Введение в тестирование

**Основные понятия**

# На этом уроке

* узнаем, что такое тестирование ПО;
* познакомимся с основными понятиями и определениями в тестировании;
* узнаем, как определяется качество ПО;
* научимся определять дефекты в ПО;
* узнаем, как проходит рабочий день тестировщика.

# Оглавление

[Глоссарий](#_xj21gfokfhg2)

[Тестирование](#_kyj76o30cajs)

[Качество ПО](#_2et92p0)

[Дефект, ошибка, сбой, отказ](#_tyjcwt)

[Отчёт о дефекте](#_3dy6vkm)

[Атрибуты отчёта о дефекте](#_4wgbjvb4vt96)

[Рабочий день тестировщика](#_1t3h5sf)

[Пути развития тестировщика](#_4d34og8)

[Команда IT-проекта](#_2s8eyo1)

[Дополнительные материалы](#_3rdcrjn)

[Используемые источники](#_lnxbz9)

# Глоссарий

**Тестирование программного обеспечения (Software Testing)** — проверка соответствия реальных и ожидаемых результатов поведения программы, производимая на конечном наборе тестов, выбранном определённым образом.

**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 определения дефектов.

**Тестировщик** — специалист, который занимается тестированием. В его обязанности входит поиск возможных ошибок и сбоев в функционале тестирования объекта, например приложения. Тестировщик моделирует ситуации, вероятные при использовании тестируемого объекта, чтобы потом разработчики могли устранить обнаруженные неполадки.

**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.

**Верификация** — процесс оценки системы или её компонентов, чтобы понять, удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в его начале: выполняются ли цели, сроки, задачи по разработке проекта. Процесс оценки соответствия продукта явным требованиям (спецификациям).

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

**Дефект** — отклонение фактического результата от ожиданий наблюдателя, сформированных на основе требований, спецификаций, иной документации или опыта и здравого смысла.

**Ошибка** — действие человека, приводящее к некорректным результатам.

**Дефект** — недостаток в компоненте или системе, способный привести к ситуации сбоя или отказа.

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

**Отказ** — событие, нарушающее работоспособность приложения.

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

**Отчёт о дефекте** — документ, описывающий и приоритизирующий обнаруженный дефект, а также содействующий его устранению.

# Тестирование

Тестирование — это не академическая дисциплина. В нём нет чётких устоявшихся определений, которые можно собрать в словарь и выучить перед собеседованием. Кроме того, каждый IT-проект уникален, что порождает огромное количество разной, часто противоречивой информации. Поэтому важным в тестировании становится понимание процессов и подходов. Важно не только знать, как называется тот или иной процесс или вид тестирования, а что он из себя представляет и для чего он нужен на проекте, какой результат принесёт его применение. Вы будете встречать много разных определений в процессе обучения. Постарайтесь не верить первому определению, которое увидите, а найти несколько определений одного и того же понятия и на основе этого сделать выводы.

Также стоит ориентироваться на терминологию, которая используется в ISTQB.

**ISTQB (International software testing qualification board)** —Международный совет по тестированию программного обеспечения. Это некоммерческая организация, которая занимается определением различных принципов развития сферы тестирования, таких как, например, правила сертификации тестировщиков и систематизация теоретических основ.

Чтобы стать ISTQB-сертифицированным тестировщиком, надо успешно сдать экзамен. Для этого нужны профессиональные знания в области тестирования и знакомство с основами рабочего процесса тестирования ПО.

Но для начала карьеры в тестировании не нужно сдавать экзамен ISTQB, достаточно уметь пользоваться глоссарием этой организации и обращаться к нему в случае сомнения в каком-либо понятии или определении.

Перейдём к основным понятиям. Есть несколько определений тестирования и специалиста по тестированию (тестировщика).

**Тестирование программного обеспечения (Software Testing)** — проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом.

**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 определения дефектов.

**Тестировщик** — специалист, который занимается тестированием. В его обязанности входит поиск возможных ошибок и сбоев в функционале тестирования объекта, например приложения. Тестировщик моделирует ситуации, вероятные при использовании тестируемого объекта, чтобы потом разработчики могли устранить обнаруженные неполадки.

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

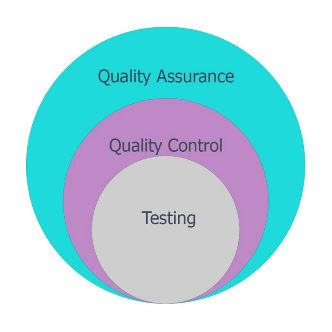
**Специалист по тестированию ПО** — специалист, основная цель деятельности которого — оценка качества разрабатываемого программного обеспечения путём проверки соответствия продукта заявленным требованиям, сбора и передачи информации о несоответствиях.

Также в процессе обучения и работы вам могут встретиться следующие понятия, относящиеся к специалисту по тестированию:

**Tester (тестировщик)** — разработка и прохождение тест-кейсов, локализация дефектов и прочее.

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

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

Тестирование — только один из аспектов обеспечения качества. Тестирование входит в контроль качества, а он входит в процесс обеспечения качества в целом.

Чтобы не вдаваться в детали, разделим тестирование и обеспечение качества, поскольку часто в вакансиях указан либо тестировщик, либо инженер по качеству.

Чем занимается тестировщик:

1. Тестирует и анализирует продукт.
2. Составляет тесты для дальнейшего проведения тестирования.
3. Взаимодействует с разработчиками и командой для анализа и исправления дефектов.
4. Составляет отчётную документацию о результатах тестирования на определённых этапах.

Чем занимается инженер по обеспечению качества:

1. Формирует критерии качества и контролирует их соблюдение.
2. Внедряет стандарты.
3. Изучает возможности изменения и улучшения процесса разработки.
4. Обучает новых специалистов.
5. Улучшает коммуникацию в команде.

Другими словами, тестировщик задействован непосредственно в тестировании, а инженер по обеспечению качества включён в выстраивание и улучшение процессов в команде и на проекте в целом.

Примеры:

1. Изменение процесса уведомления команды тестирования о поставке новой сборки, без задержек и потери времени. Вместо того чтобы каждый раз узнавать у разработчика о новой сборке, тестировщик будет получать уведомление о новой сборке и результатах прогона автотестов на этой сборке.
2. Выбор лучшего инструмента для анализа логов.
3. Введение ретроспектив в команде по окончании определённых этапов: что сделано хорошо; что сделано плохо; что можно улучшить.

Ещё два очень важных понятия в тестировании — валидация и верификация. Они вызывают много проблем у начинающих тестировщиков, поэтому лучше сразу с ними разобраться и хорошо запомнить.

**Верификация (verification)** — это процесс оценки системы или её компонентов, чтобы понять, удовлетворяют ли результаты текущего этапа разработки условиям, сформулированным в его начале. Иными словами, выполняются ли цели, сроки, задачи по разработке проекта, поставленные в начале текущей фазы. Также это оценка соответствия продукта явным требованиям (спецификациям).

Верификация связана с оценкой рабочего продукта, компонента или системы, чтобы определить, соответствует ли он установленным требованиям. Проверка фокусируется на вопросе «Соответствует ли результат спецификации?»

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

Валидация связана с оценкой рабочего продукта, компонента или системы, чтобы определить, отвечает ли продукт ожиданиям пользователя и требованиям. Валидация сфокусирована на вопросе «Соответствует ли результат поставленной цели, например даёт ли результат решение проблемы?»

Валидация ориентирована на рассмотрение продукта с точки зрения пользователя.

К примеру, есть приложение, которое использует для оплаты банковские карты. По требованиям, приложение принимает только карты Visa. Если приложение будет реализовано только для карт Visa, это сразу отсекает огромное количество пользователей. При верификации всё работает правильно, но, валидируя в целом приложение, легко понять, что у пользователей будут разные типы карт и оно не полностью соответствует их ожиданиям.

**Цель тестирования** — проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы.

Обнаружение и исправление ошибок составляет 40—80% общей стоимости разработки ПО. Такие большие средства тратятся на поиск и устранение ошибок, которые есть в любом продукте.

# Качество ПО

Качество ПО, как и качество чего бы то ни было, — понятие достаточно условное и во многом зависит от субъективной точки зрения оценивающего. Но, чтобы хорошо тестировать разрабатываемый продукт, мы должны от чего-то отталкиваться. Рассмотрим базовые определения понятия «качество» в разработке ПО.

**Качество** — степень, в которой какой-то компонент, система или процесс отвечает определённым требованиям и (или) требованиям и ожиданиям пользователя.

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

Заказчик ожидает продукт, к примеру интернет-магазин. Если требования и ожидания были сформированы и собраны поверхностно или не полностью, то по окончании разработки продукта заказчик не всегда может быть удовлетворён его качеством, даже если всё было выполнено в соответствии с его требованиями к продукту.

Иногда заказчик думает, что очевидные вещи и так понятны, или сам не знает, чего хочет. Поэтому очень важна роль технических специалистов, которые могут подсказать заказчику, как сформировать требования к продукту. И эти технические специалисты должны иметь определённые ориентиры в своей работе — стандарт качества.

Основной стандарт качества сейчас — ISO/IEC 25010, вытеснивший стандарт ISO/IEC 9126-1. Он включает в себя восемь характеристик качества и 31 подхарактеристику. Этот стандарт определяет, какие качественные характеристики будут учитываться при оценке свойства программного продукта.

Основные характеристики качества по стандарту ISO/IEC 25010:

1. Функциональность.
2. Эффективность.
3. Надёжность.
4. Удобство использования.
5. Поддерживаемость.
6. Переносимость.
7. Безопасность.
8. Совместимость.



1. **Функциональность** — степень, в которой продукт или система выполняют основные функции, то есть решают те задачи, для которых они были созданы.

Подхарактеристики:

* **функциональная полнота** — степень, в которой набор функций охватывает все указанные задачи и задачи пользователя;
* **функциональная правильность** — степень, в которой продукт или система обеспечивают правильные результаты с необходимой степенью точности;
* **функциональная целесообразность** — степень, в которой функции продукта обеспечивают потребности пользователей (востребованы пользователем).

Пример: основная функция интернет-магазина — продажа товаров. То есть пользователь должен иметь возможность зайти на сайт магазина, посмотреть предложения в каталоге, увидеть характеристики выбранного товара, добавить его в корзину, удалить из неё, оплатить.

1. **Эффективность** —эта характеристика позволяет оценить производительность продукта или системы, например как быстро обрабатываются запросы пользователей.

Подхарактеристики:

* **поведение в зависимости от времени** — степень, в которой время отклика и обработки, а также пропускная способность продукта или системы при выполнении своих функций соответствуют требованиям (время обработки запроса пользователя);
* **использование ресурсов** — степень соответствия количества и типов ресурсов, используемых продуктом или системой при выполнении своих функций (потребляемый системой объём памяти);
* **вместимость** — степень, определяющая предельные показатели системы, которые должны быть описаны в требованиях (количество пользователей системы в единицу времени).

Пример: количество потребляемых серверных ресурсов относительно производимых действий (приложение-калькулятор не должно занимать все ресурсы системы). Или количество времени, необходимое для обработки запроса (поиск авиабилетов при заданных параметрах не должен занимать много времени, иначе пользователь перейдёт на приложение конкурентов). Поддерживает обслуживание до 10 000 запросов в секунду, время отклика на запрос при максимальной загрузке не более трёх секунд, на обработку одного запроса не должно расходоваться больше 1 Mб оперативной памяти.

1. **Надёжность** — степень, в которой система, продукт или компонент выполняют указанные функции при определённых условиях в течение определённого периода времени или способность быстро восстанавливать работу при отказе оборудования.

Подхарактеристики:

* **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 работают 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.

Пример: стабильная работа системы во время пиковых нагрузок (распродажи «чёрной пятницы» для интернет-магазина), обработка DDos-атак, стабильная работа при высоких нагрузках (сайты социальных сетей), сохранение данных при сбоях и отказах.

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

Подхарактеристики:

* **узнаваемость** — степень, в которой пользователи могут распознать, соответствует ли продукт или система их потребностям;
* **обучаемость** — степень, в которой продукт или система могут использоваться кем-либо для достижения целей обучения использованию продукта или системы с эффективностью, свободой от риска и удовлетворением в указанном контексте использования;
* **работоспособность** — степень, в которой продукт или система имеют атрибуты, облегчающие управление и контроль;
* **защита пользователя от ошибок** — степень, в которой система защищает пользователей от ошибок;
* **эстетика пользовательского интерфейса** — степень, в которой интерфейс позволяет приятное взаимодействие для пользователя;
* **доступность** — степень, в которой продукт или система могут использоваться людьми с самым широким диапазоном характеристик и возможностей для достижения указанной цели в заданном контексте использования.

Пример: пользователь интуитивно понимает, как покупать в интернет-магазине. Ему понятно, как искать товары в каталоге, как оформлять покупку. Кроме того, приятно находиться на сайте, ничто не отвлекает, нет раздражающих картинок, цветов, надписей.

1. **Поддерживаемость** — степень адаптируемости приложения к изменениям.

Подхарактеристики:

* **модульность** — степень, в которой система состоит из отдельных компонентов, так что изменение одного оказывает минимальное влияние на другие;
* **повторное использование** — степень, в которой компонент системы может использоваться более чем в одной системе или при создании других компонентов;
* **пригодность к анализу** — степень эффективности и результативности, с которой можно оценить влияние на продукт или систему предполагаемого изменения одной или нескольких его частей, или 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 выполнены тесты.

Пример: при росте популярности приложения возможно увеличить его «пропускную способность» (социальная сеть, количество пользователей которой стремительно увеличивается, должна иметь возможности для расширения своих ресурсов). При появлении новых технологий их легко можно внедрить в приложение.

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

Подхарактеристики:

* **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 и лёгкость установки** — степень эффективности и продуктивности, с которой продукт или система могут быть успешно установлены и (или) удалены в определённой среде;
* **заменимость** — степень, в которой продукт может заменить другой указанный программный продукт для той же цели в той же среде.

Пример: перенос данных из одной базы данных в другую без ущерба для работы приложения и без потери данных. Установка приложения на разные серверы или операционные системы.

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

Подхарактеристики:

* **конфиденциальность** — степень, в которой продукт или система гарантируют, что данные получат только те, кому разрешён к ним доступ;
* **целостность** — степень, в которой система, продукт или компонент предотвращают несанкционированный доступ или модификацию компьютерных программ или данных;
* **невозможность отказаться** — степень, в которой действия или события могут быть доказаны, без возможности их скрыть или совершить подмену;
* **ответственность** — степень, в которой действия объекта могут быть однозначно прослежены и определены ответственные лица;
* **подлинность** — степень, в которой личность субъекта или ресурса может быть доказана как заявленная.

Пример: защищённость личных данных, которая должна быть гарантирована приложением.

1. **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.

Подхарактеристики:

* **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 системы, продукты или компоненты могут обмениваться информацией и использовать её.

Пример: взаимодействие интернет-магазина с внешними платёжными и прочими системами.

# Дефект, ошибка, сбой, отказ

Зная, что такое качество ПО и как можно его определить, нужно помнить о том, что его снижает, а именно о дефектах. Дефект — основное понятие, с которым тестировщик сталкивается ежедневно. На разных проектах могут по-разному называть дефект — баг, ошибка, проблема. Начнём с базового определения.

**Дефект** — отклонение фактического результата от ожиданий наблюдателя, сформированных на основе требований, спецификаций, иной документации или опыта и здравого смысла.

В данном случае ожидаемый результат — поведение системы, которое описано в требованиях либо определено каким-то другим образом перед началом разработки системы. Фактический результат — поведение системы, которое наблюдается в процессе тестирования или использования приложения.

Так как не все проекты могут иметь документацию (требования, спецификации, технические задания), то тестировщику приходится опираться на предыдущий опыт разработки подобных приложений, критерии качества, существующие на рынке, а также на логику и здравый смысл.

Помимо дефекта, нужно понимать такие определения, как ошибка, сбой, отказ.

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

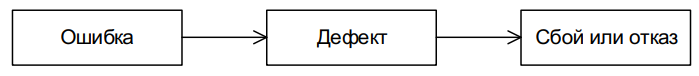
**Дефект** **(defect, bug, problem, fault)** — недостаток в компоненте или системе, способный привести к ситуации сбоя или отказа.

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

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

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

Таким образом, разработчик может совершить ошибку, написав неверный код. При выполнении функций приложения эта ошибка приводит к дефекту, который должен обнаружить тестировщик. Дефект может быть причиной сбоя или отказа.



Ещё одно понятие, связанное с ошибкой, — инцидент. Его также можно встретить в процессе разработки и эксплуатации приложения. Чаще оно используется для сбора статистики, чтобы отличать дефекты, которые были обнаружены тестировщиками, от дефектов, обнаруженных пользователями приложения.

**Инцидент** **(incident, deviation)** — любое сообщение о сбое в системе или дефекте графического интерфейса, поступившее от пользователя системы через службу технической поддержки.

# Отчёт о дефекте

Любой дефект, обнаруженный в приложении или системе, должен быть задокументирован и передан разработчику либо другому ответственному специалисту для ознакомления и исправления. Документ, с помощью которого о дефекте становится известно миру или отдельному специалисту, называется отчётом о дефекте. Есть несколько определений этого понятия.

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

**Отчёт о дефекте** — документ, описывающий и приоритизирующий обнаруженный дефект, а также содействующий его устранению.

Корректно составленный отчёт о дефекте очень важен в разработке приложения. Он позволяет не только зафиксировать обнаруженную неисправность в программе и информировать о ней, но и помогает её исправить. Чем грамотнее и точнее описан дефект, а также возможные причины его возникновения, тем проще и быстрее он будет исправлен. В этом состоит основной интерес тестировщика — хороший специалист, заинтересованный в качестве приложения, стремится не найти как можно больше дефектов, а способствует тому, чтобы как можно больше дефектов было исправлено в максимально короткие сроки.

Кроме того, отчёты о дефектах — инструменты для сбора статистики на проекте. Они помогают определить, в каких областях приложения и при каких условиях концентрируются дефекты.

Ещё одна важная функция отчётов о дефектах — приоритизация проблем, обнаруженных в приложении. При большом количестве найденных дефектов и ограниченных временных резервах разработчику важно понимать, какие дефекты хуже всего влияют на работу приложения и должны быть исправлены в первую очередь.

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.

Как у любого официального документа, у отчётов о дефектах есть свои атрибуты, которые нужно заполнять для предоставления полной информации. Не на всех проектах эти атрибуты используются, некоторые из них опускаются или изменяются. Но важно понимать базовые атрибуты отчёта о дефекте и знать, как они правильно заполняются.

## **Атрибуты отчёта о дефекте**

1. **Уникальный идентификатор** (ID)—присваивается автоматически, может содержать в себе данные о требовании, на которое ссылается дефект.
2. **Тема** (краткое описание, Summary) — кратко сформулированная суть дефекта по правилу «Что? Где? Когда?»
3. **Подробное описание** (Description)—более широкое описание сути дефекта (при необходимости).
4. **Шаги для воспроизведения** (Steps To Reproduce) —последовательное описание действий, которые привели к выявлению дефекта (которые нужно выполнить для воспроизведения дефекта). Описываются максимально подробно, с указанием конкретных вводимых значений.
5. **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?
6. **Ожидаемый результат** (Expected result) —указывается, как именно должна работать система по мнению тестировщика, основанному на требованиях и прочей проектной документации.
7. **Вложения** (Attachments)—скриншоты, видео или лог-файлы.
8. **Серьёзность дефекта** (важность, Severity)—характеризует влияние дефекта на работоспособность приложения.
9. **Приоритет дефекта** (срочность, Priority)—указывает на очерёдность выполнения задачи или устранения дефекта. Чем выше приоритет, тем быстрее нужно исправить дефект.
10. **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.

Дополнительно могут встречаться такие атрибуты, как:

* **компонент или среда** — указывает, на какой платформе этот дефект воспроизводится (iOS, Android, Windows, Mac);
* **версия** — указывает, на каком этапе разработки программного продукта был обнаружен дефект;
* **назначение** —указывается личность, на которую данный баг-репорт назначается для дальнейшей проверки или исправления;
* **номер сборки** —указывается номер билда, в котором был обнаружен дефект.

При создании отчёта о дефекте важно написать название, которое коротко и ёмко отражает суть проблемы. Хорошо написанное название позволяет разработчику быстрее сориентироваться и устранить дефект.

Что важно при написании названия отчёта о дефекте:

1. Название должно содержать ответы на следующие вопросы:

* **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 наступлении какого события или при каких условиях проблема проявляется.

1. Содержать предельно краткую, но достаточную для понимания сути проблемы информацию о дефекте.
2. Должно быть достаточно коротким, чтобы полностью помещаться на экране, потому что в системах отслеживания ошибок (bugtracker) конец предложения в поле краткого описания (summary) обрезается или приводит к появлению скроллинга.
3. Должно быть законченным предложением русского или английского (или иного) языка, построенным по соответствующим правилам грамматики.

Например:

* приложение зависает на экране загрузки файла после сохранения текстового файла размером больше 50 Мб;
* данные не сохраняются на форме «Профайл» после нажатия кнопки «Сохранить»;
* поле «Подтвердите пароль» не является обязательным на форме регистрации;
* текст выпадающего списка отображается за пределами выделенной области на главной странице после наведения курсора на меню вкладки «Курсы».

**Шаги по воспроизведению**такжеявляются ценной информацией в отчёте, поскольку представляют собой руководство к действию для тех, кто будет решать проблему. Шаги должны удовлетворять следующим требованиям:

1. На первом шаге необходимо использовать ссылку на главный домен:

* *открыть сайт http://....(Open the site http://...., Go to the site http://...);*

1. В шагах нужно отвечать на вопрос «Что необходимо сделать?»:

* *нажать кнопку «Найти» (Click the «Найти» button);*
* *ввести валидный email и пароль (Enter the valid email and the password);*
* *заполнить необходимые поля валидными данными (Fill the necessary fields with valid data).*

1. В шагах необходимо коротко писать, что сделать, куда нажимать, не уточняя названия страницы.
2. Описывается как минимум два шага, но не больше семи-восьми шагов.
3. Если шагов для воспроизведения будет очень много (больше восьми), можно указать предусловия (Preconditions), например:*Пользователь авторизован в системе. Товар был добавлен в корзину*.
4. В шагах нужно уточнять, на что именно необходимо нажимать (на ссылку, кнопку, логотип).
5. Шаги стоит писать с заглавной буквы.
6. В последнем шаге нужно описать, на какую область посмотреть, на что обратить внимание или что осмотреть:

* *осмотреть текст в выпадающем списке (подменю) (Look at the drop-down list (the submenu);*
* *обратить внимание на «Profile» форму (Take a look at the «Profile» form, Pay attention to the «Profile» form).*

Фактический и ожидаемый результаты рекомендуется описывать следующим образом:

1. Результаты необходимо описывать информативно (так же, как и тему, по принципу «Что? Где? Когда?», иногда можно использовать «Что?», «Где?»).
2. Сначала описывается фактический результат, затем ожидаемый.
3. Один результат в одном дефекте.
4. Результаты следует описывать полными предложениями, с подлежащим и сказуемым.
5. Результаты необходимо писать с заглавной буквы.

Например:

**Фактический результат.** Элементы подменю «Курсы» отображаются за границами выпадающего списка в главном меню после наведения курсора на блок «Курсы».

**Ожидаемый результат.** Элементы подменю «Курсы» находятся в пределах выпадающего списка в главном меню после наведения курсора на блок «Курсы».

**Серьёзность (severity)** показывает степень ущерба, который наносится проекту существованием дефекта.

**Срочность (priority)** показывает, как быстро дефект должен быть устранён.

На разных проектах тестировщики могут заполнять только «Серьёзность», а на некоторых и «Срочность». Часто на проектах встречается распределение ответственности при определении срочности и серьёзности дефекта. «Серьёзность» (severity) заполняет тестировщик, а «Срочность» (priority)— руководитель тестирования или проекта, исходя из того, насколько сильно дефект влияет на продукт, какие приоритеты у разработки либо на какой стадии находится разработка проекта.

Выделяют следующие **градации серьёзности**, которые могут различаться в зависимости от проекта:

* Block (блокирующий дефект) — блокирует большую часть работы в проекте;
* Crash (авария) — приводит к падению приложения (или даже операционной системы);
* Major (значительный дефект) — дефект в важном, но не ключевом функционале;
* Minor (незначительный дефект) — дефект в дополнительном функционале;
* Tweak (неудобство) — неудобство в использовании, требуется небольшая доработка для повышения удобства использования;
* Text (текст/опечатка) — небольшая текстовая ошибка, опечатка;
* Trivial (тривиальный дефект) — незначительная недоработка.

Срочность может быть высокой (High), средней (Medium), низкой (Low). Порядок исправления ошибок по их приоритетам: High → Medium → Low.

Статус дефекта зависит от практики, сложившейся на проекте, и используемой системы отслеживания дефектов. Ориентировочно можно выделить следующие:

* Новый (New) — новый отчёт о дефекте;
* Открыт (Opened) — отчёт о дефекте открыт;
* Назначен (Assigned) — отчёт о дефекте 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, отчёт о дефекте закрыт.

При создании отчёта о дефектах также важно помнить базовые правила:

1. Старайтесь следовать правилу «Что? Где? Когда?»
2. Одна ошибка — один отчёт о дефекте.
3. Старайтесь быть лаконичными при описании дефекта.
4. Каждый дефект должен быть воспроизведён повторно перед написанием отчёта.
5. Отчёт о дефекте должен быть составлен сразу, не откладывайте на потом.
6. Минимизируйте количество шагов в описании.
7. Пишите техническим языком с применением терминологии, принятой на проекте.
8. Прикрепляйте дополнительные файлы (логи, скриншоты, видео).
9. Старайтесь часть логов прикреплять в описании. Это поможет при поиске дефектов по сообщениям об ошибках, которое выдаёт приложение.
10. Используя видео, вы можете короче описать дефект и избежать недопонимания.
11. Прикрепляйте ссылки к требованиям, это поможет избежать споров и сэкономить время.
12. Избегайте дубликатов дефектов — прежде чем составлять отчёт о дефекте, спросите у коллег или проверьте в баг-трекере, есть ли уже такой дефект.
13. Указывайте версию ПО и тестовый стенд (окружение), на котором был обнаружен дефект.
14. Попробуйте воспроизвести дефект, следуя вашим собственным шагам.

# Рабочий день тестировщика

Рабочий день у каждого тестировщика свой. Но есть общие черты, которые можно выделить.

Обычно рабочий день начинается с проверки почты и получения информации о новых версиях продукта (билдах, сборках).

Затем следует ежедневный митинг, если методология разработки проекта предполагает проведение митингов, совещаний, стендапов. Подробнее о моделях и методологиях разработки вы узнаете на следующих занятиях. На митинге обсуждаются текущие цели проекта, какие задачи выполняет каждый член команды, какие существуют проблемы на текущем этапе разработки.

После митинга начинается рабочий процесс. Он может состоять из настройки тестового окружения (места, где будет проводиться проверка тестовой версии), тестирования новой функциональности, написания тест-кейсов (сценариев для проверки разных функций приложения). Часто тестировщикам приходится общаться с другими членами команды: разработчиками, аналитиками, менеджерами и уточнять у них требования к работе приложения. Иногда могут возникать дискуссии с разработчиками по поводу какой-либо функциональности.

Проверка приложения, как правило, приводит к обнаружению дефектов, которые должны быть занесены в специальную систему для их учёта. Поэтому одна из важных задач тестировщика — писать отчёты о дефектах и присваивать им уровень важности.

Разработчики постепенно исправляют дефекты, и тогда задачей тестировщика становится проверить, действительно ли дефект исправлен и больше не воспроизводится (это называется ретестом).

Помимо тестирования новых функций, которые внедрили разработчики в приложение, нужно проводить проверку функционала, который уже был протестирован ранее и стабильно работал до внесения изменений в код. Это регрессионное тестирование, оно может занимать достаточно много рабочего времени.

Также тестировщики занимаются тестированием требований к системе. Это важный вид тестирования, способный предотвратить некоторые дефекты ещё до начала процесса разработки.

Иногда нужно писать отчёты о проделанной работе, проводить анализ найденных и исправленных дефектов, заносить в специальную систему данные о времени, которое было потрачено на решение задач в течение дня.

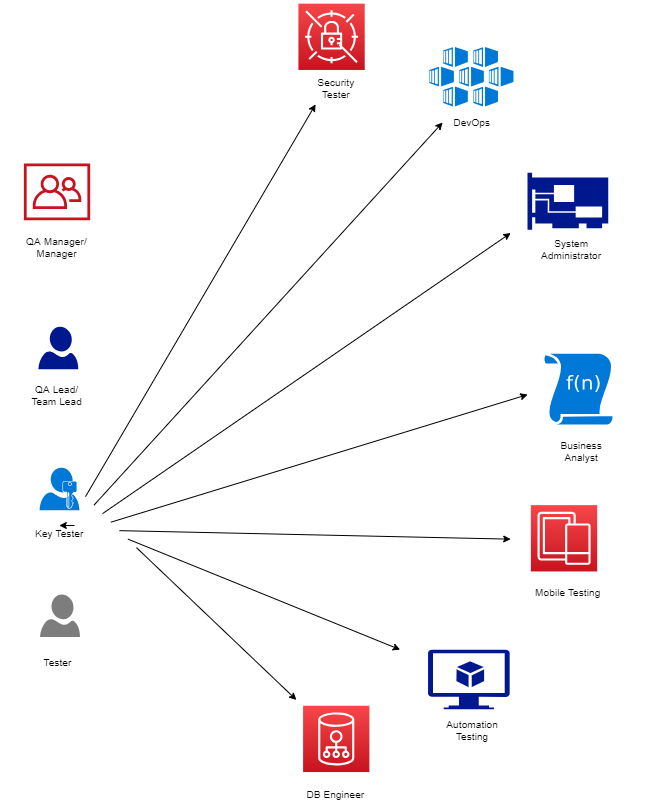
Очень много времени в работе тестировщика могут занимать различные споры, обсуждения, уточнения, что делает работу более живой и динамичной.

Тестировщику важно быть внимательным и скрупулёзным, всё ставить под сомнение и задавать много вопросов. Тестирование — в какой-то степени расследование, в котором тестировщик, как истинный детектив, должен задавать много разных вопросов, чтобы вычислить дефект, который старательно замаскировался под функциональность. А потом ещё и доказать разработчику, что это дефект, используя убедительные аргументы и доказательства.

# Пути развития тестировщика

Рассмотрим, какие существуют пути развития тестировщика.

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

Вы начнёте с должности Junior-специалиста, далее идёт Middle-специалист, затем Senior/старший специалист, иногда встречается ещё такое понятие, как Key Tester, то есть ключевой тестировщик на проекте. Это, как правило, опытный специалист, работающий достаточно давно на проекте и знающий его очень хорошо. Руководитель команд — QA Team Lead. Ещё выше — QA Manager проекта. Не путайте его с Product Manager и с руководителем отдела тестирования или QA department.

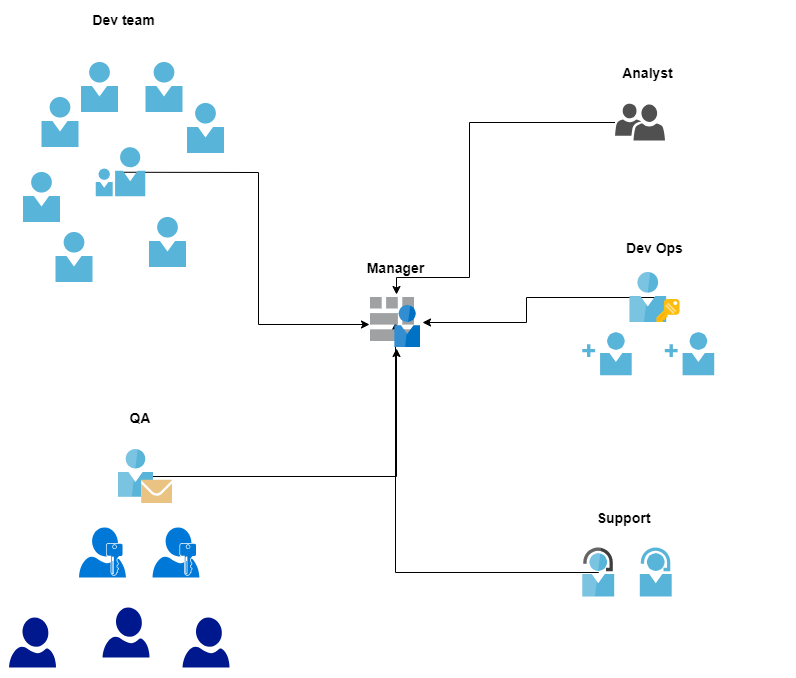
Специалист по тестированию может развиваться не только строго вверх, но и в какой-то узкой специализации. Это могут быть такие направления, как автоматизация тестирования, нагрузочное тестирование, тестирование безопасности, базы данных, переход в отдел аналитики.

# Команда IT-проекта

Обычно в команду разработки входят:

* разработчики;
* аналитики;
* тестировщики;
* менеджер проекта;
* системные администраторы.

Также следует выделить DevOps (инженер развёртывания) и службу поддержки.



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

На основании требований ставятся задачи команде разработки. Разработчики, основываясь на требованиях, создают нужную функциональность.

Когда задача реализована, а разработчики сообщили, что новая фича (функциональность) внедрена и выпущен тестовый билд, то есть тестовая сборка, в которую входят реализованные новые функции, её тестируют. Тестировщики в работе основываются на требованиях. Если поведение программы отличается от ожидаемого, тестировщик составляет отчёт о дефекте.

Тестировщики должны разбираться в готовом продукте. Исключение могут составлять узконаправленные специалисты, например специалист по нагрузочному тестированию, автоматизации или тестированию безопасности. Они часто 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 лучше договориться с заказчиком об отсрочке и закончить работу.

Если продукт уже выпущен, а по согласованию с заказчиком компания должна оказывать поддержку, то существует и служба поддержки, предоставляющая консультационную помощь при возникновении проблем с продуктом. Поэтому они должны хорошо разбираться, как работает продукт.

# Дополнительные материалы

**Составление и оформление отчётов о дефектах**

[Почему грамотное оформление баг-репортов так важно](https://training.qatestlab.com/blog/technical-articles/proper-bug-report/)  
[Как правильно необходимо описывать шаги в баг-репорте](https://training.qatestlab.com/blog/technical-articles/steps-bug-report/)  
[Основные атрибуты баг-репорта](https://training.qatestlab.com/blog/technical-articles/attributes-bug-report/)  
[Баг-репорт: об использовании «некорректных» слов](https://training.qatestlab.com/blog/technical-articles/wrong-words-report/)  
[Материалы лекции №1 «Вводная часть. Что такое баг (дефект)» по курсу «Основы тестирования ПО»](https://training.qatestlab.com/blog/course-materials/lecture-testing-introduction/)  
[Правила описания результатов в баг-репорте](https://training.qatestlab.com/blog/technical-articles/results-bug-report/)  
[Правила описания багов на английском языке. Использование пассива](https://training.qatestlab.com/blog/technical-articles/passive-voice-bugs/)  
[С чего начинать описание шагов воспроизведения дефекта (бага)?](https://training.qatestlab.com/blog/technical-articles/steps-bug-reproduction/)  
[Как правильно составить описание бага в баг-трекере. Принцип «Что? Где? Когда?»](https://training.qatestlab.com/blog/technical-articles/description-bug-tracker/)

Книга Software Testing — Base Course (Svyatoslav Kulikov), с. 164–204.

**Рассказ о себе**

[Tell me about yourself job interview question](https://www.youtube.com/watch?v=fTn2d8uh34Y)

**Общее о тестировании**

[Материалы ISTQB](https://www.rstqb.org/ru/istqb-downloads.html?file=files/content/rstqb/downloads/ISTQB%20Downloads/IS)  
[Основные положения тестирования](https://testitquickly.com/2010/03/09/testing-basics-by-barancev/)

[Образ современного тестировщика. Что нужно знать и уметь](https://habr.com/ru/company/funcorp/blog/426759/)

# Используемые источники

[Общие вопросы](https://www.rstqb.org/ru/faq/obschie-voprosy.html)

[ISTQB Glossary](https://glossary.istqb.org/en/search)

[Wikipedia, ISO/IEC 9126, Developments](https://en.wikipedia.org/wiki/ISO/IEC_9126#Developments)

[ISO 25010](https://iso25000.com/index.php/en/iso-25000-standards/iso-25010)