**БАГ**

Несоответствие ожидаемого и фактического результата. Несоответствие того как работает программа и как она должна работать по требованиям. Когда баг найден важно правильно описать его чтобы разработчик понял что не так и как должно быть. Для этого есть шаблон того что необходимо указать.

1) Название, оно должно отображать суть ошибки.

2) Шаги для воспроизведения. Что нужно сделать чтобы этот баг увидеть.

3)Ожидаемый результат и фактический.

4) Окружение: в каком браузере, в какой ОС

5) Скриншот.

6) Приоритет и критичность (для понимания как быстро его необходимо исправить)

7) В соответствии с багтрэкинговой системой могут присутствовать другие поля для заполнения. (ответственный за исправления, дата испраления, …).

В ISTQB написано, что человек совершает ошибки, которые приводят к возникновению дефектов в коде, которые, в свою очередь, приводят к сбоям и отказам приложения (однако сбои и отказы могут возникать и из-за внешних условий, таких как электромагнитное воздействие на оборудование и т.д.). Таким образом, упрощённо можно изобразить следующую схему:



**Ошибка** (error, mistake) — действие человека, приводящее к некорректным результатам. Этот термин очень часто используют как наиболее универсальный, описывающий любые проблемы («ошибка человека», «ошибка в коде», «ошибка в документации», «ошибка выполнения операции», «ошибка передачи данных», «ошибочный результат» и т.п.) Более того, куда чаще вы сможете услышать «отчёт об ошибке», чем «отчёт о дефекте». И это нормально, так сложилось исторически, к тому же термин «ошибка» на самом деле очень широкий.

**Дефект** (defect, bug, problem, fault) — недостаток в компоненте или системе, способный привести к ситуации сбоя или отказа. Этот термин также понимают достаточно широко, говоря о дефектах в документации, настройках, входных данных и т.д. Почему глава называется именно «отчёты о дефектах»? Потому что этот термин как раз стоит посередине — бессмысленно писать отчёты о человеческих ошибках, равно как и почти бесполезно просто описывать проявления сбоев и отказов — нужно докопаться до их причины, и первым шагом в этом направлении является именно описание дефекта.

**Сбой** (interruption) или отказ (failure) — отклонение поведения системы от ожидаемого. В ГОСТ 27.002-89 даны хорошие и краткие определения сбоя и отказа:

Сбой — самоустраняющийся отказ или однократный отказ, устраняемый незначительным вмешательством оператора.

Отказ — событие, заключающееся в нарушении работоспособного состояния объекта.

Эти термины скорее относятся к теории надёжности и нечасто встречаются в повседневной работе тестировщика, но именно сбои и отказы являются тем, что тестировщик замечает в процессе тестирования (и отталкиваясь от чего, проводит исследование с целью выявить дефект и его причины).

**Аномалия** (anomaly) или инцидент (incident , deviation) — любое отклонение наблюдаемого (фактического) состояния, поведения, значения, результата, свойства от ожиданий наблюдателя, сформированных на основе требований, спецификаций, иной документации или опыта и здравого смысла.

Итак, мы вернулись к тому, с чего начинали в части этой главы, описывающей предельно упрощённый взгляд на дефекты. Ошибки, дефекты, сбои, отказы и т.д. являются проявлением аномалий — отклонений фактического результата от ожидаемого. Стоит отметить, что ожидаемый результат действительно может основываться на опыте и здравом смысле, т.к. поведение программного средства никогда не специфицируют до уровня базовых элементарных приёмов работы с компьютером. Теперь, чтобы окончательно избавиться от путаницы и двусмысленности, договоримся, что мы будем считать дефектом в контексте данной книги:

**Дефект (**БАГ)— отклонение (deviation) фактического результата (actual result) от ожиданий наблюдателя, сформированных на основе требований, спецификаций, иной документации или опыта и здравого смысла. Отсюда логически вытекает, что дефекты могут встречаться не только в коде приложения, но и в любой документации, в архитектуре и дизайне, в настройках тестируемого приложения или тестового окружения — где угодно

**Отчёт о дефекте и его жизненный цикл**

при обнаружении дефекта тестировщик создаёт отчёт о дефекте.

**Отчёт о дефекте** (defect repor) — документ, описывающий и приоритизирующий обнаруженный дефект, а также содействующий его устранению. Как следует из самого определения, отчёт о дефекте пишется со следующими основными целями:

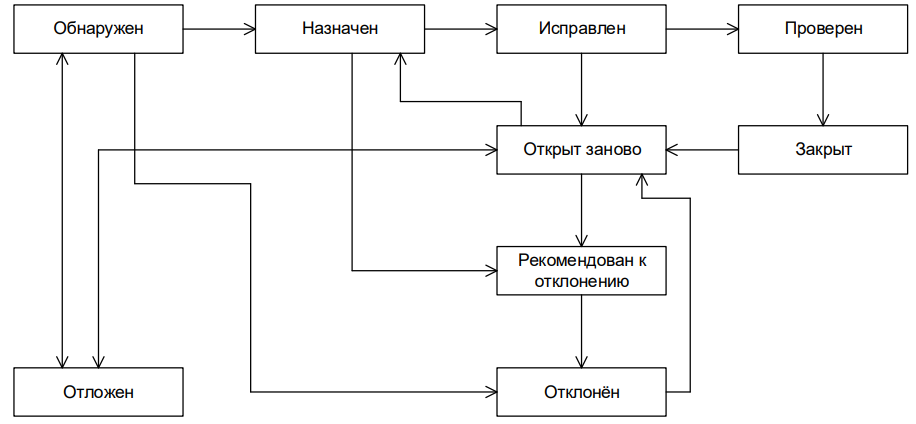
• предоставить информацию о проблеме — уведомить проектную команду и иных заинтересованных лиц о наличии проблемы, описать суть проблемы;

• приоритизировать проблему — определить степень опасности проблемы для проекта и желаемые сроки её устранения;

• содействовать устранению проблемы — качественный отчёт о дефекте не только предоставляет все необходимые подробности для понимания сути случившегося, но также может содержать анализ причин возникновения проблемы и рекомендации по исправлению ситуации.

На последней цели следует остановиться подробнее. Есть мнение, что «хорошо написанный отчёт о дефекте — половина решения проблемы для программиста». И действительно, как мы увидим далее (и особенно в главе «Типичные ошибки при написании отчётов о дефектах»{202} ), от полноты, корректности, аккуратности, подробности и логичности отчёта о дефекте зависит очень многое — одна и та же проблема может быть описана так, что программисту останется буквально исправить пару строк кода, а может быть описана и так, что сам автор отчёта на следующий день не сможет понять, что же он имел в виду.

ВАЖНО! «Сверхцель» написания отчёта о дефекте состоит в быстром исправлении ошибки (а в идеале — и недопущении её возникновения в будущем). Потому качеству отчётов о дефекте следует уделять особое, повышенное внимание. Отчёт о дефекте (и сам дефект вместе с ним) проходит определённые стадии жизненного цикла, которые схематично можно показать так:



**• Обнаружен** (submitted) — начальное состояние отчёта (иногда называется «Новый» (new)), в котором он находится сразу после создания. Некоторые средства также позволяют сначала создавать черновик (draft) и лишь потом публиковать отчёт.

**• Назначен** (assigned) — в это состояние отчёт переходит с момента, когда ктото из проектной команды назначается ответственным за исправление дефекта. Назначение ответственного производится или решением лидера команды разработки, или коллегиально, или по добровольному принципу, или иным принятым в команде способом или выполняется автоматически на основе определённых правил.

**• Исправлен** (fixed) — в это состояние отчёт переводит ответственный за исправление дефекта член команды после выполнения соответствующих действий по исправлению.

**• Проверен** (verified) — в это состояние отчёт переводит тестировщик, удостоверившийся, что дефект на самом деле был устранён. Как правило, такую проверку выполняет тестировщик, изначально написавший отчёт о дефекте.

**• Закрыт** (closed) — состояние отчёта, означающее, что по данному дефекту не планируется никаких дальнейших действий (хотя, конечно, ничто не мешает в будущем этому дефекту стать «открытым заново» (reopened)). Здесь есть некоторые расхождения в жизненном цикле, принятом в разных инструментальных средствах управления отчётами о дефектах:

o В некоторых средствах существуют оба состояния — «Проверен» и «Закрыт», чтобы подчеркнуть, что в состоянии «Проверен» ещё могут потребоваться какие-то дополнительные действия (обсуждения, дополнительные проверки в новых билдах и т.д.), в то время как состояние «Закрыт» означает «с дефектом покончено, больше к этому вопросу не возвращаемся».

o В некоторых средствах одного из состояний нет (оно поглощается другим).

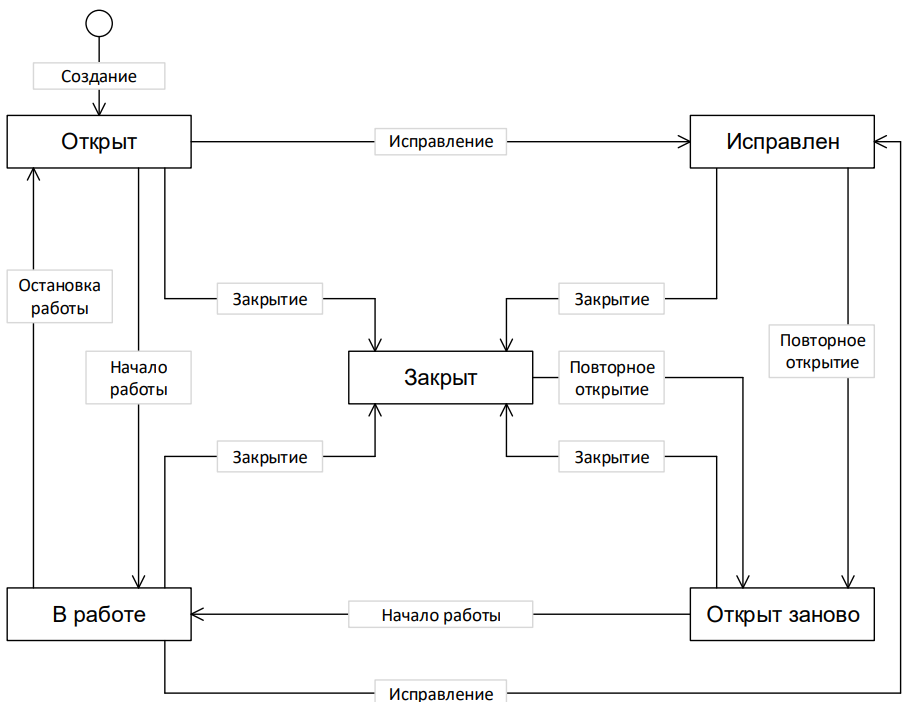
o В некоторых средствах в состояние «Закрыт» или «Отклонён» отчёт о дефекте может быть переведён из множества предшествующих состояний

• **Открыт заново** (reopened) — в это состояние (как правило, из состояния «Исправлен») отчёт переводит тестировщик, удостоверившийся, что дефект попрежнему воспроизводится на билде, в котором он уже должен быть исправлен.

**• Рекомендован к отклонению** (to be declined) — в это состояние отчёт о дефекте может быть переведён из множества других состояний с целью вынести на рассмотрение вопрос об отклонении отчёта по той или иной причине. Если рекомендация является обоснованной, отчёт переводится в состояние «Отклонён» (см. следующий пункт).

**• Отклонён** (declined) — в это состояние отчёт переводится в случаях, подробно описанных в пункте «Закрыт», если средство управления отчётами о дефектах предполагает использование этого состояния вместо состояния «Закрыт» для тех или иных резолюций по отчёту.

**• Отложен** (deferred) — в это состояние отчёт переводится в случае, если исправление дефекта в ближайшее время является нерациональным или не представляется возможным, однако есть основания полагать, что в обозримом будущем ситуация исправится (выйдет новая версия библиотеки, вернётся из отпуска специалист по некоей технологии, изменятся требования заказчика и т.д.). Для полноты рассмотрения данной подтемы приведём пример жизненного цикла, принятого по умолчанию в инструментальном средстве управления отчётами о дефектах JIRA



**Атрибуты (поля) отчёта о дефекте**

В зависимости от инструментального средства управления отчётами о дефектах внешний вид их записи может немного отличаться, могут быть добавлены или убраны отдельные поля, но концепция остаётся неизменной.

**Идентификатор** (identifier) представляет собой уникальное значение, позволяющее однозначно отличить один отчёт о дефекте от другого и используемое во всевозможных ссылках. В общем случае идентификатор отчёта о дефекте может представлять собой просто уникальный номер, но (если позволяет инструментальное средство управления отчётами) может быть и куда сложнее: включать префиксы, суффиксы и иные осмысленные компоненты, позволяющие быстро определить суть дефекта и часть приложения (или требований), к которой он относится.

**Краткое описание** (summary) должно в предельно лаконичной форме давать исчерпывающий ответ на вопросы «Что произошло?» «Где это произошло»? «При каких условиях это произошло?».

Например: «Отсутствует логотип на странице приветствия, если пользователь является администратором»:

• Что произошло? Отсутствует логотип.

• Где это произошло? На странице приветствия.

• При каких условиях это произошло? Если пользователь является администратором.

Одной из самых больших проблем для начинающих тестировщиков является именно заполнение поля «краткое описание», которое одновременно должно:

• содержать предельно краткую, но в то же время достаточную для понимания сути проблемы информацию о дефекте;

• отвечать на только что упомянутые вопросы («что, где и при каких условиях случилось») или как минимум на те 1–2 вопроса, которые применимы к конкретной ситуации;

• быть достаточно коротким, чтобы полностью помещаться на экране (в тех системах управления отчётами о дефектах, где конец этого поля обрезается или приводит к появлению скроллинга);

• при необходимости содержать информацию об окружении, под которым был обнаружен дефект;

• по возможности не дублировать краткие описания других дефектов (и даже не быть похожими на них), чтобы дефекты было сложно перепутать или посчитать дубликатами друг друга;

• быть законченным предложением русского или английского (или иного) языка, построенным по соответствующим правилам грамматики.

Для создания хороших кратких описаний дефектов рекомендуется пользоваться следующим алгоритмом:

1. Полноценно понять суть проблемы. До тех пор, пока у тестировщика нет чёткого, кристально чистого понимания того, «что сломалось», писать отчёт о дефекте вообще едва ли стоит.

2. Сформулировать подробное описание (description) дефекта — сначала без оглядки на длину получившегося текста.

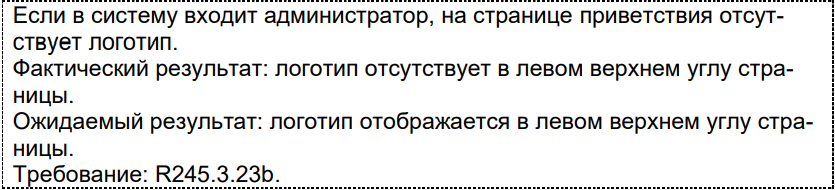
3. Убрать из получившегося подробного описания всё лишнее, уточнить важные детали.

4. Выделить в подробном описании слова (словосочетания, фрагменты фраз), отвечающие на вопросы, «что, где и при каких условиях случилось».

5. Оформить получившееся в пункте 4 в виде законченного грамматически правильного предложения.

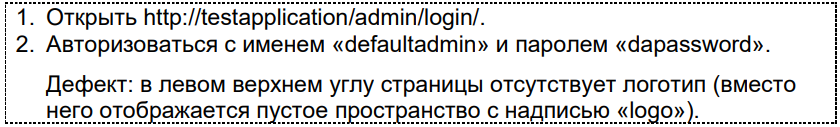
6. Если предложение получилось слишком длинным, переформулировать его, сократив длину (за счёт подбора синонимов, использования общепринятых аббревиатур и сокращений). К слову, в английском языке предложение почти всегда будет короче русского аналога.

**Подробное описание** (description) представляет в развёрнутом виде необходимую информацию о дефекте, а также (обязательно!) описание фактического результата, ожидаемого результата и ссылку на требование (если это возможно). Пример подробного описания:



В отличие от краткого описания, которое, как правило, является одним предложением, здесь можно и нужно давать подробную информацию. Если одна и та же проблема (вызванная одним источником) проявляется в нескольких местах приложения, можно в подробном описании перечислить эти места.

**Шаги по воспроизведению** (steps to reproduce, STR) описывают действия, которые необходимо выполнить для воспроизведения дефекта. Это поле похоже на шаги тест-кейса, за исключением одного важного отличия: здесь действия прописываются максимально подробно, с указанием конкретных вводимых значений и самых мелких деталей, т.к. отсутствие этой информации в сложных случаях может привести к невозможности воспроизведения дефекта. Пример шагов воспроизведения:



**Воспроизводимость** (reproducibility) показывает, при каждом ли прохождении по шагам воспроизведения дефекта удаётся вызвать его проявление. Это поле принимает всего два значения: всегда (always) или иногда (sometimes). Можно сказать, что воспроизводимость «иногда» означает, что тестировщик не нашёл настоящую причину возникновения дефекта. Это приводит к серьёзным дополнительным сложностям в работе с дефектом: • Тестировщику нужно потратить много времени на то, чтобы удостовериться в наличии дефекта (т.к. однократный сбой в работе приложения мог быть вызван огромным количеством посторонних причин).

• Разработчику тоже нужно потратить время, чтобы добиться проявления дефекта и убедиться в его наличии. После внесения исправлений в приложение разработчик фактически должен полагаться только на свой профессионализм, т.к. даже многократное прохождение по шагам воспроизведения в таком случае не гарантирует, что дефект был исправлен (возможно, через ещё 10–20 повторений он бы проявился).

• Тестировщику, верифицирующему исправление дефекта и вовсе остаётся верить разработчику на слово по той же самой причине: даже если он попытается воспроизвести дефект 100 раз и потом прекратит попытки, может так случиться, что на 101-й раз дефект всё же воспроизвёлся бы. Как легко догадаться, такая ситуация является крайне неприятной, а потому рекомендуется один раз потратить время на тщательную диагностику проблемы, найти её причину и перевести дефект в разряд воспроизводимых всегда

**Важность** (severity) показывает степень ущерба, который наносится проекту существованием дефекта. В общем случае выделяют следующие градации важности:

• Критическая (critical) — существование дефекта приводит к масштабным последствиям катастрофического характера, например: потеря данных, раскрытие конфиденциальной информации, нарушение ключевой функциональности приложения и т.д.

• Высокая (major) — существование дефекта приносит ощутимые неудобства многим пользователям в рамках их типичной деятельности, например: недоступность вставки из буфера обмена, неработоспособность общепринятых клавиатурных комбинаций, необходимость перезапуска приложения при выполнении типичных сценариев работы.

• Средняя (medium) — существование дефекта слабо влияет на типичные сценарии работы пользователей, и/или существует обходной путь достижения цели, например: диалоговое окно не закрывается автоматически после нажатия кнопок «OK»/«Cancel», при распечатке нескольких документов подряд не сохраняется значение поля «Двусторонняя печать», перепутаны направления сортировок по некоему полю таблицы.

• Низкая (minor) — существование дефекта редко обнаруживается незначительным процентом пользователей и (почти) не влияет на их работу, например: опечатка в глубоко вложенном пункте меню настроек, некое окно сразу при отображении расположено неудобно (нужно перетянуть его в удобное место), неточно отображается время до завершения операции копирования файлов.

**Срочность** (priority) показывает, как быстро дефект должен быть устранён. В общем случае выделяют следующие градации срочности:

• Наивысшая (ASAP, as soon as possible) срочность указывает на необходимость устранить дефект настолько быстро, насколько это возможно. В зависимости от контекста «настолько быстро, насколько возможно» может варьироваться от «в ближайшем билде» до единиц минут.

• 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.

**Симптом** (symptom) — позволяет классифицировать дефекты по их типичному проявлению. Не существует никакого общепринятого списка симптомов. Более того, далеко не в каждом инструментальном средстве управления отчётами о дефектах есть такое поле, а там, где оно есть, его можно настроить. В качестве примера рассмотрим следующие значения симптомов дефекта.

• Косметический дефект (cosmetic flaw) — визуально заметный недостаток интерфейса, не влияющий на функциональность приложения (например, надпись на кнопке выполнена шрифтом не той гарнитуры).

• Повреждение/потеря данных (data corruption/loss) — в результате возникновения дефекта искажаются, уничтожаются (или не сохраняются) некоторые данные (например, при копировании файлов копии оказываются повреждёнными).

• Проблема в документации (documentation issue) — дефект относится не к приложению, а к документации (например, отсутствует раздел руководства по эксплуатации).

• Некорректная операция (incorrect operation) — некоторая операция выполняется некорректно (например, калькулятор показывает ответ 17 при умножении 2 на 3).

• Проблема инсталляции (installation problem) — дефект проявляется на стадии установки и/или конфигурирования приложения (см. инсталляционное тестирование{86} ).

• Ошибка локализации (localization issue) — что-то в приложении не переведено или переведено неверно на выбранный язык интерфейса (см. локализационное тестирование{89} ).

• Нереализованная функциональность (missing feature) — некая функция приложения не выполняется или не может быть вызвана (например, в списке форматов для экспорта документа отсутствует несколько пунктов, которые там должны быть).

• Проблема масштабируемости (scalability) — при увеличении количества доступных приложению ресурсов не происходит ожидаемого прироста производительности приложения (см. тестирование производительности{91} и тестирование масштабируемости{92} ).

• Низкая производительность (low performance) — выполнение неких операций занимает недопустимо большое время (см. тестирование производительности{91} ).

• Крах системы (system crash) — приложение прекращает работу или теряет способность выполнять свои ключевые функции (также может сопровождаться крахом операционной системы, веб-сервера и т.д.).

• Неожиданное поведение (unexpected behavior) — в процессе выполнения некоторой типичной операции приложение ведёт себя необычным (отличным от общепринятого) образом (например, после добавления в список новой записи активной становится не новая запись, а первая в списке).

• Недружественное поведение (unfriendly behavior) — поведение приложения создаёт пользователю неудобства в работе (например, на разных диалоговых окнах в разном порядке расположены кнопки «OK» и «Cancel»).

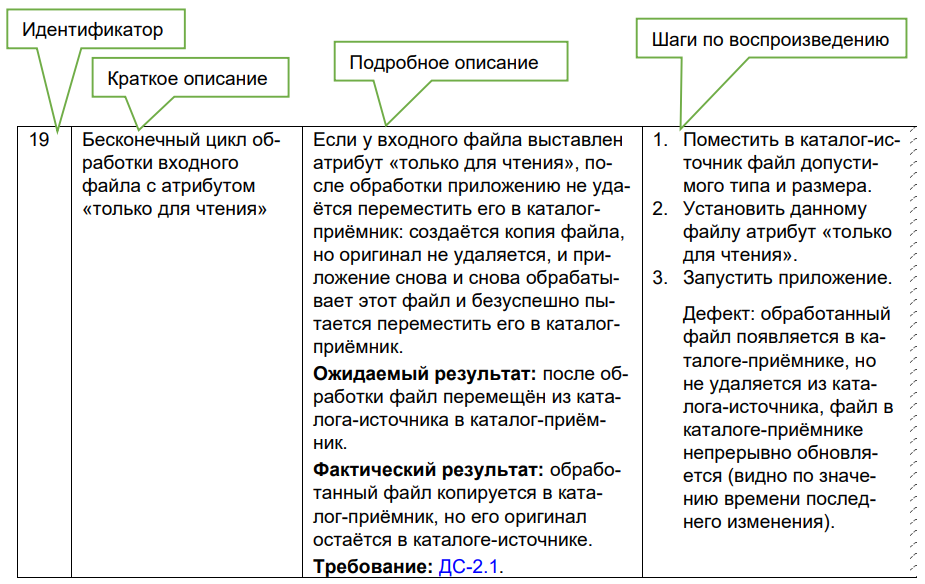
• Расхождение с требованиями (variance from specs) — этот симптом указывают, если дефект сложно соотнести с другими симптомами, но тем не менее приложение ведёт себя не так, как описано в требованиях.

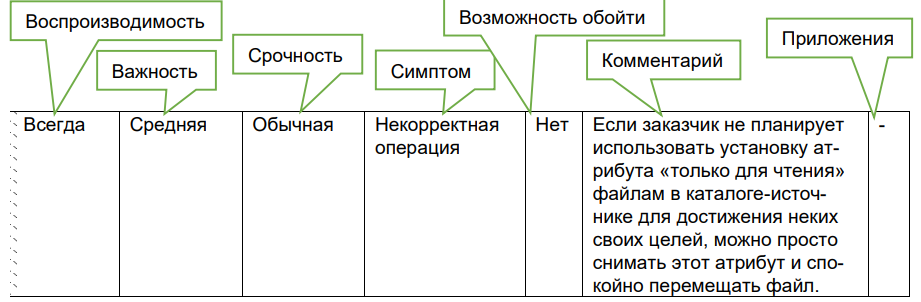
• Предложение по улучшению (enhancement) — во многих инструментальных средствах управления отчётами о дефектах для этого случая есть отдельный вид отчёта, т.к. предложение по улучшению формально нельзя считать дефектом: приложение ведёт себя согласно требованиям, но у тестировщика есть обоснованное мнение о том, как ту или иную функциональность можно улучшить.

Часто встречается вопрос о том, может ли у одного дефекта быть сразу несколько симптомов. Да, может

**Возможность обойти** (workaround) — показывает, существует ли альтернативная последовательность действий, выполнение которой позволило бы пользователю достичь поставленной цели (например, клавиатурная комбинация Ctrl+P не работает, но распечатать документ можно, выбрав соответствующие пункты в меню). В некоторых инструментальных средствах управления отчётами о дефектах это поле может просто принимать значения «Да» и «Нет», в некоторых при выборе «Да» появляется возможность описать обходной путь. Традиционно считается, что дефектам без возможности обхода стоит повысить срочность исправления. **Комментарий** (comments, additional info) — может содержать любые полезные для понимания и исправления дефекта данные. Иными словами, сюда можно писать всё то, что нельзя писать в остальные поля.

Таблица длинная в один скриншот не поместилась!





**Вложения** (attachments) — представляет собой не столько поле, сколько список прикреплённых к отчёту о дефекте приложений (копий экрана, вызывающих сбой файлов и т.д.). Общие рекомендации по формированию приложений таковы:

• Если вы сомневаетесь, делать или не делать приложение, лучше сделайте.

• Обязательно прилагайте т.н. «проблемные артефакты» (например, файлы, которые приложение обрабатывает некорректно).

• Если вы прилагаете копию экрана:

- Чаще всего вам будет нужна копия активного окна (Alt+PrintScreen), а не всего экрана (PrintScreen). o Обрежьте всё лишнее (используйте Snipping Tool или Paint в Windows, Pinta или XPaint в Linux).

- Отметьте на копии экрана проблемные места (обведите, нарисуйте стрелку, добавьте надпись — сделайте всё необходимое, чтобы с первого взгляда проблема была заметна и понятна).

- В некоторых случаях стоит сделать одно большое изображение из нескольких копий экрана (разместив их последовательно), чтобы показать процесс воспроизведения дефекта. Альтернативой этого решения является создание нескольких копий экрана, названных так, чтобы имена образовывали последовательность, например: br\_9\_sc\_01.png, br\_9\_sc\_02.png, br\_9\_sc\_03.png.

- Сохраните копию экрана в формате JPG (если важна экономия объёма данных) или PNG (если важна точная передача картинки без искажений)

• Если вы прилагаете видеоролик с записью происходящего на экране, обязательно оставляйте только тот фрагмент, который относится к описываемому дефекту (это будет буквально несколько секунд или минут из возможных многих часов записи). Старайтесь подобрать настройки кодеков так, чтобы получить минимальный размер ролика при сохранении достаточного качества изображения.

• Поэкспериментируйте с различными инструментами создания копий экрана и записи видеороликов с происходящим на экране. Выберите наиболее удобное для вас программное обеспечение и приучите себя постоянно его использовать.

**Инструментальные средства управления отчётами о дефектах**

Так называемые «инструментальные средства управления отчётами о дефектах» в обычной разговорной речи называют «баг-трекинговыми системами», «баг-трекерами» и т.д. Но мы здесь по традиции будем придерживаться более строгой терминологии.

Инструментальных средств управления отчётами о дефектах (bug tracking system, defect management tool) очень много к тому же многие компании разрабатывают свои внутренние средства решения этой задачи. Зачастую такие инструментальные средства являются частями инструментальных средств управления тестированием . Как и в случае с инструментальными средствами управления тестированием, здесь не имеет смысла заучивать, как работать с отчётами о дефектах в том или ином средстве.

Мы лишь рассмотрим общий набор функций, как правило, реализуемых такими средствами:

• Создание отчётов о дефектах, управление их жизненным циклом с учётом контроля версий, прав доступа и разрешённых переходов из состояния в состояние.

• Сбор, анализ и предоставление статистики в удобной для восприятия человеком форме.

• Рассылка уведомлений, напоминаний и иных артефактов соответствующим сотрудникам.

• Организация взаимосвязей между отчётами о дефектах, тест-кейсами, требованиями и анализ таких связей с возможностью формирования рекомендаций.

• Подготовка информации для включения в отчёт о результатах тестирования.

• Интеграция с системами управления проектами. Иными словами, хорошее инструментальное средство управления жизненным циклом отчётов о дефектах не только избавляет человека от необходимости внимательно выполнять большое количество рутинных операций, но и предоставляет дополнительные возможности, облегчающие работу тестировщика и делающие её более эффективной.

Для общего развития и лучшего закрепления темы об оформлении отчётов о дефектах мы сейчас рассмотрим несколько картинок с формами из разных инструментальных средств.

**Jira**

**1**. Project (проект) позволяет указать, к какому проекту относится дефект.

**2**. Issue type (тип записи/артефакта) позволяет указать, что именно представляет собой создаваемый артефакт. JIRA позволяет создавать не только отчёты о дефектах, но и множество других артефактов, типы которых можно настраивать. По умолчанию представлены:

• Improvement (предложение по улучшению) — было описано подробно в разделе, посвящённом полям отчёта о дефекте (см. описание поля «симптом», значение «предложение по улучшению»).

• New feature (новая особенность) — описание новой функциональности, нового свойства, новой особенности продукта.

• Task (задание) — некое задание для выполнения тем или иным участником проектной команды.

• Custom issue (произвольный артефакт) — как правило, это значение при настройке JIRA удаляют, заменяя своими вариантами, или переименовывают в Issue.

**3**. Summary (краткое описание) позволяет указать краткое описание дефекта.

**4**. Priority (срочность) позволяет указать срочность исправления дефекта. По умолчанию JIRA предлагает следующие варианты:

• Highest (самая высокая срочность).

• High (высокая срочность).

• Medium (обычная срочность).

• Low (низкая срочность).

• Lowest (самая низкая срочность).

Обратите внимание: по умолчанию поля severity (важность) нет. Но его можно добавить.

**5**. Components (компоненты) содержит перечень компонентов приложения, затронутых дефектом (хотя иногда здесь перечисляют симптомы дефектов).

**6**. Affected versions (затронутые версии) содержит перечень версий продукта, в которых проявляется дефект.

**7**. Environment (окружение) содержит описание аппаратной и программной конфигурации, в которой проявляется дефект.

**8**. Description (подробное описание) позволяет указать подробное описание дефекта.

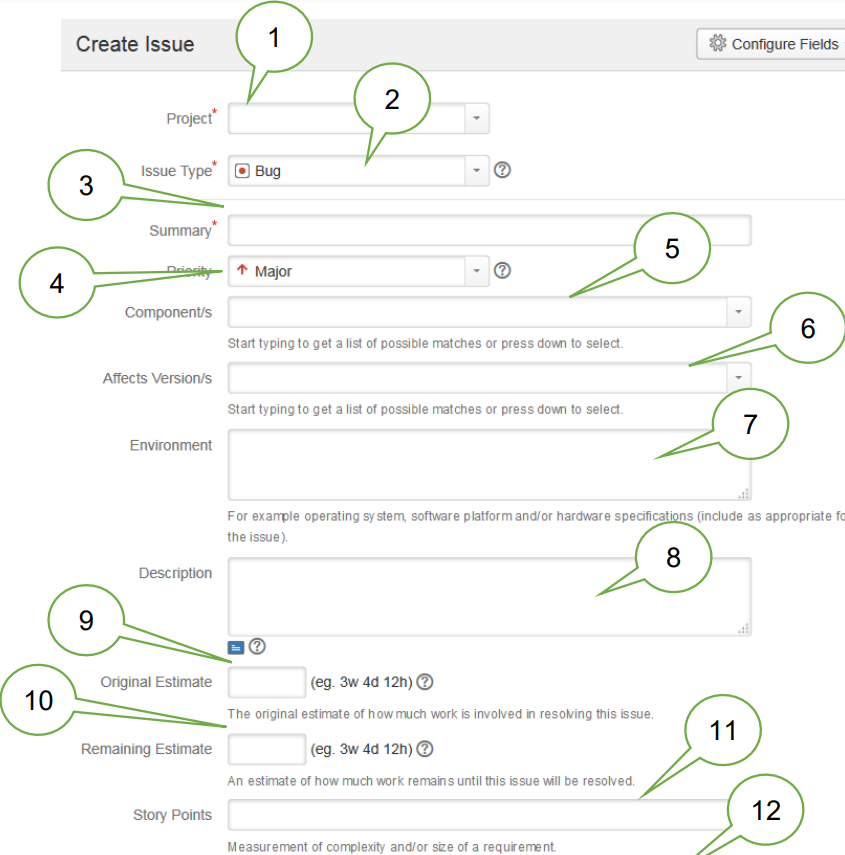
**9**. Original estimate (начальная оценка времени исправления) позволяет указать начальную оценку того, сколько времени займёт устранение дефекта.

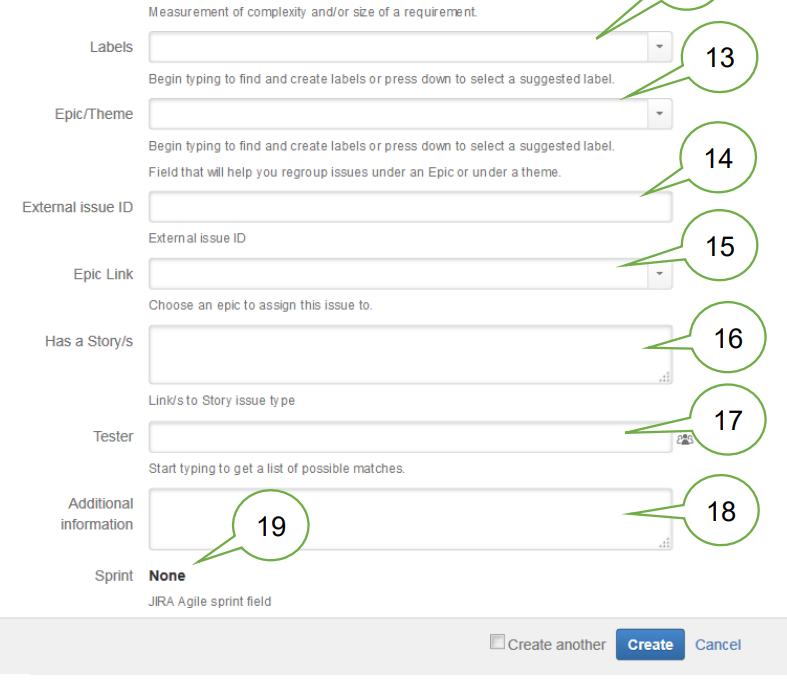
**10**.Remaining estimate (расчётное остаточное время исправления) показывает, сколько времени осталось от начальной оценки.

**11**.Story points (оценочные единицы) позволяет указать сложность дефекта (или иного артефакта) в специальных оценочных единицах, принятых в гибких методологиях управления проектами.

**12**.Labels (метки) содержит метки (теги, ключевые слова), по которым можно группировать и классифицировать дефекты и иные артефакты.

**13**.Epic/Theme (история/область) содержит перечень высокоуровневых меток, описывающих относящиеся к дефекту крупные области требований, крупные модули приложения, крупные части предметной области, объёмные пользовательские истории и т.д.





**14**.External issue id (идентификатор внешнего артефакта) позволяет связать отчёт о дефекте или иной артефакт с внешним документом.

**15**.Epic link (ссылка на историю/область) содержит ссылку на историю/область (см. пункт 13), наиболее близко относящуюся к дефекту.

**16**.Has a story/s (истории) содержит ссылки и/или описание пользовательских историй, связанных с дефектом (как правило, здесь приводятся ссылки на внешние документы).

**17**.Tester (тестировщик) содержит имя автора описания дефекта.

**18**.Additional information (дополнительная информация) содержит полезную дополнительную информацию о дефекте.

**19**.Sprint (спринт) содержит номер спринта (2–4-недельной итерации разработки проекта в терминологии гибких методологий управления проектами), во время которого был обнаружен дефект.

**Свойства качественных отчётов о дефектах**

Отчёт о дефекте может оказаться некачественным (а следовательно, вероятность исправления дефекта понизится), если в нём нарушено одно из следующих свойств.

**Тщательное заполнение всех полей точной и корректной информацией**. Нарушение этого свойства происходит по множеству причин: недостаточный опыт тестировщика, невнимательность, лень и т.д. Самыми яркими проявлениями такой проблемы можно считать следующие:

• Часть важных для понимания проблемы полей не заполнена. В результате отчёт превращается в набор обрывочных сведений, использовать которые для исправления дефекта невозможно.

• Предоставленной информации недостаточно для понимания сути проблемы. Например, из такого плохого подробного описания вообще не ясно, о чём идёт речь: «Приложение иногда неверно конвертирует некоторые файлы».

• Предоставленная информация является некорректной (например, указаны неверные сообщения приложения, неверные технические термины и т.д.). Чаще всего такое происходит по невнимательности (последствия ошибочного copy-paste и отсутствия финальной вычитки отчёта перед публикацией).

• «Дефект» (именно так, в кавычках) найден в функциональности, которая ещё не была объявлена как готовая к тестированию. То есть тестировщик констатирует, что неверно работает то, что и не должно было (пока!) верно работать.

• В отчёте присутствует жаргонная лексика: как в прямом смысле — нелитературные высказывания, так и некие технические жаргонизмы, понятные крайне ограниченному кругу людей. Например: «Фигово подцепились чартники». (Имелось в виду: «Не все таблицы кодировок загружены успешно».)

• Отчёт вместо описания проблемы с приложением критикует работу кого-то из участников проектной команды. Например: «Ну каким дураком надо быть, чтобы такое сделать?!»

• В отчёте упущена некая незначительная на первый взгляд, но по факту критичная для воспроизведения дефекта проблема. Чаще всего это проявляется в виде пропуска какого-то шага по воспроизведению, отсутствию или недостаточной подробности описания окружения, чрезмерно обобщённом указании вводимых значений и т.п.

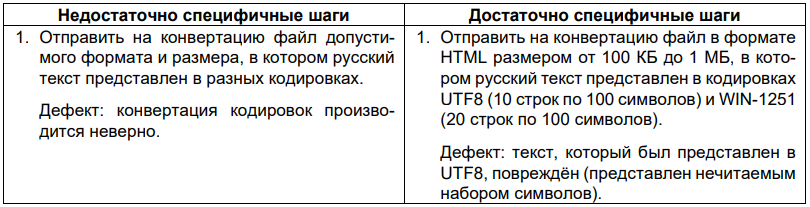
• Отчёту выставлены неверные (как правило, заниженные) важность или срочность. Чтобы избежать этой проблемы, стоит тщательно исследовать дефект, определять его наиболее опасные последствия и аргументированно отстаивать свою точку зрения, если коллеги считают иначе.

• К отчёту не приложены необходимые копии экрана (особенно важные для косметических дефектов) или иные файлы. Классика такой ошибки: отчёт описывает неверную работу приложения с некоторым файлом, но сам файл не приложен.

• Отчёт написан безграмотно с точки зрения человеческого языка. Иногда на это можно закрыть глаза, но иногда это становится реальной проблемой, например: «Not keyboard in parameters accepting values» (это реальная цитата; и сам автор так и не смог пояснить, что же имелось в виду).

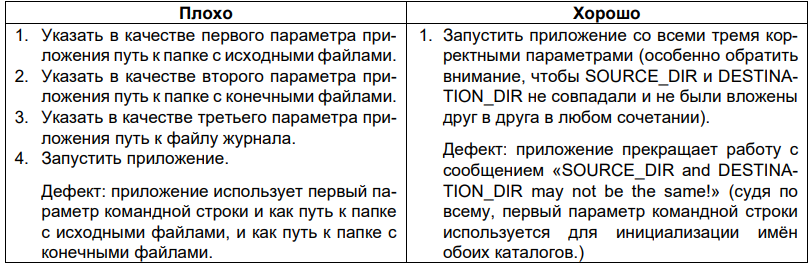
**Правильный технический язык.** Это свойство в равной мере относится и к требованиям, и к тест-кейсам, и к отчётам о дефектах — к любой документации, а потому не будем повторяться — см. описанное ранее

**Специфичность описания шагов**. Говоря о тест-кейсах, мы подчёркивали, что в их шагах стоит придерживаться золотой середины между специфичностью и общностью. В отчётах о дефектах предпочтение, как правило, отдаётся специфичности по очень простой причине: нехватка какой-то мелкой детали может привести к невозможности воспроизведения дефекта. Потому, если у вас есть хоть малейшее сомнение в том, важна ли какая-то деталь, считайте, что она важна.

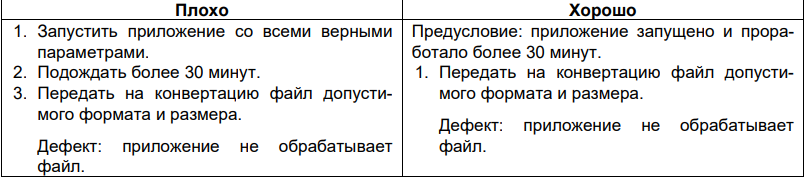


В первом случае воспроизвести дефект практически нереально, т.к. он заключается в особенностях работы внешних библиотек по определению кодировок текста в документе, в то время как во втором случае данных достаточно если и не для понимания сути происходящего (дефект на самом деле очень «хитрый»), то хотя бы для гарантированного воспроизведения и признания факта его наличия. Ещё раз главная мысль: в отличие от тест-кейса отчёт о дефекте может обладать повышенной специфичностью, и это будет меньшей проблемой, чем невозможность воспроизведения дефекта из-за излишне обобщённого описания проблемы.

**Отсутствие лишних действий и/или их длинных описаний**. Чаще всего это свойство подразумевает, что не нужно в шагах по воспроизведению дефекта долго и по пунктам расписывать то, что можно заменить одной фразой:



Вторая по частоте ошибка — начало каждого отчёта о дефекте с запуска приложения и подробного описания по приведению его в то или иное состояние. Вполне допустимой практикой является написание в отчёте о дефекте приготовлений (по аналогии с тест-кейсами) или описание нужного состояния приложения в одном (первом) шаге



**Отсутствие дубликатов.** Когда в проектной команде работает большое количество тестировщиков, может возникнуть ситуация, при которой один и тот же дефект будет описан несколько раз разными людьми. А иногда бывает так, что даже один и тот же тестировщик уже забыл, что когда-то давно уже обнаруживал некую проблему, и теперь описывает её заново. Избежать подобной ситуации позволяет следующий набор рекомендаций:

• Если вы не уверены, что дефект не был описан ранее, произведите поиск с помощью вашего инструментального средства управления жизненным циклом отчётов о дефектах.

• Пишите максимально информативные краткие описания (т.к. поиск в первую очередь проводят по ним). Если на вашем проекте накопится множество дефектов с краткими описаниями в стиле «Кнопка не работает», вы потратите очень много времени, раз за разом перебирая десятки таких отчётов в поисках нужной информации.

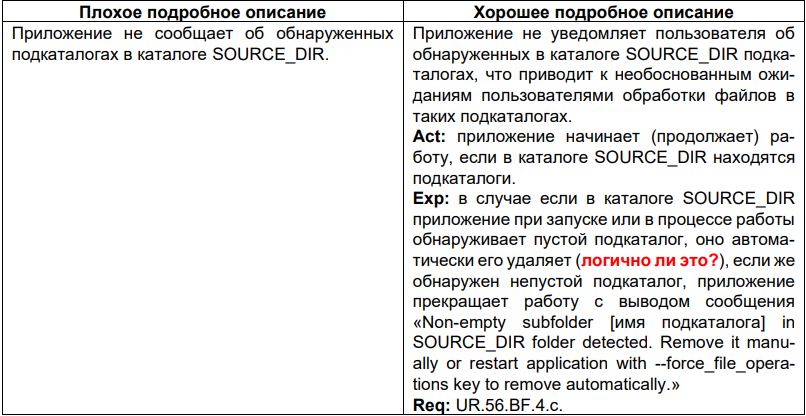
• Используйте по максимуму возможности вашего инструментального средства: указывайте в отчёте о дефекте компоненты приложения, ссылки на требования, расставляйте теги и т.д. — всё это позволит легко и быстро сузить в будущем круг поиска.

• Указывайте в подробном описании дефекта текст сообщений от приложения, если это возможно. По такому тексту можно найти даже тот отчёт, в котором остальная информация приведена в слишком общем виде.

• Старайтесь по возможности участвовать в т.н. «собраниях по прояснению» (clarification meetings326), т.к. пусть вы и не запомните каждый дефект или каждую пользовательскую историю дословно, но в нужный момент может возникнуть ощущение «что-то такое я уже слышал», которое заставит вас произвести поиск и подскажет, что именно искать.

• Если вы обнаружили какую-то дополнительную информацию, внесите её в уже существующий отчёт о дефекте (или попросите сделать это его автора), но не создавайте отдельный отчёт.

**Очевидность и понятность.** Описывайте дефект так, чтобы у читающего ваш отчёт не возникло ни малейшего сомнения в том, что это действительно дефект. Лучше всего это свойство достигается за счёт тщательного объяснения фактического и ожидаемого результата, а также указания ссылки на требование в поле «Подробное описание».



В первом случае после прочтения подробного описания очень хочется спросить: «И что? А оно разве должно уведомлять?» Второй же вариант подробного описания даёт чёткое пояснение, что такое поведение является ошибочным согласно текущему варианту требований. Более того, во втором варианте отмечен вопрос (а в идеале нужно сделать соответствующую отметку и в самом требовании), призывающий пересмотреть алгоритм корректного поведения приложения в подобной ситуации: т.е. мы не только качественно описываем текущую проблему, но и поднимаем вопрос о дальнейшем улучшении приложения.

**Прослеживаемость.** Из содержащейся в качественном отчёте о дефекте информации должно быть понятно, какую часть приложения, какие функции и какие требования затрагивает дефект. Лучше всего это свойство достигается правильным использованием возможностей инструментального средства управления отчётами о дефектах: указывайте в отчёте о дефекте компоненты приложения, ссылки на требования, тест-кейсы, смежные отчёты о дефектах (похожих дефектах; зависимых и зависящих от данного дефектах), расставляйте теги и т.д

Некоторые инструментальные средства даже позволяют строить визуальные схемы на основе таких данных, что позволяет управление прослеживаемостью даже на очень больших проектах превратить из непосильной для человека задачи в тривиальную работу.

**Отдельные отчёты для каждого нового дефекта.** Существует два незыблемых правила:

• В каждом отчёте описывается ровно один дефект (если один и тот же дефект проявляется в нескольких местах, эти проявления перечисляются в подробном описании).

• При обнаружении нового дефекта создаётся новый отчёт. Нельзя для описания нового дефекта править старые отчёты, переводя их в состояние «открыт заново». Нарушение первого правила приводит к объективной путанице, которую проще всего проиллюстрировать так: представьте, что в одном отчёте описано два дефекта, один из которых был исправлен, а второй — нет. В какое состояние переводить отчёт? Неизвестно. Нарушение второго правила вообще порождает хаос: мало того, что теряется информация о ранее возникавших дефектах, так к тому же возникают проблемы со всевозможными метриками и банальным здравым смыслом. Именно во избежание этой проблемы на многих проектах правом перевода отчёта из состояния «закрыт» в состояние «открыт заново» обладает ограниченный круг участников команды.

**Соответствие принятым шаблонам оформления и традициям**. Как и в случае с тест-кейсами, с шаблонами оформления отчётов о дефектах проблем не возникает: они определены имеющимся образцом или экранной формой инструментального средства управления жизненным циклом отчётов о дефектах. Но что касается традиций, которые могут различаться даже в разных командах в рамках одной компании, то единственный совет: почитайте уже готовые отчёты о дефектах, перед тем как писать свои. Это может сэкономить вам много времени и сил.

**Логика создания эффективных отчётов о дефектах**

При создании отчёта о дефекте рекомендуется следовать следующему алгоритму:

0. Обнаружить дефект ☺.

1. Понять суть проблемы.

2. Воспроизвести дефект.

3. Проверить наличие описания найденного вами дефекта в системе управления дефектами.

4. Сформулировать суть проблемы в виде «что сделали, что получили, что ожидали получить».

5. Заполнить поля отчёта, начиная с подробного описания.

6. После заполнения всех полей внимательно перечитать отчёт, исправив неточности и добавив подробности.

7. Ещё раз перечитать отчёт, т.к. в пункте 6 вы точно что-то упустили ☺.

**Понять суть проблемы** Всё начинается именно с понимания происходящего с приложением. Только при наличии такого понимания вы сможете написать по-настоящему качественный отчёт о дефекте, верно определить важность дефекта и дать полезные рекомендации по его устранению. **В идеале отчёт о дефекте описывает именно суть проблемы, а не её внешнее проявление.**

**Воспроизвести дефект** Это действие не только поможет в дальнейшем правильно заполнить поле «Воспроизводимость», но и позволит избежать неприятной ситуации, в которой за дефект приложения будет принят некий кратковременный сбой, который (скорее всего) произошёл где-то в вашем компьютере или в иной части ИТ-инфраструктуры, не имеющей отношения к тестируемому приложению.

**Проверить наличие описания найденного вами дефекта** Обязательно стоит проверить, нет ли в системе управления дефектами описания именно того дефекта, который вы только что обнаружили. Это простое действие, не относящееся непосредственно к написанию отчёта о дефекте, но значительно сокращающее количество отчётов, отклонённых с резолюцией «дубликат».

**Сформулировать суть проблемы** Формулировка проблемы в виде «что сделали (шаги по воспроизведению), что получили (фактический результат в подробном описании), что ожидали получить (ожидаемый результат в подробном описании)» позволяет не только подготовить данные для заполнения полей отчёта, но и ещё лучше понять суть проблемы. В общем же случае формула «что сделали, что получили, что ожидали получить» хороша по следующим причинам:

• Прозрачность и понятность: следуя этой формуле, вы готовите именно данные для отчёта о дефекте, не скатываясь в пространные отвлечённые рассуждения.

• Лёгкость верификации дефекта: разработчик, используя эти данные, может быстро воспроизвести дефект, а тестировщик после исправления дефекта удостовериться, что тот действительно исправлен.

• Очевидность для разработчиков: ещё до попытки воспроизведения дефекта видно, на самом ли деле описанное является дефектом, или тестировщик где-то ошибся, записав в дефекты корректное поведение приложения.

• Избавление от лишней бессмысленной коммуникации: подробные ответы на «что сделали, что получили, что ожидали получить» позволяют решать проблему и устранять дефект без необходимости запроса, поиска и обсуждения дополнительных сведений.

• Простота: на финальных стадиях тестирования с привлечением конечных пользователей можно ощутимо повысить эффективность поступающей обратной связи, если объяснить пользователям суть этой формулы и попросить их придерживаться её при написании сообщений об обнаруженных проблемах. Информация, полученная на данном этапе, становится фундаментом для всех дальнейших действий по написанию отчёта.

**Заполнить поля отчёта** Поля отчёта о дефекте мы уже рассмотрели ранее, сейчас лишь подчеркнём, что начинать лучше всего с подробного описания, т.к. в процессе заполнения этого поля может выявиться множество дополнительных деталей, а также появятся мысли по поводу формулирования сжатого и информативного краткого описания. Если вы понимаете, что для заполнения какого-то поля у вас не хватает данных, проведите дополнительное исследование. Если и оно не помогло, опишите в соответствующем поле (если оно текстовое), почему вы затрудняетесь с его заполнением, или (если поле представляет собой список) выберите значение, которое, на ваш взгляд, лучше всего характеризует проблему (в некоторых случаях инструментальное средство позволяет выбрать значение наподобие «неизвестно», тогда выберите его). Если у вас нет идей по поводу устранения дефекта, или он настолько тривиален, что не нуждается в подобных пояснениях, не пишите в комментариях «текст ради текста»: комментарии вида «рекомендую устранить этот дефект» не просто бессмысленны, но ещё и раздражают

**Перечитать отчёт** (и ещё раз перечитать отчёт) После того как всё написано, заполнено и подготовлено, ещё раз внимательно перечитайте написанное. Очень часто вы сможете обнаружить, что в процессе доработки текста где-то получились логические нестыковки или дублирование, где-то вам захочется улучшить формулировки, где-то что-то поменять.

После оформления отчёта о дефекте рекомендуется дополнительно исследовать ту область приложения, в которой вы только что обнаружили дефект. Практика показывает, что дефекты часто проявляются группами.

**Типичные ошибки при написании отчётов о дефектах**

**Плохие краткие описания (summary).** Формально эта проблема относится к оформлению, но фактически она куда опаснее: ведь чтение отчёта о дефекте и осознание сути проблемы начинается именно с краткого описания. Ещё раз напомним его суть:

• Отвечает на вопросы «что?», «где?», «при каких условиях?».

• Должно быть предельно кратким.

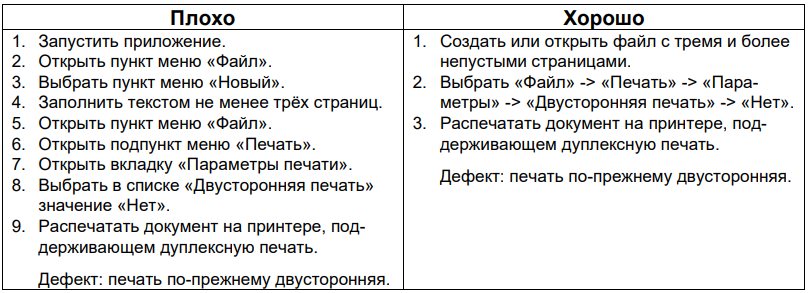
• Должно быть достаточно информативным для понимания сути проблемы.

**Идентичные краткое и подробное описания** (summary и description). Да, изредка бывают настолько простые дефекты, что для них достаточно одного краткого описания (например, «Опечатка в имени пункта главного меню “File” (сейчас “Fille”)»), но если дефект связан с неким более-менее сложным поведением приложения, стоит продумать как минимум три способа описания проблемы: • краткий для поля «краткое описание» (его лучше формулировать в самом конце размышлений); • подробный для поля «подробное описание» (поясняющий и расширяющий информацию из «краткого описания»); • ещё один краткий для последнего шага в шагах по воспроизведению дефекта.

**Отсутствие в подробном описании явного указания фактического результата, ожидаемого результата и ссылки на требование** - …

**Игнорирование кавычек, приводящее к искажению смысла**. Как вы поймёте такое краткое описание, как «запись исчезает при наведении мыши»? Какая то запись исчезает при наведении мыши? А вот и нет, оказывается, «поле “Запись” исчезает при наведении мыши». Даже если не дописать слово «поле», кавычки подскажут, что имеется в виду имя собственное, т.е. название некоего элемента.

**Лишние пункты в шагах воспроизведения**. Не стоит начинать «с сотворения мира», большинство проектной команды достаточно хорошо знает приложение, чтобы «опознать» его ключевые части, потому — сравните:



**Копии экрана в виде «копий всего экрана целиком»**. Чаще всего нужно сделать копию какого-то конкретного окна приложения, а не всего экрана — тогда поможет Alt+PrintScreen. Даже если важно захватить больше, чем одно окно, практически любой графический редактор позволяет отрезать ненужную часть картинки.

**Копии экрана, на которых не отмечена проблема.** Если обвести проблемную область красной линией, это в разы повысит скорость и простоту понимания сути проблемы в большинстве случаев

**Откладывание написания отчёта «на потом».** Стремление сначала найти побольше дефектов, а уже потом их описывать приводит к тому, что какие-то важные детали (а иногда и сами дефекты!) забываются. Если «на потом» измеряется не минутами, а часами или даже днями, проектная команда не получает важную информацию вовремя. Вывод простой: описывайте дефект сразу же, как только обнаружили его. **Пунктуационные, орфографические, синтаксические и им подобные ошибки**. Без комментариев.

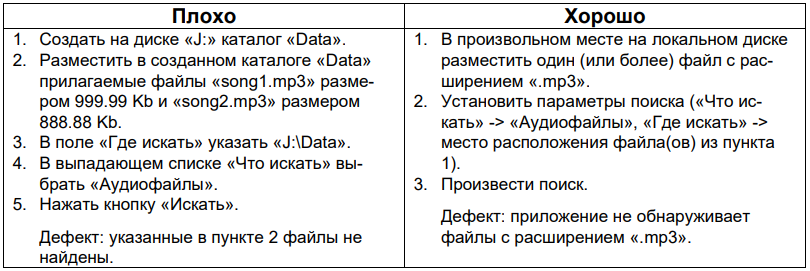
**Выдуманные дефекты**. Одной из наиболее обидных для тестировщика причин отклонения отчёта о дефекте является так называемое «описанное поведение не является дефектом» («not a bug»), когда по какой-то причине корректное поведение приложения описывается как ошибочное. Но ещё хуже, когда «якобы ожидаемое поведение» просто… придумывается из головы. То есть нигде в требованиях не сказано, что приложение должно что-то такое делать, но при этом создаётся отчёт о дефекте из-за того, что приложение этого не делает. И ладно бы это были некие спорные случаи или случайно попавшие в отчёты о дефектах предложения по улучшению, но нет — от приложения начинают требовать каких-то совершенно нелогичных, нерациональных, безумных вещей. Откуда это? Зачем? Просто не делайте так.

**Отнесение расширенных возможностей приложения к дефектам.** Самым ярким примером этого случая является описание как дефекта того факта, что приложение запускается под операционными системами, не указанными явно в списке поддерживаемых.

**Чрезмерно заниженные (или завышенные) важность и срочность**. С этой бедой достаточно эффективно борются проведением общих собраний и пересмотром отчётов о дефектах силами всей команды

**Концентрация на мелочах в ущерб главному**. Здесь стоит упомянуть хрестоматийный пример, когда тестировщик нашёл проблему, приводящую к краху приложения с потерей пользовательских данных, но записал её как косметический дефект (в сообщении об ошибке, которое «перед смертью» показывало приложение, была опечатка).

**Указание в шагах воспроизведения информации, неважной для воспроизведения ошибки**. Стремление прописать всё максимально подробно иногда принимает нездоровую форму, когда в отчёт о дефекте начинает попадать чуть ли не информация о погоде за окном и курс национальной валюты. Сравните:



Действительно ли для воспроизведения дефекта важно, чтобы поиск производился на диске «J:»? Действительно ли важно производить поиск именно файлов с такими именами и размерами в таком каталоге? Возможно, в некоем бесконечно маловероятном случае и да, тогда вопрос снимается. Но, скорее всего, это совершенно не важно, и нет никакой необходимости записывать эту информацию.

**Игнорирование т.н. «последовательных дефектов».** Иногда один дефект является следствием другого (допустим, файл повреждается при передаче на сервер, а затем приложение некорректно обрабатывает этот повреждённый файл). Да, если файл будет передан без повреждений, второй дефект может не проявиться.

Но может и проявиться в другой ситуации, т.к. проблема никуда не исчезла: приложение некорректно обрабатывает повреждённые файлы. Потому стоит описать оба дефекта.