

Введение в тестирование  
Урок 1

# Введение в тестирование ПО





# Оглавление

На этом уроке	4
Введение	<b>Error! Bookmark not defined.</b>
Что такое тестирование?	4
Что такое качество?	6
Критерии качества	7
Функциональность	7
Эффективность	7
Надёжность	8
Удобство использования	8
Поддерживаемость	9
Переносимость	10
Безопасность	10
Совместимость	11
Цели тестирования и роль тестировщика	11
Зачем тестировать ПО?	11
Цели тестирования ПО	12
Обязанности инженера QA	12
Принципы тестирования	133
1. Тестирование демонстрирует наличие дефектов	133
2. Исчерпывающее тестирование недостижимо	13
3. Раннее тестирование	13
4. Скопление дефектов	13
5. Парадокс пестицида	134
6. Тестирование зависит от контекста	144
7. Заблуждение об отсутствии ошибок	144



## Урок 1. Введение в тестирование ПО

Тестовая среда и тестовые данные	14
Testing, QC, QA	156
Hard Skills & Soft Skills	16
Карьера и рост	17
Контрольные вопросы	<b>Error! Bookmark not defined.</b>
Дополнительные материалы	<b>Error! Bookmark not defined.8</b>



## На этом уроке

1. Познакомимся с курсом.
2. Узнаем, что такое тестирование и категории качества тестирования.
3. Познакомимся с целями тестирования и ролью тестировщика.
4. Перечислим принципы тестирования.
5. Познакомимся с тестовой средой и тестовыми данными.
6. Определим понятия Testing, QA и QC.
7. Обсудим hard и soft skills тестировщика и его карьерные возможности.

## Введение

Приветствуем вас на вводном курсе по тестированию программного обеспечения. Наша цель — больше узнать о тестировании, понять, зачем оно нужно на проекте, и разобраться, как тестируется продукт.

Сегодня мы узнаем, что такое тестирование, каковы его цели и принципы. Поймём, в чём разница между тестированием, QC и QA. Разберёмся, какими навыками должен обладать тестировщик, как можно развиваться в сфере и профессии.

Тестирование — не точная наука, как физика или математика. У одного и того же понятия может быть несколько определений, и каждое будет верным. Академические, чётко сформулированные определения можно посмотреть в глоссарии ISTQB.

ISTQB (International Software Testing Qualifications Board) — некоммерческая организация, которая занимается вопросами развития сферы тестирования ПО. Основана представителями 8 стран: Австрии, Дании, Финляндии, Германии, Швеции, Швейцарии, Нидерландов и Великобритании. ISTQB — международный стандарт в области тестирования ПО.

## Что такое тестирование?

В широком смысле, **тестирование** — это проверка чего угодно с помощью тестов, то есть заранее подготовленного списка вопросов, проверок, испытаний. Преподаватели тестируют знания студентов. Производители автомобилей проводят краш-тесты, то есть тестируют их безопасность. Фармацевтические компании тестируют лекарства на добровольцах, определяя их безопасность и действенность.



Программное обеспечение (ПО) нуждается в тестировании так же, как автомобили и лекарства. Прежде чем отдать пользователю программу, поставщик должен убедиться, что ПО:

- безопасно,
- надёжно,
- понятно,
- доступно,
- обладает прочими показателями, совокупность которых определяет качество программы.

**Тестирование программного обеспечения** — это проверка соответствия реального и ожидаемого поведения программы, а также степени удовлетворения потребностей пользователя. Выполняется на конечном наборе тестов, который составляет тестировщик.

Тестирование — одна из техник контроля качества. Включает в себя:

- планирование работ (Test Management);
- проектирование тестов (Test Design);
- выполнение тестирования (Test Execution);
- анализ полученных результатов (Test Analysis).

**Общие признаки тестирования:**

- Продукт, показатели которого неизвестны. Например, неизвестно, безопасен ли автомобиль.
- Заранее подготовленный список проверок — стандартизированные краш-тесты.
- Правила проведения исследования — описание процессов, протоколы.
- Ожидаемый результат тестирования — если автомобиль сталкивается с препятствием, срабатывают подушки безопасности.
- Фактический результат тестирования — подушка сработала или не сработала.

ISTQB определяет тестирование как процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические. Эти активности включают планирование, подготовку и оценку программного продукта и связанных с этим результаты работ.

На первый взгляд, определение сложное. Разберём его по частям.

1. Тестирование — это поэтапный процесс. Включает:



- a. планирование тестирования;
  - b. анализ требований;
  - c. проектирование и реализация тестов;
  - d. создание отчётов о ходе и результатах тестирования;
  - e. оценка качества объекта тестирования.
2. Совокупность этапов — это жизненный цикл тестирования.
3. В тестировании выделяются динамические и статические активности:
  - динамические предполагают исполнение готового кода — выполнение тестов на работающей версии программного продукта (это может быть запущенное приложение или открытый сайт);
  - статические не требуют запуска кода на исполнение, это анализ макетов, рецензирование документации и кода.
4. Цели тестирования:
  - a. определить, насколько ПО соответствует заявленным требованиям;
  - b. выявить дефекты, то есть отклонения ПО от требований.

Таким образом, тестирование отвечает на вопрос: **каков уровень качества ПО?**

## Что такое качество?

**Качество программного обеспечения** — это совокупность характеристик, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Качество программного продукта характеризует набор свойств, определяющих, насколько продукт «хорош» с точки зрения заинтересованных сторон: заказчика, спонсора, пользователя, разработчиков, тестировщиков, инженеров поддержки, сотрудников отдела маркетинга, обучения и продаж. У каждого участника может быть своё представление о качестве продукта.

**Качество** — это степень того, насколько компонент, система или процесс соответствует требованиям и ожиданиям пользователя.



## Критерии качества

Выделяют восемь критериев качества: функциональность, эффективность, надёжность, удобство использования, поддерживаемость, переносимость, безопасность, совместимость. Рассмотрим их на примере сигнализации автомобиля. Может показаться, что пример нереальный, но это не так. Сигнализация — сложное устройство, которое включает в себя сам блок (мозги), мобильное приложение и серверную часть (например, если автомобиль заводится удалённо).

### Функциональность

Продукт или система должны выполнять основные функции — то есть решать те задачи, для которых они созданы. Например, сигнализация с автозапуском создана, чтобы сообщить владельцу, что его автомобиль ударили или пытаются угнать, а также для удалённого запуска.

Критерии:

- **Функциональная полнота** — насколько функциональность охватывает и решает задачи пользователя. Например, сигнализация включается во время удара, автозапуск срабатывает.
- **Функциональная правильность** — насколько продукт или система обеспечивают правильные результаты с требуемой степенью точности. Например, через какое время после удара сработает сигнализация.
- **Функциональная целесообразность** — насколько функции продукта востребованы пользователем. Иногда дополнительные ненужные функции навязываются пользователю.

### Эффективность

Производительность продукта или системы. Например, как быстро обрабатываются запросы пользователей.

Критерии:

- **Поведение в зависимости от времени** — насколько быстро и оптимально обрабатывается запрос пользователя. Например, как быстро заводится автомобиль после нажатия кнопки «завести».



- **Использование ресурсов** — насколько оптимально ПО расходует ресурсы. Например, к утру у автомобиля не разряжается аккумулятор.
- **Вместимость** — каковы предельные показатели системы, количество пользователей в единицу времени. Что будет, если все пользователи сервиса решили завести автомобили разом в 8 утра?

## Надёжность

Степень выполнения указанных функций системой, продуктом или компонентом при определённых условиях в течение конкретного периода времени. А также способность быстро восстанавливать работу при отказе оборудования. Например, когда сел аккумулятор, его заменили, и телеметрическое оборудование перезапустилось.

Критерии:

- **Завершённость** — степень удовлетворения потребности при нормальной работе системы, продукта или компонента. Например, насколько хорошо сигнализация работает при нормальном использовании.
- **Доступность** — степень работоспособности и доступности системы, продукта или компонента при использовании.
- **Отказоустойчивость** — насколько стабильно работает ПО при аппаратных или программных сбоях. Например, при отключении электричества или интернета.
- **Восстанавливаемость** — насколько продукт или система могут восстановить данные и желаемое состояние системы при прерывании или сбое.

## Удобство использования

Насколько эффективно продукт удовлетворяет потребности конкретного пользователя. Например, пользователю может быть неудобно работать в приложении — непонятно, куда нажимать.

Критерии:

- **Узнаваемость** — насколько пользователи распознают, что умеет делать продукт.
- **Обучаемость** — насколько продукт помогает пользователю научиться эффективно в нём работать. Есть ли в нём подсказки, «обучалки» при первом запуске, неизменный основной интерфейс.





- **Работоспособность** — степень, в которой продукт или система имеют атрибуты, облегчающие управление и контроль.
- **Защита от ошибок** — насколько система защищает пользователей от ошибок: предотвращает деструктивные действия, показывает сообщения при сбоях или при вводе некорректной информации. Возможно, вам знакома ситуация: заполнили много полей, свернули приложение, а вернувшись, обнаружили, что всё пропало.
- **Эстетика пользовательского интерфейса** — степень «приятности» взаимодействия пользователя и интерфейса.
- **Доступность** — насколько широкий круг людей может пользоваться продуктом. Зависит от совместимости с программным или аппаратным обеспечением. Например, если приложение работает только на iOS и не работает на Android, оно не доступно множеству пользователей.

## Поддерживаемость

Степень адаптируемости приложения к изменениям. Например, если пользователь подключит дополнительные датчики, увидит ли их система?

Критерии:

- **Модульность** — степень, в которой система состоит из отдельных компонентов. Изменение одного оказывает минимальное влияние на другие.
- **Повторное использование** — степень, в которой компонент системы используется более чем в одной системе или при создании других компонентов.
- **Пригодность к анализу** — степень эффективности и результативности, с которой оценивается влияние на продукт или систему предполагаемого изменения одной или нескольких его частей. Продукт диагностируется на наличие недостатков или причин отказов. Определяются детали, подлежащие модификации.
- **Модифицируемость** — степень, в которой продукт или система могут эффективно модифицироваться без внесения дефектов или ухудшения качества продукта.
- **Тестируемость** — степень эффективности и результативности, с которой устанавливаются критерии тестирования для системы, продукта или компонента, а также выполняются тесты.



## Переносимость

Степень эффективности и результативности, с которой система, продукт или компонент переносятся из одного аппаратного, программного, операционного или эксплуатационного окружения в другое. Например, если пользователь захотел перенести сигнализацию на новый автомобиль.

Критерии:

- **Адаптивность** — степень, в которой продукт или система эффективно адаптируются к разным или развивающимся аппаратным средствам, программному обеспечению или другим операционным средам, а также к средам использования.
- **Простота и лёгкость установки** — степень эффективности и продуктивности, с которой продукт или система успешно устанавливаются и (или) удаляются в конкретной среде.
- **Заменяемость** — степень, в которой продукт может заменить другой указанный программный продукт для идентичной цели и среды.

## Безопасность

Степень, в которой продукт или система защищают информацию и данные таким образом, чтобы лица или другие системы имели степень доступа к ним, соответствующую их типам и уровням авторизации. Например, чтобы злоумышленник не смог перехватить запрос от вашего приложения и не получил управление автомобилем.

Критерии:

- **Конфиденциальность** — степень, в которой продукт или система гарантируют, что данные получают только те, кому разрешён доступ к ним.
- **Целостность** — степень, в которой система, продукт или компонент предотвращают несанкционированный доступ или модификацию компьютерных программ (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 однозначно прослеживаются, а также определяются ответственные лица.



- **Подлинность** — степень, в которой личность субъекта или ресурса доказывается как заявленная.

## Совместимость

Степень, в которой продукт, система или компонент обмениваются информацией с другими продуктами, системами или компонентами и выполняют требуемые функции, совместно используя ту же аппаратную или программную среду.

- **Сосуществование** — степень, в которой продукт эффективно выполняет требуемые функции при совместном использовании общей среды и ресурсов с другими продуктами без вредного воздействия на любой другой продукт. Например, сигнализация использует экран бортового компьютера, чтобы отображать датчики парктроника.
- **Совместимость** — степень, в которой системы, продукты или компоненты обмениваются информацией и используют её.

## Цели тестирования и роль тестировщика

### Зачем тестировать ПО?

У разработчиков и менеджеров продукта часто возникает вопрос: «Для чего нужно тестирование на проекте? Мы можем проверить всё сами!». Есть несколько причин:

1. Процесс тестирования гарантирует, что ПО будет работать в соответствии с ожиданиями клиентов на имеющемся у них оборудовании.
2. Выявление проблем на начальном этапе разработки уменьшает объём работ по исправлению ошибок на более поздних стадиях, обеспечивает правильное использование ресурсов и предотвращает повышение стоимости.
3. Команда тестирования привносит взгляд клиента в процесс и находит варианты использования, о которых разработчик может не подумать.

Любой сбой, дефект или ошибка, обнаруженные клиентом в готовом продукте, рушат доверие к компании и увеличивают финансовые затраты.



## Цели тестирования ПО

Цели тестирования — это те задачи, которыми вы будете заниматься практически на любом проекте:

1. Оценка соответствия стандартам компании рабочих продуктов: требований, пользовательских историй, проектирования и кода.
2. Проверка выполнения требований.
3. Поиск и предотвращение дефектов.

Такая постановка целей даёт возможность:

1. Повысить вероятность, что приложение, предназначенное для тестирования, будет работать правильно.
2. Повысить вероятность, что приложение, предназначенное для тестирования, будет соответствовать всем описанным требованиям.
3. Предоставить актуальную информацию о состоянии продукта в конкретный момент.

## Обязанности инженера QA

1. Изучение и уточнение требований к ПО у заказчика (в больших проектах этим могут заниматься бизнес-аналитики).
2. Написание и доработка сценариев тестирования.
3. Тестирование функционала ПО.
4. Составление отчётов по обнаруженным недочётам в трекинговой системе — программе, в которую разработчики, программисты и тестировщики могут вносить найденные ошибки и отслеживать их выполнение / невыполнение.
5. Анализ результатов и показателей проведённых тестов.
6. Составление ТЗ на устранение найденных недочётов.
7. Мониторинг и отслеживание правок.
8. Проведение повторных тестов, чтобы убедиться, что найденные ошибки исправлены.
9. Анализ и оптимизация этапов разработки для устранения причин ошибок и защиты от их повторного появления.
10. Работа с тестовой документацией.



# Принципы тестирования

Как определить, что во время тестирования вы придерживаетесь правильной стратегии? Нужно следовать семи принципам тестирования, которые определяет стандарт ISTQB. Они оптимизируют количество и качество работ в QA.

## 1. Тестирование демонстрирует наличие дефектов

Тестирование показывает, что дефекты есть, но не может доказать, что их нет.

Тестирование снижает вероятность, что в ПО есть дефекты, но, даже если дефекты не обнаружены, не доказывает его полную корректность.

## 2. Исчерпывающее тестирование недостижимо

Полное тестирование с использованием всех комбинаций вводов и предусловий физически невыполнимо, кроме тривиальных случаев. Вместо исчерпывающего тестирования используется анализ рисков и расстановка приоритетов, чтобы точнее сфокусировать усилия по тестированию.

## 3. Раннее тестирование

Тестирование должно начинаться как можно раньше в жизненном цикле разработки программного обеспечения, и его усилия должны концентрироваться на конкретных целях. Такой подход поможет раньше найти дефекты.

## 4. Скопление дефектов

Как правило, большинство дефектов обнаруживают в небольшом количестве модулей. Усилия по тестированию сосредотачивают пропорционально ожидаемой, а затем и реальной плотности дефектов в модулях.

## 5. Парадокс пестицида

Если одни и те же тесты выполняются много раз, в конечном счёте этот набор тестовых сценариев больше не будет находить новые дефекты.

Чтобы преодолеть парадокс пестицида, тестовые сценарии нужно регулярно рецензировать и корректировать. При этом новые тесты должны быть



разносторонними, чтобы охватить все компоненты ПО или системы, и найти как можно больше дефектов.

## 6. Тестирование зависит от контекста

Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, где критически важна безопасность, тестируется иначе, чем сайт электронной коммерции.

## 7. Заблуждение об отсутствии ошибок

Обнаружение и исправление дефектов не помогут, если созданная система не подходит пользователю и не удовлетворяет его ожиданиям и потребностям.

## Тестовая среда и тестовые данные

**Тестовая среда** — по сути, полигон для испытаний: например, сервер, мобильный телефон и так далее. Настраивается в соответствии с требованиями тестируемого приложения.

Для чего это нужно? Всё, чем пользуются люди, проходит испытания. Если сделать прибор и сразу продавать его, он может оказаться небезопасным. Поэтому его нужно проверить на полигоне с условиями, максимально приближенными к реальным.

Так и в разработке ПО: если код сразу выкатить в продакшен (применить в том окружении, где есть пользователи), есть высокая вероятность что-то сломать. Поэтому любая задача проходит тестирование в нескольких окружениях — копиях основного сайта или приложения.

**Окружение** (environment) — это среда, место или машина, на которой находится приложение / сайт.

На проекте можно встретить несколько видов таких сред:

- **Локальное окружение** — всего одна машина, на которой разрабатывают и тестируют приложение. Тестировщик или разработчик может развернуть её у себя на ноутбуке. Грубо говоря, это виртуалка с копией проекта. Часто у неё ограниченный функционал: например, нет реальных товаров или возможности их оплатить.



- **Тестовая** — среда тестирования функциональности. Она наполнена тестовыми данными, которые могут показывать случаи, редко возникающие в рабочей среде или удобные для тестирования. Здесь идёт тестирование того, что готовится в продакшен.
- **Stage или staging-окружение** — среда, в точности похожая на продакшен-окружение. Может подключаться к другим продакшен-сервисам и данным, таким как базы данных.
- **Боевое окружение** — реальная сеть машин, совокупность нескольких окружений. Не тестовое, а самое настоящее окружение, где работают пользователи. Тестирование в такой среде практически не проводится.

**Тестовые данные** — это набор входных значений, нужных для выполнения тестов. Тестировщики определяют данные в соответствии с требованиями. Для генерации тестовых данных используют различные инструменты.

Инструменты генерации — это программы, которые быстро создают тестовые данные. Их целесообразно использовать в случаях, когда:

- нужно много однотипных данных, например, паспортных;
- ручная генерация тестовых данных занимает много времени (в случае тестирования сложных систем);
- данные нужны для автотестов (код, который пишут тестировщики для покрытия ручных проверок).

Инструменты генерации:

- таблицы Excel;
- онлайн-утилиты, например, [Pairwise Pict Online](#) для генерации попарных данных;
- скрипты для выполнения в командной строке;
- сложные программы, созданные самими тестировщиками.

## Testing, QC, QA

QC и QA — похожие, но не взаимозаменяемые термины.

**QC** (Quality Control) — контроль качества: анализ результатов тестирования и качества новых версий выпускаемого продукта.

Задачи:



- проверка готовности ПО к релизу;
- проверка соответствия требований и качества проекта.

**QA** (Quality Assurance) — обеспечение качества продукта: изучение возможностей изменения и улучшения процесса разработки и коммуникаций в команде. Тестирование — один из аспектов обеспечения качества.

Задачи:

- проверка технических характеристик и требований к ПО;
- оценка рисков;
- планирование задач для улучшения качества продукции;
- подготовка документации, тестового окружения и данных;
- тестирование;
- анализ результатов тестирования, а также составление отчётов и других документов.

**Тестирование** — процесс проверки результатов работы на соответствие установленным требованиям. Тестировщик — специалист, который занимается такой проверкой: тестирует компоненты продукта или весь продукт целиком на предмет ошибок или неточностей разработки.

Тестирование — один из ключевых процессов в системе обеспечения качества.

## Hard Skills & Soft Skills

**Hard skills** (с англ. «жёсткие навыки») — технические компетенции, которые можно наглядно продемонстрировать, оценить и проверить. К hard skills относятся навыки работы с инструментами, знание языков программирования, SQL, системы контроля версий Git, основ тестирования, техники тест-дизайна, а также понимание жизненного цикла тестирования, компьютерная грамотность и многое другое.

**Soft skills** (с англ. «мягкие навыки») — универсальные социально-психологические качества, которые не зависят от профессии, но непосредственно влияют на успешность человека. К soft skills относятся коммуникативные навыки, организованность, способность решать конфликты, умение убеждать, работать в команде, адаптивность, обучаемость. Они могут зависеть от характера человека и формироваться с опытом.





В последние годы на soft skills обращают всё больше внимания. Иногда они оказываются решающими для оффера — проще подтянуть хардовую часть, чем изменить характер человека.

## Карьера и рост

В тестировании есть несколько грейдов: Junior QA (стартовая позиция), Middle QA, Senior QA (опытный специалист, хорошо знакомый с проектом).

Вы можете прокачивать знания в области автоматизации тестирования, в нагрузочном тестировании, в тестировании баз данных и в тестировании безопасности. Подробнее о компетенциях специалиста — в [roadmap](#).

Можно прокачивать не только профессиональные, но и менеджерские скилы, чтобы расти в сторону QA Team Lead.

Чтобы определиться с оптимальным направлением для развития, важно понять, какой тип навыков у вас развит лучше — hard skills или soft skills.

Если мы говорим о заработной плате, то в Москве стартовая зарплата тестировщика с уверенным знанием теории и опытом практической работы — 40–55 тыс. рублей (может быть выше). В регионах оклад ниже. Специалист с опытом от 1 года может рассчитывать на зарплату в 80–150 тыс. рублей.

Познакомиться с результатами опроса, где специалисты указали свой первоначальный доход, можно по [ссылке](#).

## Контрольные вопросы

1. Что такое тестирование?
2. Что такое качество?
3. Какие цели у тестирования?
4. Какие критерии качества есть?
5. Какие принципы тестирования есть?
6. Что такое тестовая среда?
7. Что такое тестовые данные?
8. Чем отличаются между собой тестирование, QC и QA?

## Дополнительные материалы



1. [Фундаментальная теория тестирования](#)
2. [Профессия тестировщик: разбираемся в QA, QC и testing](#)
3. [Семь главных принципов тестирования](#)