Тестирование ПО. Уровень 1

**Основные понятия в тестировании**

Кто такой тестировщик? Что такое тестирование? Что такое качество? Стандарт ISO/IEC 25010. Что такое дефект? Правила оформления дефектов. Состав команды разработки. Направления развития.

**Оглавление**

[Введение](#_c75j823j481j)

[Тестировщик и тестирование](#_tyjcwt)

[Кто такой тестировщик?](#_3dy6vkm)

[Что такое тестирование?](#_1t3h5sf)

[Что такое качество?](#_ybu97jj96tj5)

[Международный стандарт качества ISO/IEC 25010](#_tydynsve21dp)

[Ошибка, дефект и сбой / отказ](#_exsbve9d66uu)

[отчёт о дефекте](#_fy61uulpkezd)

[Жизненный цикл дефекта](#_mhnw1cpxmay9)

[Атрибуты отчёта о дефекте](#_4dgp8broykno)

[Правила оформления дефектов](#_c01lwyodamo7)

[Общие правила при оформлении отчётов о дефектах:](#_3dlf8d7m4s1m)

[Дополнительно об отчётах](#_22hzm245phh)

[Пример оформления отчёта о дефекте](#_5n64d772m1k8)

[Тестировщик в составе команды разработки](#_dm2x8k7ol6op)

[Пути развития в тестировании](#_s5pcf7h9u1a6)

[Рабочий день тестировщика](#_mk9ew8ssj14e)

[Глоссарий](#_tyjcwt)

[Теория урока](#_3dy6vkm)

[Сложности в теме](#_4d34og8)

[Практическое задание](#_clx8y3gby0rl)

[Дополнительные материалы](#_3rdcrjn)

[Используемые источники](#_lnxbz9)

# Введение

В рамках теоретического курса вы сможете освоить следующие навыки:

1. Владение основной терминологией.
2. Поиск дефектов и составление отчётов по найденным дефектам.
3. Использование техник тест-дизайна для составления тест-кейсов.
4. Ознакомитесь с основными видами и подходами ручного тестирования.
5. Узнаете основные виды тестовой документации.
6. Получите первые практические навыки тестирования приложений.

**Регламент курса**

1. В курс входит 8 занятий по 1,5–2 часа.
2. После каждого занятия вам будет дано практическое задание.
3. Срок сдачи задания — за 4 часа до начала следующего урока.
4. Видеозапись лекции будет выкладываться автоматически. В зависимости от нагрузки на платформу, запись может быть прикреплена к уроку в течение нескольких часов.
5. Вопросы преподавателю вы можете задавать в комментариях к уроку либо в личных сообщениях.
6. Также за вами закреплён наставник, к которому вы тоже можете обращаться.

В качестве рекомендации — несколько советов, придерживаясь которых, вы получите максимальный результат от обучения:

1. Старайтесь выполнять домашнее задание в полном объёме. Так вы получите практический опыт.
2. Используйте текстовые редакторы либо OneNote для ведения записей.
3. Создайте для себя глоссарий определений и терминов, который вы могли бы использовать в работе или для повторения.
4. Постарайтесь составить резюме как можно раньше. После каждого урока анализируйте, можете ли вы добавить в резюме что-то новое, что вы действительно поняли и закрепили. Но имейте в виду, закреплять надо и дальше.
5. Практику старайтесь закреплять на краудтестинговых площадках, таких как uTest.com или test.io. Их на самом деле больше, но эти — одни из самых популярных.
6. Структурируйте и фильтруйте поступающую информацию. Пользуйтесь только проверенными источниками.
7. Старайтесь помогать советом и делом своим сокурсникам.

В рамках курса рекомендуется использовать определения из глоссария ISTQB. Ссылки на ресурсы вы найдёте в конце методички.

В первом занятии будут рассмотрены следующие темы:

1. Кто такой тестировщик.
2. Что такое тестирование.
3. Что такое валидация и верификация.
4. Что такое качество ПО.
5. Что такое качество по международному стандарту.
6. Что такое ошибки, дефект и сбой.
7. Что такое жизненный цикл дефекта на примере баг-трекинговой системы Jira.
8. Что такое отчёт о дефекте и как его составлять.
9. Основные правила работы с дефектами.
10. Пути развития в тестировании и Quality Assurance.
11. Состав команды разработки.

# Тестировщик и тестирование

## Кто такой тестировщик?

Рассмотрим два определения «тестировщик» — в соответствии с ISTQB и с профстандартом РФ.

В соответствии с ISTQB:

**Тестировщик (tester)** — опытный специалист, принимающий участие в тестировании компонента или системы.

В соответствии с профстандартом (Приказ Минтруда России от 11.04.2014 «Об утверждении профессионального стандарта «Специалист по тестированию в области информационных технологий»):

**Специалист по тестированию ПО** — специалист, основная цель деятельности которого — оценка качества разрабатываемого программного обеспечения путём проверки соответствия продукта заявленным требованиям, сбора и передачи информации о несоответствиях.

Определение профстандарта говорит, что основная задача тестировщика — проверка соответствия продукта заявленным требованиям. Или, если иначе, — тестирование.

Здесь мы можем выделить для себя следующий термин, с которым мы очень часто сталкиваемся на работе, а именно — требования.

**Требование** — это описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной̆ для пользователя задачи.

Надо помнить, что требования могут быть явными, или четко выраженными. А могут быть и неявными, которые не озвучены вслух или не прописаны в документе, но подразумеваются.

## Что такое тестирование?

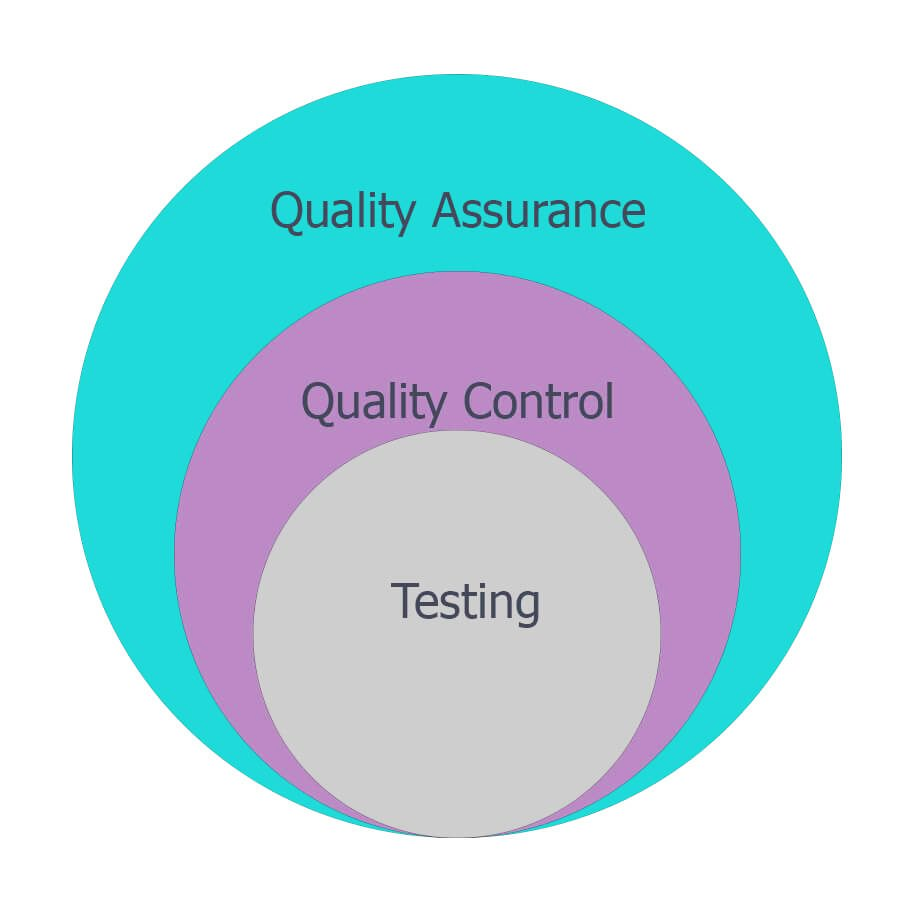
Что такое тестирование в соответствии с ГОСТ Р 56920-2016?

**Тестирование** — набор операций, проводимых для выявления и/или оценки свойств одного или более элементов тестирования (ГОСТ Р 56920-2016 — Системная и программная инженерия. Тестирование программного обеспечения).

Теперь посмотрим определение, которое дает нам ISTQB.

**Тестирование** — процесс, содержащий все активности жизненного цикла — как динамические, так и статические, — касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ, чтобы определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения дефектов.

Тестирование — один из аспектов контроля качества. Он входит в обеспечение качеством.



**Testing (тестирование)** — разработка и прохождение тест-кейсов, локализация дефектов и пр.

**QC (контроль качества продукта)** — анализ результатов тестирования и качества билдов в процессе разработки.

**QA (обеспечение качества)** — изучение возможностей по изменению и улучшению процесса разработки, улучшению коммуникаций в команде и т. д.

Сейчас необходимо знать и понимать разницу между тем, что такое тестирование и что такое обеспечение качества. Чаще всего встречаются вакансии тестировщика либо инженера по обеспечению качества.

**Чем занимается тестировщик**:

1. Тестирование и анализ продукта.
2. Составление тестов для дальнейшего тестирования.
3. Взаимодействие с разработчиками и в целом с командой для анализа и исправления дефектов.
4. Составление отчётной документации о результатах тестирования на определённых этапах.

**Чем занимается инженер по обеспечению качества (помимо тестирования):**

1. Формирование критериев качества и контроль за их соблюдением.
2. Внедрение стандартов.
3. Изучение возможности по изменению и улучшению процесса разработки.
4. Обучение специалистов.
5. Улучшение коммуникаций в команде.

В процессе тестирования часто встречаются два термина — валидация и верификация.

**Верификация (verification)** — оценка системы или её компонентов, чтобы определить, удовлетворяют ли результаты текущего этапа разработки условиям, сформулированным в его начале. Т. е., выполняются ли цели, сроки, задачи по разработке проекта, определённые в начале текущей фазы. Процесс оценки соответствия продукта явным требованиям (спецификациям).

**Валидация (validation)** — определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, его требованиям к системе.

Верификация фокусируется на вопросе «Соответствует ли результат определённой технической спецификации?».

Валидация ориентирована на рассмотрение продукта с точки зрения пользователя и сфокусирована на вопросе «Соответствует ли результат поставленной цели?». Например, дает ли результат решение проблемы?

## Что такое качество?

Качество — понятие, которое многие могут понимать по-своему.

По ISTQB:

**Качество** — степень, с которой какой-то компонент, система или процесс отвечает определённым требованиям и/или требованиям и ожиданиям пользователя/заказчика.

Мы видим, что качество связано с требованиями, которые явно (или неявно) предъявляются к продукту.

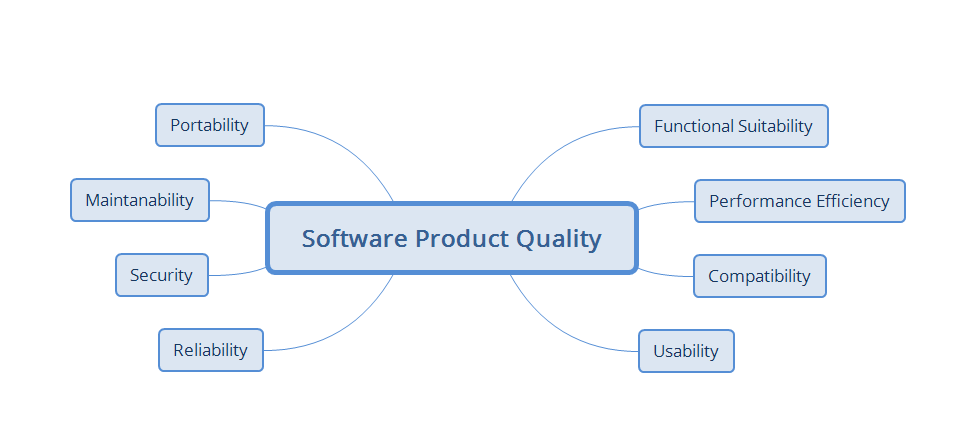
Иногда заказчик может думать, что очевидные вещи и так понятны. Иногда он сам не может четко сформулировать определённые технические аспекты. Поэтому очень важна роль технических специалистов, которые могут подсказать заказчику, как сформировать требования к продукту и понять, что именно необходимо ему и конечному пользователю.

В целом, можно сказать, что продукт качественный, если он имеет определённую ценность для пользователя, то есть решает какую-то определённую задачу.

Таким образом **качество продукта** — это совокупность функциональных возможностей и характеристик ПО, которые удовлетворяют заявленным или подразумеваемым требованиям.

# Международный стандарт качества ISO/IEC 25010

Сейчас действующий международный стандарт качества — ISO/IEC 25010, который вытеснил стандарт ISO/IEC 9126-1 и включает в себя 8 характеристик качества и 31 подхарактеристику.





Рассмотрим 8 характеристик этого стандарта.

**1.** **Функциональная пригодность** (здесь — функциональность) — степень, в которой программный продукт или система обеспечивает функции, удовлетворяющие заявленным и подразумеваемым потребностям при использовании в определённых условиях.

Эта характеристика состоит из следующих подхарактеристик:

* Функциональная полнота — степень, в которой набор функций охватывает все указанные задачи и задачи пользователя.
* Функциональная правильность — степень, в которой продукт или система обеспечивает правильные результаты с необходимой степенью точности.
* Функциональная целесообразность — степень, в которой функции облегчают выполнение указанных задач и целей.

К примеру, мы определяем, что через интернет-магазин можно купить то, что он продаёт. Можно ли сделать доставку товара на дом, как заявляется на его странице?

**2.** **Эффективность производительности** (здесь — эффективность) — характеристика, которая представляет производительность относительно количества ресурсов, используемых в указанных условиях.

Эта характеристика состоит из следующих подхарактеристик:

* Поведение в зависимости от времени (эффективность во времени) — степень, в которой время отклика и его обработки, а также пропускная способность продукта или системы при выполнении своих функций соответствуют требованиям.
* Эффективность использование ресурсов — степень соответствия количества и типов ресурсов, используемых продуктом или системой при выполнении своих функций.
* Вместимость — степень, в которой максимальные пределы продукта или системного параметра соответствуют требованиям.

К примеру, сколько ваше приложение потребляет серверных ресурсов.

**3.** **Надёжность** — степень, в которой система, продукт или компонент выполняет указанные функции при определённых условиях в течение определённого времени.

Эта характеристика состоит из следующих подхарактеристик:

* Завершённость — степень, в которой система, продукт или компонент удовлетворяют потребности в надёжности при нормальной работе.
* Доступность — степень работоспособности и доступности системы, продукта или компонента, когда это необходимо для использования.
* Отказоустойчивость — степень, в которой система, продукт или компонент работает, как предполагалось, несмотря на наличие аппаратных или программных сбоев.
* Восстанавливаемость — степень, в которой, в случае прерывания или сбоя, продукт или система могут восстановить данные, на которые непосредственно влияют, и желаемое состояние системы.

К примеру, ваш интернет-магазин может поддерживать безотказную работу. Способен ли он быстро восстанавливать работоспособность в случае отказа или при падении сайта, к примеру, во время чёрной пятницы или распродаж, когда количество посетителей превысило норму и ваш сайт упал?

**4.** **Удобство использования** — это степень, в которой продукт или система могут использоваться для достижения указанных целей с эффективностью, действенностью и удовлетворением в указанном контексте использования.

Эта характеристика состоит из следующих подхарактеристик:

* Соответствие узнаваемости — степень, в которой пользователи могут распознать, соответствует ли продукт или система их потребностям.
* Обучаемость — степень, с которой будущим пользователям легко будет работать с разрабатываемым приложением. Или насколько легко его будет изучить.
* Работоспособность — степень, в которой продукт или система имеет атрибуты, облегчающие управление и контроль.
* Защита пользователя от ошибок — степень, в которой система защищает пользователей от ошибок.
* Эстетика пользовательского интерфейса — степень, в которой пользовательский интерфейс позволяет приятное и результативное взаимодействие.
* Доступность — степень, в которой продукт или система с самым широким диапазоном характеристик и возможностей для достижения указанной цели в заданном контексте использования могут использоваться людьми.

К примеру, пользователям должно быть интуитивно понятно, как пользоваться вашим интернет-магазином для покупок.

**5.** **Совместимость** — степень, в которой продукт, система или компонент могут обмениваться информацией с другими продуктами, системами или компонентами, и/или выполнять требуемые функции, совместно используя ту же аппаратную или программную среду.

Эта характеристика состоит из следующих подхарактеристик:

* Сосуществование — степень, в которой продукт может эффективно выполнять требуемые функции при совместном использовании общей среды и ресурсов с другими продуктами без вредного воздействия на любой другой продукт.
* Совместимость — степень, в которой два или более продукта, системы или компонента могут обмениваться информацией и использовать её.

К примеру, ваш интернет магазин может работать с платежными системами или другими внешними системами.

**6.** **Безопасность** — степень, в которой продукт или система защищает информацию и данные таким образом, чтобы лица или другие продукты, или системы имели степень доступа к данным, соответствующую их типам и уровням авторизации.

Эта характеристика состоит из следующих подхарактеристик:

* Конфиденциальность — степень, в которой продукт или система гарантируют, что данные доступны только тем, кому разрешен доступ.
* Целостность — степень, в которой система, продукт или компонент предотвращают несанкционированный доступ, модификацию компьютерных программ или данных.
* Отслеживаемость или невозможность отказаться (non-repudiation) — степень, в которой может быть доказано, что действия или события могут иметь место, так что они не будут отклонены позже.
* Ответственность (Accountability) — степень, в которой действия объекта могут быть однозначно прослежены до него.

Примечание: В оригинале это слово "Accountability - Degree to which the actions of an entity can be traced uniquely to the entity".

Можно перевести по смыслу как "ответственность", поскольку действия объекта могут быть отслежены и объект не сможет отказаться от того, что было им сделано и будет нести ответственность.

* Подлинность — степень, в которой личность субъекта или ресурса может быть доказана как заявленная.

К примеру, защищённость ваших личных данных при создании аккаунта электронной почты.

**7.** **Переносимость** — степень эффективности и результативности, с которой система, продукт или компонент могут быть перенесены из одного аппаратного, программного, операционного или эксплуатационного окружения в другое.

Эта характеристика состоит из следующих подхарактеристик:

* Адаптивность — степень, в которой продукт или система могут быть эффективно и действенно адаптированы к разным или развивающимся аппаратным средствам, программному обеспечению или другим операционным средам или средам использования.
* Простота (легкость установки) — степень эффективности и продуктивности, с которой продукт или система могут быть успешно установлены и/или удалены в определённой среде.
* Заменяемость — степень, в которой продукт может заменить другой указанный программный продукт для той же цели в той же среде.

К примеру, вы можете переносить данные из одной базы данных в другую без их потери.

**8.** **Поддерживаемость**. Эта характеристика представляет степень эффективности и действенности, с которой продукт или система могут быть изменены для его улучшения, исправления или адаптации к изменениям в окружающей среде и в требованиях.

Эта характеристика состоит из следующих подхарактеристик:

* Модульность — степень, в которой система или компьютерная программа состоит из отдельных компонентов, так что изменение одного компонента оказывает минимальное влияние на другие.
* Повторное использование — степень, в которой актив может использоваться более чем в одной системе или при создании других активов.
* Пригодность к анализу — степень эффективности и результативности, с которой можно оценить влияние на продукт или систему предполагаемого изменения одной или нескольких его частей, диагностировать продукт на наличие недостатков или причин отказов, или определить детали, подлежащие модификации.
* Модифицируемость — степень, в которой продукт или система могут быть эффективно и действенно модифицированы без внесения дефектов или ухудшения качества существующего продукта.
* Тестируемость — степень эффективности и результативности, с которой могут быть установлены критерии тестирования для системы, продукта или компонента, и могут быть проведены тесты для определения того, были ли выполнены эти критерии.

К примеру, вы можете дорабатывать интернет-магазин и расширять его функциональные возможности.

# Ошибка, дефект и сбой/отказ

При разработке программного продукта в силу тех или иных причин может быть допущена ошибка, которая приведёт к появлению дефекта в продукте.

**Ошибка** (error, mistake) — действие человека, приводящее к некорректным результатам.

Какая это может быть ошибка?

* Неправильно или неполно сформированные требования.
* Ошибка при выборе архитектурного построения системы приложения.
* Ошибка при написании программного кода.
* Ошибка при составлении дизайна или макета приложения.

То есть мы можем встретить дефекты в разных местах или документах — требованиях, спецификациях, в коде, в архитектуре или дизайне, а также в настройках тестируемого ПО.

Ошибки приводят к появлению дефектов в приложении, которые, в свою очередь, могут приводить к сбоям в работе программы.

Найденные дефекты могут снижать качество продукта.

**Дефект** — это несовершенство или недостаток в работе продукта, когда он не отвечает требованиям или спецификации.

Можно сказать, проще: дефект — расхождение ожидаемого и фактического результата.

**Ожидаемый результат** — поведение системы, которое мы можем взять из требований.

**Фактический результат** — поведение системы, наблюдаемое в процессе тестирования.

**Сбой** (interruption) или **отказ** (failure) — отклонение поведения системы от ожидаемого.

Российский ГОСТ 27.002-89 расшифровывает сбой и отказ следующим образом:

* **Отказ** — событие, заключающееся в нарушении работоспособного состояния приложения.
* **Сбой** — самоустраняющийся или однократный отказ, устраняемый незначительным вмешательством.

Находя дефект в работе приложения, тестировщики или инженеры по обеспечению качества создают отчёты о дефектах.

# Отчёт о дефекте

**Отчёт о дефекте, или bug report** — документ, описывающий ситуацию, которая привела к обнаружению дефекта, с указанием причин и ожидаемого результата.

В отчёте о дефекте описывается и назначается приоритет обнаруженному дефекту.

Можно выделить три основные цели создания отчётов о дефектах:

* предоставить информацию о проблеме,
* назначить приоритет проблеме,
* содействовать устранению проблемы.

В зависимости от полноты и корректности созданного отчёта о дефекте зависит, как быстро разработчик сможет выяснить корень проблемы (root cause) и приступить к её устранению. Поэтому на многих проектах стараются придерживаться определённой стилистики и требований к написанию\составлению отчётов о дефектах.

Пример — работа с баг-трекинговой системой Redmine.

В Redmine можно выделить следующие основные поля для отчёта о дефекте:

1. Тип задачи.
2. Тема.
3. Описание.
4. Статус.
5. Приоритет.
6. Назначена или нет.
7. Категория.
8. Версия.
9. Важность.
10. Дата начала.
11. Срок завершения.
12. Оценка временных затрат.
13. Готовность.
14. Модуль.
15. Файлы.

В большинстве баг-трекинговых систем поля для отчётов о дефектах аналогичны. Кроме того, можно дополнительно добавлять необходимые поля в соответствии с потребностями проекта.

Вы можете с легкостью установить на свою локальную машину Redmine. Для этого [скачайте установочный файл](https://bitnami.com/stack/redmine/installer), соответствующий вашей операционной системе, и запустите установку.

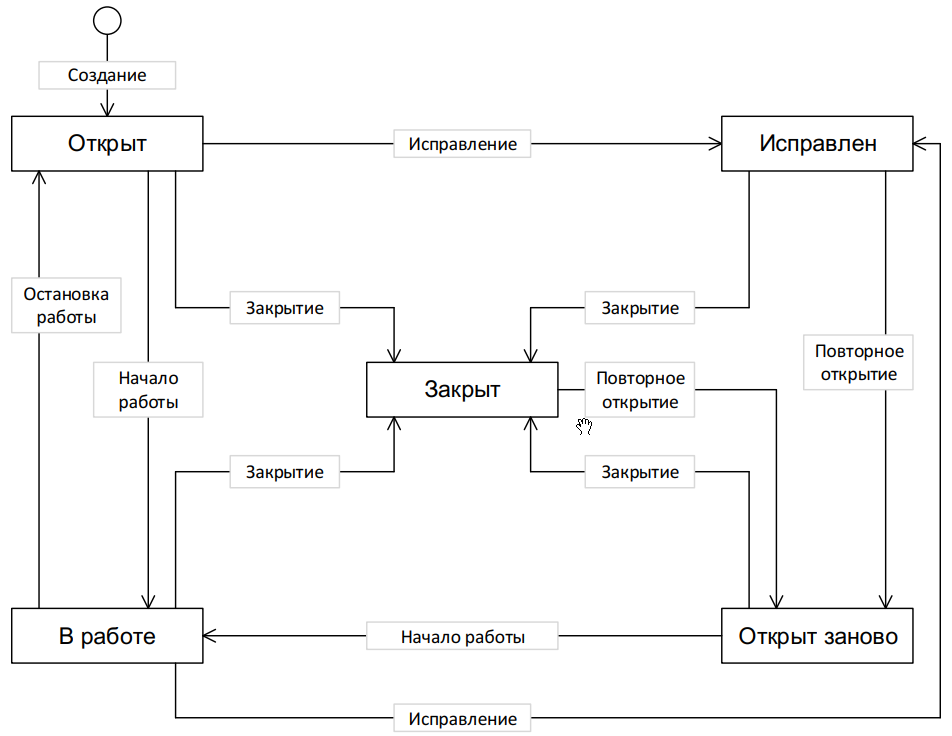
Если что-то не получилось или вы хотите узнать больше, смотрите [дополнительную информацию по установке](https://docs.bitnami.com/installer/apps/redmine/).

# Жизненный цикл дефекта

Каждый дефект имеет свой жизненный цикл. Рассмотрим жизненный цикл дефекта на примере баг-трекинговой системы JIRA.

Основные этапы это:

1. Дефект **ОТКРЫТ**, когда его обнаружили и создали отчёт о нем.
2. Дефект **В РАБОТЕ**, когда он назначен на разработчика и тот начал работу над ним.
3. Дефект **ИСПРАВЛЕН**, когда разработчик внес изменения (исправление или fix) в программный код и сохранил эти изменения в общем репозитории кода.
4. Дефект **ЗАКРЫТ**, когда проверили, что он действительно исправлен в новой версии продукта.
5. Дефект **ОТКРЫТ ЗАНОВО** в случае, если ранее исправленный дефект снова был обнаружен в системе.



# Атрибуты отчёта о дефекте

Среди основных атрибутов дефекта можно выделить следующие:

1. Уникальный идентификатор (ID).
2. Тема (краткое описание) — Summary.
3. Подробное описание — Description.
4. Шаги для воспроизведения — Steps To Reproduce.
5. Фактический результат — Actual result.
6. Ожидаемый результат — Expected result.
7. Вложения — Attachments.
8. Серьёзность дефекта (важность) — Severity.
9. Приоритет дефекта (срочность) — Priority.
10. Статус — Status.

**Уникальный идентификатор (ID)**. Присваивается автоматически (может содержать в себе данные о требовании, на которое ссылается дефект).

Такой идентификатор уникален в баг-трекинговой системе.

К примеру, ID = 1254\_17.

**Тема (краткое описание или Summary)** — кратко сформулированная суть дефекта по правилу «Что? Где? Когда?».

В идеале, очень хорошо, если из названия дефекта можно понять, что и где сломано, и при каких условиях это ломается.

К примеру: «Обязательные поля для заполнения не отмечены на форме регистрации после нажатия на кнопку «Отправить».

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

**Подробное описание (Description)**. Более широкое описание сути дефекта (при необходимости).

К примеру, «Обязательные поля для заполнения не отмечены на форме регистрации после нажатия на кнопку «Отправить» (отсутствует выделение поля красной рамкой и сообщение об ошибки внизу поля)».

**Шаги для воспроизведения (Steps To Reproduce)**. Последовательное описание действий, которые привели к выявлению дефекта (которые необходимо выполнить для воспроизведения дефекта). Описываются максимально подробно, с указанием конкретных вводимых значений.

Они похожи на шаги тест-кейса, и порой могут браться из тест-кейсов, если дефект был обнаружен при его прохождении.

Как правило, шаги рекомендуется прописывать максимально подробно, указывая вводимые значения.

Пример:

1. Перейти на сайт (доменное имя сайта).
2. Нажать на кнопку «Регистрация».
3. Нажать на кнопку «Отправить».
4. Обратить внимание на форму регистрации.

**Фактический результат (Actual result)**. Указывается что не так работает, в каком месте продукта и при каких условиях (описывая фактический результат, необходимо ответить на три вопроса: «Что? Где? Когда?».

К примеру, «Обязательные поля для заполнения не отмечены на форме регистрации после нажатия на кнопку «Отправить».

**Ожидаемый результат (Expected result)**. Указывается, как именно должна работать система по мнению тестировщика (основанному на требованиях и прочей проектной документации).

К примеру, «Обязательные поля для заполнения отмечены на форме регистрации после нажатия на кнопку «Отправить».

**Вложения (Attachments)**. Дополнительная информация, которая может помочь определить, в чем заключается проблема.

К примеру, скриншоты, видео или лог-файлы.

**Серьёзность дефекта (важность), или Severity**. Характеризует влияние дефекта на работоспособность приложения. Как правило, её определяет тестировщик.

Один из вариантов градации:

* S1. Блокирующая (Blocker).
* S2. Критическая (Critical).
* S3. Значительная (Major).
* S4. Незначительная (Minor).
* S5. Неудобство (Tweak).
* S6. Текст/опечатка (Text).
* S7. Тривиальная (Trivial).

**Приоритет дефекта (срочность), или Priority**. Указывает на очерёдность выполнения задачи или устранения дефекта. Чем выше приоритет, тем быстрее нужно исправить дефект. Как правило, определяет либо менеджер проекта, либо руководитель отдела разработки.

Один из вариантов градации:

* P1. Высокий (High).
* P2. Средний (Medium).
* P3. Низкий (Low).

Порядок исправления ошибок по их приоритетам: High -> Medium -> Low.

На разных проектах тестировщики могут заполнять только «Важность, а на некоторых заполнять и «Важность», и «Срочность».

**Статус (Status)**. Определяет текущее состояние дефекта. Отражает жизненный цикл дефекта от начального состояния до завершения. Названия статусов дефектов могут быть разными в разных баг-трекинговых системах.

Пример градации:

* Новый (New) — новый баг-репорт.
* Открыт (Opened) — баг-репорт открыт.
* Назначен (Assigned) — баг-репорт назначен разработчику.
* Исправлен (Fixed) — дефект исправлен.
* Ретест (Retest) — дефект исправлен, требуется проверка тестировщиком.
* Открыт снова (Reopened) — дефект проверен тестировщиком, ошибка не исправлена.
* Закрыт (Closed) — дефект проверен тестировщиком, баг-репорт закрыт.

**Дополнительными атрибутами могут быть:**

* **Component (Компонент), или Environment (Среда)**. Указывает, на какой платформе этот дефект воспроизводится (iOS, Android, Windows, Mac и т. д.).
* **Assignee (Назначение)**. Указывается тот, на кого назначается баг-репорт для дальнейшей проверки или исправления.
* **Build (Номер сборки)**. Указывается номер билда, в котором был обнаружен дефект.

# Правила оформления дефектов

**I.** **Заголовок отчёта о дефекте.** **Старайтесь следовать правилу WWW — What? Where? Under Which circumstances? при описании дефекта в теме.**

Отвечая на вопросы «Что? Где? Когда (при каких условиях)?», вы сможете легче сформулировать заголовок для дефекта.

**«Что?»** — что происходит или не происходит согласно спецификации или представлению тестировщика о нормальной работе продукта.

**«Где?»** — в каком месте интерфейса пользователя или архитектуры программного продукта находится проблема

**«Когда?»** — в какой момент работы программного продукта, по наступлению какого события или при каких условиях проблема проявляется

Примеры:

1. Приложение зависает на уровне редактирования данных о пользователе во время сохранения текстового файла размером больше 500 Мб.
2. Данные не сохраняются на форме «Профайл» после нажатия кнопки «Сохранить».
3. Поле «Подтвердите пароль» не является обязательным во время регистрации аккаунта.
4. Текст выпадающего списка отображается за пределами выделенной области на главной странице во время просмотра вкладки меню «Книги».

**II. Шаги воспроизведения:**

* являются самой ценной информацией в отчёте,
* представляют собой руководство к действию для тех, кто будет исправлять проблему,
* в шагах необходимо отвечать на вопрос «Что необходимо сделать?»,
* необходимо коротко написать, что сделать, куда нажимать, не уточняя названия страницы,
* описывается как минимум 2 шага, но не больше 7-8 шагов,
* в шагах необходимо уточнять, на что именно необходимо нажать (ссылку, кнопку, логотип…),
* шаги необходимо писать с заглавной буквы,
* в последнем шаге необходимо указать, на какую область посмотреть, на что обратить внимание,
* иногда для воспроизведения дефекта нужен ряд условий — их можно вынести в «Предусловие».

**III. Фактический и ожидаемый результаты:**

* необходимо описывать информативно (стараясь тоже соблюдать правило «Что? Где? Когда?»),
* сначала фактический результат, затем ожидаемый,
* один результат в одном дефекте,
* результаты следует описывать полными предложениями с подлежащим и сказуемым,
* результаты необходимо писать с заглавной буквы.

# Общие правила при оформлении отчётов о дефектах:

1. Старайтесь следовать правилу WWW — What? Where? Under Which circumstances? при описании дефекта в теме дефекта.
2. Старайтесь быть лаконичными при описании дефекта в теме.
3. Каждый дефект должен быть воспроизведен повторно перед написанием отчёта.
4. Отчёт о дефекте должен быть составлен сразу, не откладывайте на потом.
5. Минимизируйте количество шагов в описании.
6. Пишите техническим языком с применением принятой на проекте терминологии.
7. Прикрепляйте дополнительные файлы (логи, скриншоты, видео).

# Дополнительно об отчётах

1. Старайтесь часть самих логов прикреплять в описании. Это поможет при поиске дефектов по сообщениям об ошибках, которое выдает само приложение.
2. Используя видео, вы можете избежать недопонимания.
3. Прикрепляйте ссылки к требованиям, это поможет избежать споров и сэкономить время.
4. Не плодите дубликаты дефектов — проверьте в баг-трекинговой системе, прежде чем составлять отчёт о дефекте, или спросите у коллег, создавал кто-нибудь такой дефект или нет.
5. Указывайте версию ПО и тестовый стенд (окружение), на котором был обнаружен дефект.
6. Попробуйте воспроизвести дефект следуя вашим собственным шагам.

## Пример оформления отчёта о дефекте

### Дефект 1

**Тема:** сумма скидки прибавляется к основной стоимости книги в корзине Shopping Cart при расчёте итоговой стоимости книги.

**Описание:** при проведении покупки книги прибавляется (а не вычитается) значение скидки к финальной стоимости покупки в корзине для авторизованного пользователя.

**Предварительные условия:** пользователь зарегистрирован.

**Шаги воспроизведения:**

1. Перейти на страницу http://www.sharelane.com/cgi-bin/main.py.
2. Авторизоваться под существующим в системе пользователем.
3. Добавить в корзину книгу.
4. Перейти в корзину.
5. Обратить внимание на расчет итоговой стоимости книги с учетом скидки.

**Фактический результат:** сумма скидки прибавляется к стоимости книги в корзине Shopping Cart при расчёте итоговой стоимости книги.

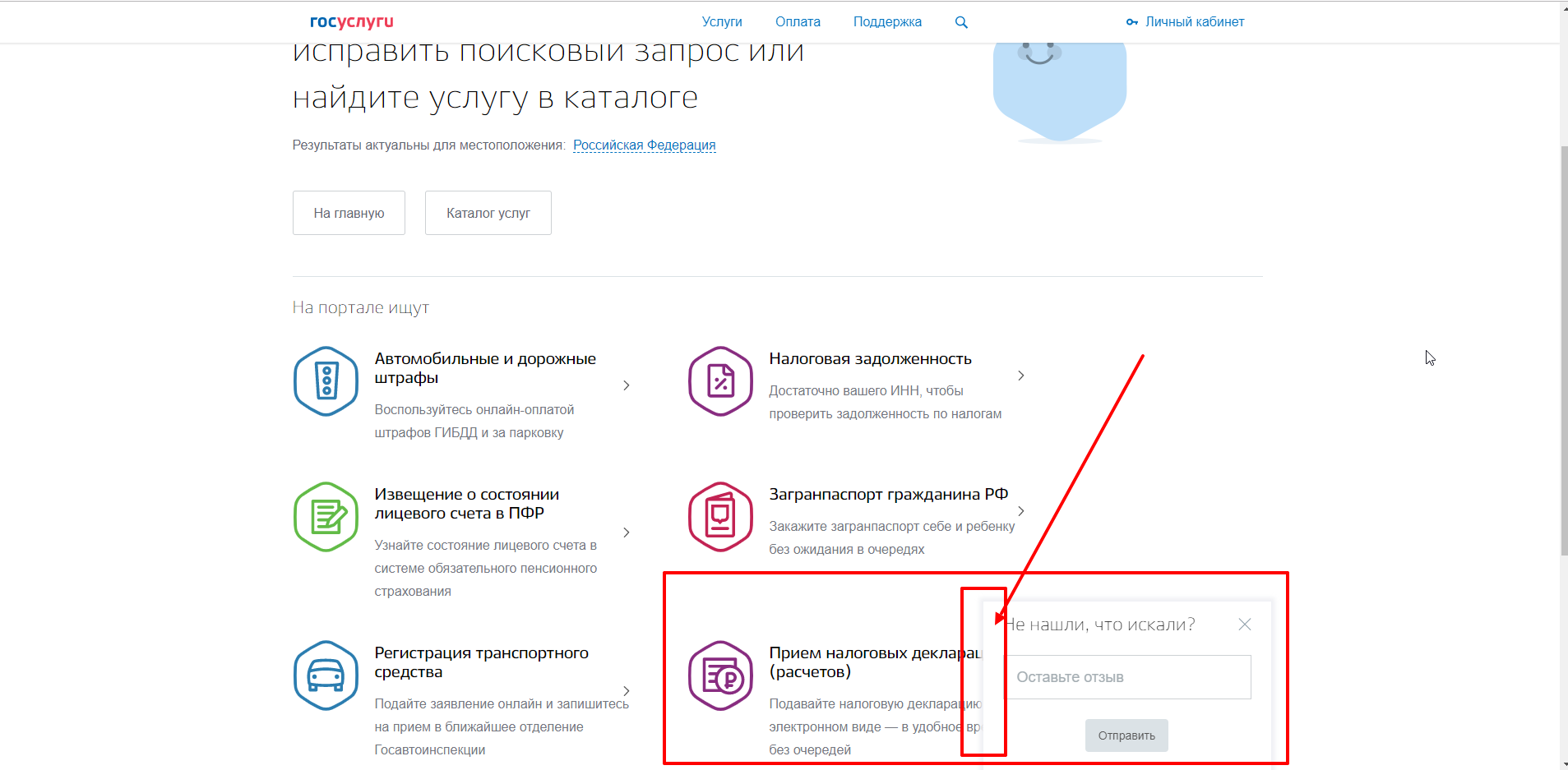
**Ожидаемый результат:** сумма скидки вычитается из стоимости книги в корзине Shopping Cart пользователя при расчете итоговой стоимости книги.

Тестовое окружение: браузер Chrome 77.012.32.

Вложения: https://is.gd/l7MZqy.

### Дефект 2

**Скриншот\_1**



**Тема:** окно «Не нашли, что искали?» перекрывает каталог частых услуг в случае неуспешного поиска для неавторизованного пользователя.

**Описание:** в случае, если неавторизованный пользователь введёт в поле поиска запрос и результат поиска будет пустым, на странице с результатами поиска окно «Не нашли, что искали?» частично перекроет каталог частых услуг.

**Предварительные условия:** пользователь не авторизован.

**Шаги воспроизведения:**

1. Перейти на страницу [https://www.gosuslugi.ru](https://www.gosuslugi.ru/).
2. В поле поиска ввести «gender» и нажать кнопку поиска.

**Фактический результат:** окно «Не нашли, что искали?» частично перекрывает каталог частых услуг.

**Ожидаемый результат:** окно «Не нашли, что искали?» не перекрывает каталог частых услуг, список которых отображается на странице с пустым результатом поиска.

Тестовое окружение: браузер Chrome 77.012.32

Вложения: скриншот 1.

# Тестировщик в составе команды разработки

Как правило, команда разработки состоит из:

* Разработчика/ов.
* Аналитика/ов.
* Тестировщика/ов.
* Менеджера проекта.

Дополнительно бывает DevOps и служба поддержки.

Как правило, аналитики работают непосредственно с заказчиком и обсуждают детали реализации той или иной функциональности, с последующим написанием требований к продукту. Очень часто бывает, что после проверки требований тестировщиками, их приходится уточнять с заказчиком.

На основании требований составляются задачи команде разработки. Разработчики, основываясь на требованиях, создают требуемую функциональность. Иногда при отсутствии какого-либо объяснения разработчик может реализовать функциональность на свое усмотрение или как понял. Если, конечно, он не уточнил у аналитика. В целом, разработчик не должен знать, как работает приложение в целом, если оно большое.

Когда задача реализована, и разработчики сообщили что новая фича/функциональность (feature) внедрена, выпускается новый тестовый билд, то есть сборка, в которую входят реализованные новые фичи. Дальше билд передается команде тестирования. Тестировщики в своей работе основываются на требования. Если поведение программы отличается от ожидаемого, они составляют отчёт о дефекте. Тестировщики, как правило, должны разбираться в готовом продукте. Исключения могут составлять узконаправленные специалисты по тестированию, например, по нагрузочному тестированию, автоматизации или тестированию безопасности. Здесь часто бывает, что они узко ограничены сферой предметной области продукта.

Менеджер проекта следит за общим ходом разработки, за ситуацией на проекте. На нём — ответственность в принятии решений, касающихся всего проекта. Бывает, что необходимо принять решение, стоит ли выпускать функциональность «сырой», зная, что в ней есть дефекты, или нужно договориться с заказчиком об отсрочке и закончить функциональность полностью. Если продукт уже выпущен, и по согласованию с заказчиком компания должна оказывать поддержку, то есть и служба поддержки, которая консультирует при возникновении каких-либо проблем с продуктом. Поэтому они должны хорошо разбираться, как продукт должен работать в целом.

# Пути развития в тестировании

Возможно, вы уже слышали, что дорога в IT через тестирование считается одной из самых лёгких. Но какой карьерный путь может пройти начинающий тестировщик?

Вы начнёте с должности Junior-специалиста, далее идёт Middle-специалист, дальше Senior/старший специалист.

Иногда встречается такое понятие, как Key Tester, то есть ключевой тестировщик на проекте. Это, как правило, опытный специалист, проработавший достаточно давно на проекте и знающий его очень хорошо. Также есть руководитель команды тестирования — QA Team Lead.

Выше руководителя команды тестирования — QA Manager проекта. Не стоит путать с Product Manager, руководителем отдела/департамента по тестированию или Head QA Department. QA Manager проекта следит за качеством продукта на проекте и руководит командами тестирования, если их несколько.

Развитие специалиста по тестированию может быть не строго вверх, но и в какой-то узкой специализации. Это могут быть такие направления, как автоматизация тестирования, нагрузочное тестирование, тестирование безопасности, специалист по базам данных, переход в отдел аналитики и так далее.

Надо хорошо понимать, в каком направлении вы хотите расти и развиваться, какие темы, технологии или направления вас увлекают. Один из распространенных вопросов на собеседованиях «Кем вы видите себя через 1, 3, 5 лет?».

# Рабочий день тестировщика

Новый день на проекте может отличаться от предыдущих. Но в целом можно выделить основные активности в рабочем дне инженера по тестированию:

1. Проверка почты о новых билдах/сборках. Уточнение у разработчиков.
2. Ежедневный митинг Stand Up. Общение с командой.
3. Комментарии на дефекты. Обсуждение дефектов.
4. Задание от Team Lead, обсуждение задач.
5. Написание тест-кейсов, уточнение/обсуждение требований с аналитиком.
6. Тестирование новой функциональности.
7. Ретест (повторное тестирование) исправленных дефектов.
8. Регрессионное тестирование.
9. Создание отчётов о дефектах, обсуждение дефектов с разработчиком.
10. Тестирование требований. Общение с аналитиком, разработчиком.
11. Результаты тестирования, обсуждение результатов с руководителем.

# Глоссарий

**Тестировщик (tester)** — опытный специалист, принимающий участие в тестировании компонента или системы.

**Специалист по тестированию ПО** — специалист, основная цель деятельности которого — оценка качества разрабатываемого программного обеспечения путём проверки соответствия продукта заявленным требованиям, сбора и передачи информации о несоответствиях.

**Требование** — описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной̆ для пользователя задачи.

**Тестирование** — деятельность, выполняемая для оценки и улучшения качества программного обеспечения (Профессиональный стандарт Министерства труда РФ от 11 апреля 2014 года).

**Тестирование** — процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения дефектов (ISTQB).

**Верификация (verification)** — процесс оценки системы или её компонентов, чтобы определить, удовлетворяют ли результаты текущего этапа разработки условиям, сформулированным в начале этого этапа.

**Валидация (validation)** — определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, его требованиям к системе.

**Качество** — степень, с которой какой-то компонент, система или процесс отвечает определённым требованиям и/или требованиям и ожиданиям пользователя/заказчика.

**Качество продукта** — совокупность функциональных возможностей и характеристик ПО, которые удовлетворяют заявленным или подразумеваемым требованиям.

**Ошибка** (error, mistake) — действие человека, приводящее к некорректным результатам.

**Дефект** — несовершенство или недостаток в работе продукта, когда он не отвечает требованиям или спецификации.

**Ожидаемый результат** — поведение системы, которое мы можем взять из требований.

**Фактический результат** — поведение системы, наблюдаемое в процессе тестирования.

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

**Отказ** — событие, заключающееся в нарушении работоспособного состояния приложения.

**Отчёт о дефекте, или bug report** — документ, описывающий ситуацию, которая привела к обнаружению дефекта, с указанием причин и ожидаемого результата.

# Практическое задание

**Используя приложенную таблицу, выполните следующие задания:**

1. Прочитать книгу Святослава Куликова (Software Testing — Base Course (Svyatoslav Kulikov)), страницы 164-204.
2. Ответить на вопросы в приложенной ниже форме.
3. Составить отчёты о дефектах в приложенной форме.
4. \* Создать свой собственный глоссарий определений для подготовки к собеседованиям в будущем.

## Требования к выполненной работе

1. Практическое задание должно быть выполнено в приложенной форме.
2. Расширение файла должно быть формата Excel (XLS, XLSX).
3. Название файла не менять, и вместо ФИО укажите свои фамилию и имя.

# Дополнительные материалы

1. [Что такое ISTQB](https://www.istqb.org/about-as.html):
2. [Глоссарий ISTQB на русском языке](https://www.rstqb.org/ru/istqb-downloads.html).
3. [Статья-обзор за 2019 год о тестировщиках, их уровне ЗП и востребованных навыках](http://software-testing.ru/library/testing/general-testing/3051-salary2018).
4. [Почему грамотное оформление баг-репортов так важно](https://training.qatestlab.com/blog/technical-articles/proper-bug-report/).
5. [Как правильно необходимо описывать шаги в баг-репорте](https://training.qatestlab.com/blog/technical-articles/steps-bug-report/).
6. [Основные атрибуты баг-репорта](https://training.qatestlab.com/blog/technical-articles/attributes-bug-report/).
7. [Баг-репорт: об использовании «некорректных» слов](https://training.qatestlab.com/blog/technical-articles/wrong-words-report/).
8. [Материалы лекции №1 «Вводная часть. что такое баг (дефект)» по курсу «основы тестирования ПО»](https://training.qatestlab.com/blog/course-materials/lecture-testing-introduction/).
9. [Правила описания результатов в баг-репорте](https://training.qatestlab.com/blog/technical-articles/results-bug-report/).
10. [Правила описания багов на английском языке. Использование пассива](https://training.qatestlab.com/blog/technical-articles/passive-voice-bugs/).
11. [С чего начинать описание шагов воспроизведения дефекта (бага)?](https://training.qatestlab.com/blog/technical-articles/steps-bug-reproduction/)
12. [Как правильно составить описание бага в баг-трекере. Принцип «Что? Где? Когда?»](https://training.qatestlab.com/blog/technical-articles/description-bug-tracker/).
13. [Рассказ о себе](https://www.youtube.com/watch?v=fTn2d8uh34Y).
14. [Материалы ISTQB](https://www.rstqb.org/ru/istqb-downloads.html?file=files/content/rstqb/downloads/ISTQB%20Downloads/IS).
15. [Основные положения тестирования](https://testitquickly.com/2010/03/09/testing-basics-by-barancev/).

# Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

1. Software Testing — Base Course (Svyatoslav Kulikov).
2. [ГОСТ 27.002-89 Надежность в технике (ССНТ). Основные понятия. Термины и определения](http://docs.cntd.ru/document/1200004984).
3. [ISTQB Glossary Search](https://glossary.istqb.org/search).
4. [ISO/IEC 25010](https://iso25000.com/index.php/en/iso-25000-standards/iso-25010).
5. [ГОСТ Р 56920-2016 Системная и программная инженерия. Тестирование программного обеспечения. Часть 1. Понятия и определения](http://protect.gost.ru/document1.aspx?control=31&baseC=6&page=4&month=6&year=2016&search=&id=203397).