламент определения критичности и приоритетности дефектов.

«Регламент определения критичности и приоритетности дефектов» вводится для упорядочивания работы над задачами и ошибками, унификации и упрощения таких задач, как определение очерёдности, в которой должны выполняться задания, простановка критичности тестировщиками, предотвращения споров о важности и приоритетности дефектов и задач.

|  |  |
| --- | --- |
| **Дополнительный функционал** | |
| **Функция** | **Пояснение** |
| Добавление контента в частном случае | Backend сайта. Ошибкой затрагиваются отдельные типы контента |
| Чтение контента в частном случае | Frontend сайта. Для конкретных браузеров и настроек. Ошибкой затрагивается не менее 25% страниц или любая из 10 самых посещаемых |
| Прочие функции backend сайта | Любые другие функции |
| Прочие функции frontend сайта | Любые другие функции |

**Для определения критичности используется следующий перечень функционала системы:**

|  |  |
| --- | --- |
| **Базовый функционал** |  |
| **Функция** | **Пояснение** |
| Реклама | Показ внешних баннеров и тизеров |
| Добавление контента в общем случае | Backend сайта. Ошибкой затрагивается большинство типов контента. |
| Чтение контента в общем случае | Frontend сайта. Все браузеры и настройки. Ошибкой затрагивается не менее 25% страниц или любая из 10 самых посещаемых |
| Продажи | Совершение покупки в интернет-магазине |