



2.5. Отчёты о дефектах

2.5.1. Ошибки, дефекты, сбои, отказы и т.д.

Упрощённый взгляд на понятие дефекта

Далее в этой главе мы глубоко погрузимся в терминологию (она действительно важна!), а потому начнём с очень простого: дефектом упрощённо можно считать любое расхождение ожидаемого (свойства, результата, поведения и т.д., которое мы ожидали увидеть) и фактического (свойства, результата, поведения и т.д., которое мы на самом деле увидели). При обнаружении дефекта создаётся отчёт о дефекте.

	Дефект — расхождение ожидаемого и фактического результата. Ожидаемый результат — поведение системы, описанное в требованиях. Фактический результат — поведение системы, наблюдаемое в процессе тестирования.
	ВАЖНО! Эти три определения приведены в предельно упрощённой (и даже искажённой) форме с целью первичного ознакомления. Полноценные формулировки см. далее в этой же главе.

Поскольку столь простая трактовка не покрывает все возможные формы проявления проблем с программными продуктами, мы сразу же переходим к более подробному рассмотрению соответствующей терминологии.

Расширенный взгляд на терминологию, описывающую проблемы

Разберёмся с широким спектром синонимов, которыми обозначают проблемы с программными продуктами и иными артефактами и процессами, сопутствующими их разработке.

В силлабусе ISTQB написано³⁰⁸, 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:

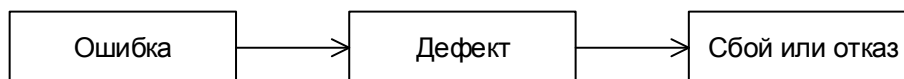


Рисунок 2.5.а — Ошибки, дефекты, сбои и отказы

³⁰⁸ A human being can make an error (mistake), which produces a defect (fault, bug) in the program code, or in a document. If a defect in code is executed, the system may fail to do what it should do (or do something it shouldn't), causing a failure. Defects in software, systems or documents may result in failures, but not all defects do so. Defects occur because human beings are fallible and because there is time pressure, complex code, complexity of infrastructure, changing technologies, and/or many system interactions. Failures can be caused by environmental conditions as well. For example, radiation, magnetism, electronic fields, and pollution can cause faults in firmware or influence the execution of software by changing the hardware conditions. [ISTQB Syllabus]

Если же посмотреть на англоязычную терминологию, представленную в глоссарии ISTQB и иных источниках, можно построить чуть более сложную схему:

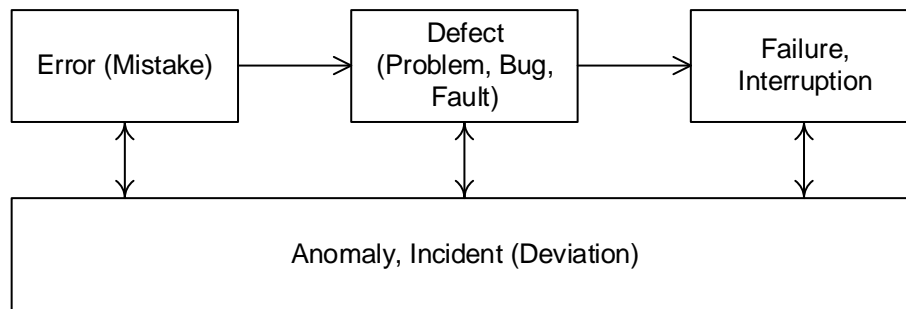


Рисунок 2.5.b — Взаимосвязь проблем в разработке программных продуктов

Рассмотрим все соответствующие термины.

!!! **Ошибка** (error³⁰⁹, 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** (defect³¹⁰, 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** (interruption³¹¹) или **отказ** (failure³¹²) — 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).

³⁰⁹ **Error, Mistake.** A human action that produces an incorrect result. [ISTQB Glossary]

³¹⁰ **Defect, Bug, Problem, Fault.** A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g. an incorrect statement or data definition. A defect, if encountered during execution, may cause a failure of the component or system. [ISTQB Glossary]

³¹¹ **Interruption.** A suspension of a process, such as the execution of a computer program, caused by an event external to that process and performed in such a way that the process can be resumed. [<http://www.electropedia.org/iev/iev.nsf/display?open-form&ievref=714-22-10>]

³¹² **Failure.** Deviation of the component or system from its expected delivery, service or result. [ISTQB Glossary]



Аномалия (anomaly³¹³) или **инцидент** (incident³¹⁴, 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 — отклонение (deviation³¹⁴) фактического результата (actual result³¹⁵) от ожиданий наблюдателя (expected result³¹⁶), 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.



Важно понимать, что приведённое определение дефекта позволяет лишь поднять вопрос о том, является ли некое поведение приложения дефектом. В случае, если из проектной документации не следует однозначного положительного ответа, обязательно стоит обсудить свои выводы с коллегами и добиться донесения поднятого вопроса до заказчика, если его мнение по обсуждаемому «кандидату в баги» неизвестно.



Хорошее представление о едва-едва затронутой нами теме теории надёжности можно получить, прочитав книгу Рудольфа Стапелберга «Руководство по надёжности, доступности, ремонтпригодности и безопасности в инженерном проектировании» (Rudolph Frederick Stapelberg, «Handbook of Reliability, Availability, Maintainability and Safety in Engineering Design»).

А краткую, но достаточно подробную классификацию аномалий в программных продуктах можно посмотреть в стандарте «IEEE 1044:2009 Standard Classification For Software Anomalies».

³¹³ **Anomaly.** Any condition that deviates from expectation based on requirements specifications, design documents, user documents, standards, etc. or from someone's perception or experience. Anomalies may be found during, but not limited to, reviewing, testing, analysis, compilation, or use of software products or applicable documentation. See also bug, defect, deviation, error, fault, failure, incident, problem. [ISTQB Glossary]

³¹⁴ **Incident, Deviation.** Any event occurring that requires investigation. [ISTQB Glossary]

³¹⁵ **Actual result.** The behavior produced/observed when a component or system is tested. [ISTQB Glossary]

³¹⁶ **Expected result, Expected outcome, Predicted outcome.** The behavior predicted by the specification, or another source, of the component or system under specified conditions. [ISTQB Glossary]

2.5.2. Отчёт о дефекте и его жизненный цикл

Как было сказано в предыдущей главе, при обнаружении дефекта тестировщик создаёт отчёт о дефекте.



Отчёт о дефекте (defect report³¹⁷) — 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»⁽¹⁹⁷⁾), 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 (рисунок 2.5.с):

- 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 кто-то 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.

³¹⁷ **Defect report, Bug report.** A document reporting on any flaw in a component or system that can cause the component or system to fail to perform its required function. [ISTQB Glossary]



По поводу того, должен ли проверять факт устранения дефекта именно тот тестировщик, который его обнаружил, или обязательно другой, есть много «священных войн». Сторонники второго варианта утверждают, что свежий взгляд человека, ранее не знакомого с данным дефектом, позволяет ему в процессе верификации с большой вероятностью обнаружить новые дефекты.

Несмотря на то, что такая точка зрения имеет право на существование, всё же отметим: при грамотной организации процесса тестирования поиск дефектов эффективно происходит на соответствующей стадии работы, а верификация силами тестировщика, обнаружившего данный дефект, всё же позволяет существенно сэкономить время.

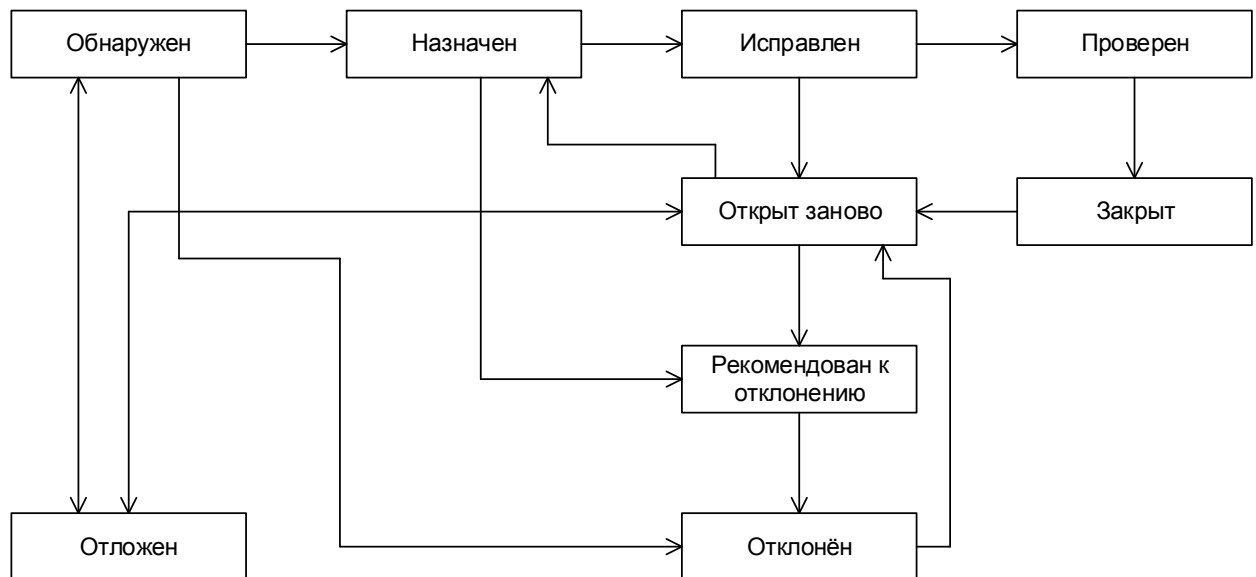


Рисунок 2.5.с — Жизненный цикл отчёта о дефекте с наиболее типичными переходами между состояниями



Набор стадий жизненного цикла, их наименование и принцип перехода от стадии к стадии может различаться в разных инструментальных средствах управления жизненным циклом отчётов о дефектах. Более того — многие такие средства позволяют гибко настраивать эти параметры. На рисунке 2.5.с показан лишь общий принцип.

- **Закрит (closed)** — состояние отчёта, означающее, что по данному дефекту не планируется никаких дальнейших действий. Здесь есть некоторые расхождения в жизненном цикле, принятом в разных инструментальных средствах управления отчётами о дефектах:
 - В некоторых средствах существуют оба состояния — «Проверен» и «Закрит», 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 «Закрит» означает «с дефектом покончено, больше к этому вопросу не возвращаемся».
 - В некоторых средствах одного из состояний нет (оно поглощается другим).

- В некоторых средствах в состояние «Закрыт» или «Отклонён» отчёт о дефекте может быть переведён из множества предшествующих состояний с резолюциями наподобие:
 - «Не является дефектом» — приложение так и должно работать, описанное поведение не является аномальным.
 - «Дубликат» — данный дефект уже описан в другом отчёте.
 - «Не удалось воспроизвести» — разработчикам не удалось воспроизвести проблему на своём оборудовании.
 - «Не будет исправлено» — дефект есть, но по каким-то серьёзным причинам его решено не исправлять.
 - «Невозможно исправить» — непреодолимая причина дефекта находится вне области полномочий команды разработчиков, например существует проблема в операционной системе или аппаратном обеспечении, влияние которой устранить разумными способами невозможно.

Как было только что подчёркнуто, в некоторых средствах отчёт о дефекте в подобных случаях будет переведён в состояние «Закрыт», в некоторых — в состояние «Отклонён», в некоторых — часть случаев закреплена за состоянием «Закрыт», часть — за «Отклонён».

- Открыт заново (reopened) — в это состояние (как правило, из состояния «Исправлен») отчёт переводит тестировщик, удостоверившийся, что дефект по-прежнему 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 JIRA³¹⁸ (рисунок 2.5.d).

³¹⁸ «What is Workflow». [<https://confluence.atlassian.com/display/JIRA/What+is+Workflow>]

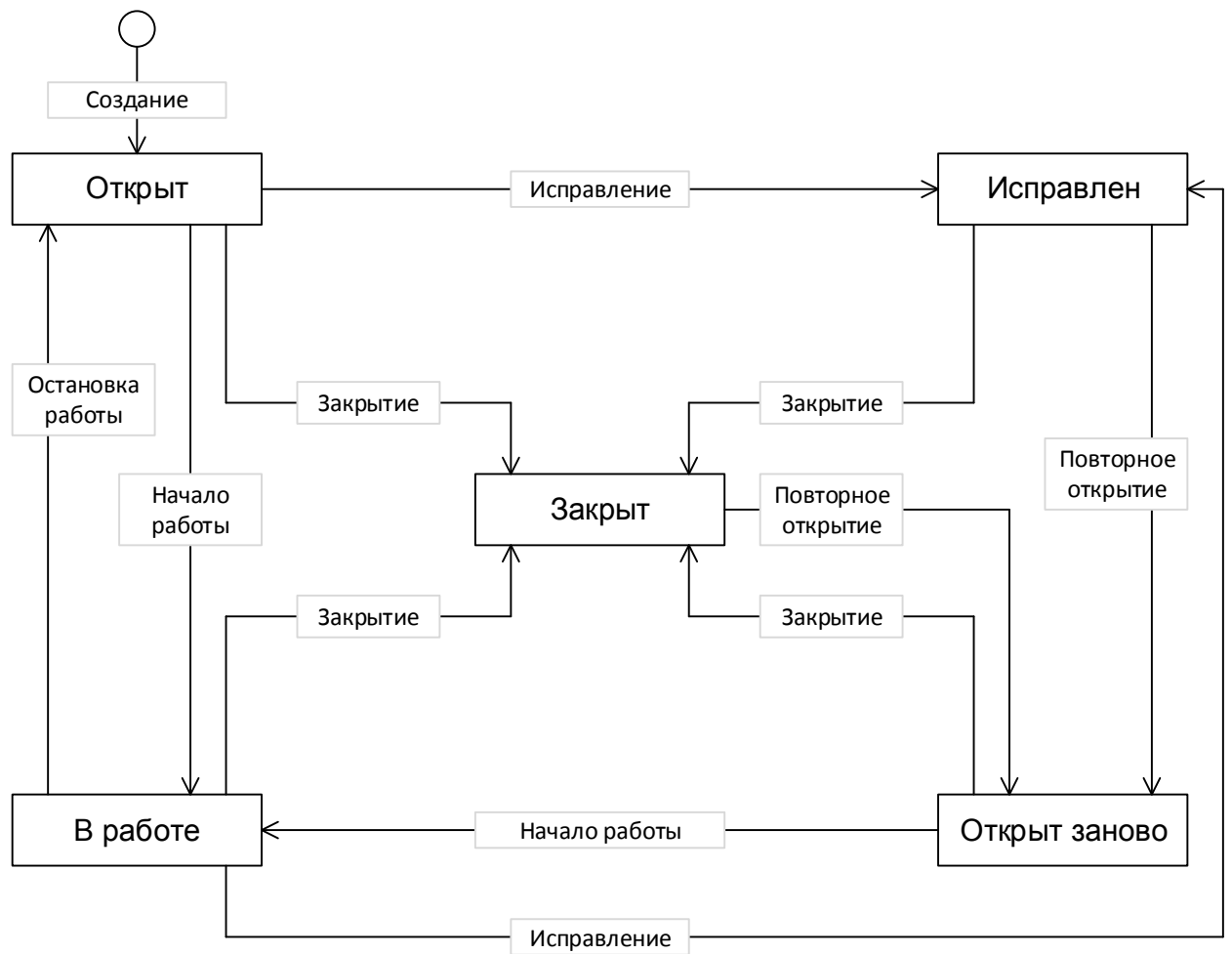


Рисунок 2.5.d — Жизненный цикл отчёта о дефекте в JIRA

2.5.3. 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.

Общий вид всей структуры отчёта о дефекте представлен на рисунке 2.5.е.

Идентификатор	Краткое описание	Подробное описание	Шаги по воспроизведению
19	Бесконечный цикл обработки входного файла с атрибутом «только для чтения»	<p>Если у входного файла выставлен атрибут «только для чтения», после обработки приложению не удаётся переместить его в каталог-приёмник: создаётся копия файла, но оригинал не удаляется, и приложение снова и снова обрабатывает этот файл и безуспешно пытается переместить его в каталог-приёмник.</p> <p>Ожидаемый результат: после обработки файл перемещён из каталога-источника в каталог-приёмник.</p> <p>Фактический результат: обработанный файл копируется в каталог-приёмник, но его оригинал остаётся в каталоге-источнике.</p> <p>Требование: ДС-2.1.</p>	<ol style="list-style-type: none">1. Поместить в каталог-источник файл допустимого типа и размера.2. Установить данному файлу атрибут «только для чтения».3. Запустить приложение. <p>Дефект: обработанный файл появляется в каталоге-приёмнике, но не удаляется из каталога-источника, файл в каталоге-приёмнике непрерывно обновляется (видно по значению времени последнего изменения).</p>

Воспроизводимость	Важность	Срочность	Симптом	Возможность обойти	Комментарий	Приложения
Всегда	Средняя	Обычная	Некорректная операция	Нет	Если заказчик не планирует использовать установку атрибута «только для чтения» файлам в каталоге-источнике для достижения неких своих целей, можно просто снимать этот атрибут и спокойно перемещать файл.	-

Рисунок 2.5.е — Общий вид отчёта о дефекте

	Задание 2.5.а: как вы думаете, почему этот отчёт о дефекте можно по формальным признакам отклонить с резолюцией «не является дефектом»?
---	--

Теперь рассмотрим каждый атрибут подробно.

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.

Рассмотрим несколько примеров применения этого алгоритма.

Ситуация 1. Тестированию подвергается некое веб-приложение, поле описания товара должно допускать ввод максимум 250 символов; в процессе тестирования оказалось, что этого ограничения нет.

1. Суть проблемы: исследование показало, что ни на клиентской, ни на серверной части нет никаких механизмов, проверяющих и/или ограничивающих длину введенных в поле «О товаре» данных.
2. Исходный вариант подробного описания: в клиентской и серверной части приложения отсутствуют проверка и ограничение длины данных, вводимых в поле «О товаре» на странице <http://testapplication/admin/goods/edit/>.
3. Конечный вариант подробного описания:
 - Фактический результат: в описании товара (поле «О товаре», <http://testapplication/admin/goods/edit/>) отсутствуют проверка и ограничение длины вводимого текста (MAX=250 символов).
 - Ожидаемый результат: в случае попытки ввода 251+ символов выводится сообщение об ошибке.
4. Определение «что, где и при каких условиях случилось»:
 - Что: отсутствуют проверка и ограничение длины вводимого текста.
 - Где: описание товара, поле «О товаре», <http://testapplication/admin/goods/edit/>.
 - При каких условиях: – (в данном случае дефект присутствует всегда, вне зависимости от каких бы то ни было особых условий).
5. Первичная формулировка: отсутствуют проверка и ограничение максимальной длины текста, вводимого в поле «О товаре» описания товара.
6. Сокращение (итоговое краткое описание): нет ограничения максимальной длины поля «О товаре». Английский вариант: no check for «О товаре» max length.

Ситуация 2. Попытка открыть в приложении пустой файл приводит к краху клиентской части приложения и потере несохранённых пользовательских данных на сервере.

1. Суть проблемы: клиентская часть приложения начинает «вслепую» читать заголовок файла, не проверяя ни размер, ни корректность формата, ничего; возникает некая внутренняя ошибка, и клиентская часть приложения некорректно прекращает работу, не закрыв сессию с сервером; сервер закрывает сессию по таймауту (повторный запуск клиентской части запускает новую сессию, так что старая сессия и все данные в ней в любом случае утеряны).
2. Исходный вариант подробного описания: некорректный анализ открываемого клиентом файла приводит к краху клиента и необратимой утере текущей сессии с сервером.
3. Конечный вариант подробного описания:
 - Фактический результат: отсутствие проверки корректности открываемого клиентской частью приложения файла (в том числе пустого) приводит к краху клиентской части и необратимой потере текущей сессии с сервером (см. BR852345).
 - Ожидаемый результат: производится анализ структуры открываемого файла; в случае обнаружения проблем отображается сообщение о невозможности открытия файла.
4. Определение «что, где и при каких условиях случилось»:
 - Что: крах клиентской части приложения.
 - Где: – (конкретное место в приложении определить едва ли возможно).
 - При каких условиях: при открытии пустого или повреждённого файла.

5. Первичная формулировка: отсутствие проверки корректности открываемого файла приводит к краху клиентской части приложения и потере пользовательских данных.
6. Сокращение (итоговое краткое описание): крах клиента и потеря данных при открытии повреждённых файлов. Английский вариант: client crash and data loss on damaged/empty files opening.

Ситуация 3. Крайне редко по совершенно непонятным причинам на сайте нарушается отображение всего русского текста (как статических надписей, так и данных из базы данных, генерируемых динамически и т.д. — всё «становится вопросиками»).

1. Суть проблемы: фреймворк, на котором построен сайт, подгружает специфические шрифты с удалённого сервера; если соединение обрывается, нужные шрифты не подгружаются, и используются шрифты по умолчанию, в которых нет русских символов.
2. Исходный вариант подробного описания: ошибка загрузки шрифтов с удалённого сервера приводит к использованию локальных несовместимых с требуемой кодировкой шрифтов.
3. Конечный вариант подробного описания:
 - Фактический результат: периодическая невозможность загрузить шрифты с удалённого сервера приводит к использованию локальных шрифтов, несовместимых с требуемой кодировкой.
 - Ожидаемый результат: необходимые шрифты подгружаются всегда (или используется локальный источник необходимых шрифтов).
4. Определение «что, где и при каких условиях случилось»:
 - Что: используются несовместимые с требуемой кодировкой шрифты.
 - Где: — (по всему сайту).
 - При каких условиях: в случае ошибки соединения с сервером, с которого подгружаются шрифты.
5. Первичная формулировка: периодические сбои внешнего источника шрифтов приводят к сбою отображения русского текста.
6. Сокращение (итоговое краткое описание): неверный показ русского текста при недоступности внешних шрифтов. Английский вариант: wrong presentation of Russian text in case of external fonts inaccessibility.

Для закрепления материала ещё раз представим эти три ситуации в виде таблицы 2.5.а.

Таблица 2.5.a — Проблемные ситуации и формулировки кратких описаний дефектов

Ситуация	Русский вариант краткого описания	Английский вариант краткого описания
Тестированию подвергается некое веб-приложение, поле описания товара должно допускать ввод максимум 250 символов; в процессе тестирования оказалось, что этого ограничения нет.	Нет ограничения максимальной длины поля «О товаре».	No check for «О товаре» max length.
Попытка открыть в приложении пустой файл приводит к краху клиентской части приложения и потере несохранённых пользовательских данных на сервере.	Крах клиента и потеря данных при открытии повреждённых файлов.	Client crash and data loss on damaged/empty files opening.
Крайне редко по совершенно непонятным причинам на сайте нарушается отображение всего русского текста (как статических надписей, так и данных из базы данных, генерируемых динамически и т.д. — всё «становится вопросами»).	Неверный показ русского текста при недоступности внешних шрифтов.	Wrong presentation of Russian text in case of external fonts inaccessibility.

Возвращаемся к рассмотрению полей отчёта о дефекте.

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:

Если в систему входит администратор, на странице приветствия отсутствует логотип.
 Фактический результат: логотип отсутствует в левом верхнем углу страницы.
 Ожидаемый результат: логотип отображается в левом верхнем углу страницы.
 Требование: R245.3.23b.

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:

1. Открыть <http://testapplication/admin/login/>.
2. Авторизоваться с именем «defaultadmin» и паролем «dapassword».

Дефект: в левом верхнем углу страницы отсутствует логотип (вместо него отображается пустое пространство с надписью «logo»).

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 воспроизведётся 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 «ОК»/«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 «насколько 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 очередности. 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.

Несколько дополнительных рассуждений о важности и срочности стоит рассмотреть отдельно.

Один из самых частых вопросов относится к тому, какая между ними связь. Никакой. Для лучшего понимания этого факта можно сравнить важность и срочность с координатами X и Y точки на плоскости. Хоть «на бытовом уровне» и кажется, что дефект с высокой важностью следует исправить в первую очередь, в реальности ситуация может выглядеть совсем иначе.

Чтобы проиллюстрировать эту мысль подробнее, вернёмся к перечню градаций: заметили ли вы, что для разных степеней важности примеры приведены, а для разных степеней срочности — нет? И это не случайно.

Зная суть проекта и суть дефекта, его важность определить достаточно легко, т.к. мы можем проследить влияние дефекта на критерии качества, степень выполнения требований той или иной важности и т.д. Но срочность исправления дефекта можно определить только в конкретной ситуации.

Поясним на жизненном примере: насколько для жизни человека важна вода? Очень важна, без воды человек умирает. Значит, важность воды для человека можно оценить как критическую. Но можем ли мы ответить на вопрос «Как быстро человеку нужно выпить воды?», не зная, о какой ситуации идёт речь? Если рассматриваемый человек умирает от жажды в пустыне, срочность будет наивысшей. Если он просто сидит в офисе и думает, не попить ли чая, срочность будет обычной или даже низкой.

Вернёмся к примерам из разработки программного обеспечения и покажем четыре случая сочетания важности и срочности в таблице 2.5.b.

Таблица 2.5.b — Примеры сочетания важности и срочности дефектов

		Важность	
		Критическая	Низкая
Срочность	Наивысшая	Проблемы с безопасностью во введённом в эксплуатацию банковском ПО.	На корпоративном сайте повреждена картинка с фирменным логотипом.
	Низкая	В самом начале разработки проекта обнаружена ситуация, при которой могут быть повреждены или вовсе утеряны пользовательские данные.	В руководстве пользователя обнаружено несколько опечаток, не влияющих на смысл текста.

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⁽⁸¹⁾).
- Ошибка локализации (localization issue) — что-то в приложении не переведено или переведено неверно на выбранный язык интерфейса (см. локализационное тестирование⁽⁸⁴⁾).
- 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⁽⁸⁶⁾ и тестирование масштабируемости⁽⁸⁷⁾).
- Низкая производительность (low performance) — выполнение неких операций занимает недопустимо большое время (см. тестирование производительности⁽⁸⁶⁾).
- 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 «ОК» 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. Например, крах системы очень часто ведёт к потере или повреждению данных. Но в большинстве инструментальных средств управления отчётами о дефектах значение поля «Симптом» выбирается из списка, и потому нет возможности указать два и более симптома одного дефекта. В такой ситуации рекомендуется выбирать либо симптом, который лучше всего описывает суть ситуации, либо «наиболее опасный» симптом (например, недружественное поведение, состоящее в том, что приложение не запрашивает подтверждения перезаписи существующего файла, приводит к потере данных; здесь «потеря данных» куда уместнее, чем «недружественное поведение»).

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.

Приложения (attachments) — представляет собой не столько поле, сколько список прикрепленных 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 прикладывайте 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).
 - 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.

Для более глубокого понимания принципов оформления отчётов о дефектах рекомендуется прямо сейчас ознакомиться с главой «Типичные ошибки при написании отчётов о дефектах»⁽¹⁹⁷⁾.

2.5.4. 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 tool³¹⁹) очень много³²⁰, к тому же многие компании разрабатывают свои внутренние средства решения этой задачи. Зачастую такие инструментальные средства являются частями инструментальных средств управления тестированием^[125].

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.

Здесь вполне сознательно не приведено никакого сравнения или подробного описания — подобных обзоров достаточно в Интернете, и они стремительно устаревают по мере выхода новых версий обозреваемых продуктов.

Но интерес представляют отдельные особенности интерфейса, на которые мы обратим внимание в каждом из примеров (важно: если вас интересует подробное описание каждого поля, связанных с ним процессов и т.д., обратитесь к официальной документации — здесь будут лишь самые краткие пояснения).

³¹⁹ **Defect management tool, Incident management tool.** A tool that facilitates the recording and status tracking of defects and changes. They often have workflow-oriented facilities to track and control the allocation, correction and re-testing of defects and provide reporting facilities. See also incident management tool. [ISTQB Glossary]

³²⁰ «Comparison of issue-tracking systems», Wikipedia [http://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems]