

Основы ручного тестирования

# Виды тестирования



# На этом уроке

1. Узнаем о видах тестирования.
2. Рассмотрим тест-кейсы для различных видов тестирования.

## Оглавление

### [Три ящика тестирования](#)

#### [Тестирование методом чёрного ящика](#)

##### [Преимущества чёрного ящика](#)

##### [Недостатки чёрного ящика](#)

#### [Тестирование белого ящика](#)

##### [Преимущества белого ящика](#)

##### [Недостатки белого ящика](#)

#### [Тестирование серого ящика](#)

### [Нефункциональное тестирование](#)

#### [Тестирование производительности](#)

##### [Нагрузочное тестирование](#)

##### [Тестирование масштабируемости](#)

##### [Объёмное тестирование](#)

##### [Стрессовое тестирование](#)

##### [Основные понятия в тестировании производительности](#)

#### [Прочие виды нефункционального тестирования](#)

### [Прочие разновидности тестирования](#)

#### [Виды тестирования, связанные с изменениями в коде](#)

##### [Статическое и динамическое тестирование](#)

##### [Позитивное и негативное тестирование](#)

##### [Альфа- и бета-тестирование](#)

##### [Тестирование на основе тест-кейсов и исследовательское тестирование](#)

### [Контрольные вопросы](#)

### [Практическое задание](#)

### [Глоссарий](#)

### [Дополнительные материалы](#)

# Три ящика тестирования

Работая с программой, тестировщик обычно «держит в уме» её архитектурные компоненты и особенности их взаимодействия.

Бывают ситуации, когда тестировщику ничего не известно об устройстве программы «под капотом». Или, наоборот: тестировщик видит код программы и пишет тесты, опираясь на него. В каждом случае тестирование имеет особенности, обусловленные уровнем знаний о внутреннем устройстве программы. В зависимости от этого уровня выделяются три вида тестирования: чёрного, белого и серого ящика.

## Тестирование методом чёрного ящика

Чёрный ящик — это система, внутреннее устройство и механизм работы которой сложны, неизвестны или не важны в рамках решения задачи.

Метод чёрного ящика — метод исследования систем, когда вместо свойств и взаимосвязей составных частей системы, изучается реакция системы, как целого, на изменяющиеся условия.

У чёрного ящика есть «вход» для ввода информации и «выход» для отображения результатов работы. При этом происходящие процессы во время работы системы наблюдателю неизвестны. Состояние выходов функционально зависит от состояния входов.

Тестирование чёрного ящика (black box testing) — тестирование, основанное на анализе функциональной или нефункциональной спецификации системы без знания внутренней структуры.

У тестировщика нет доступа к коду, он видит приложение как пользователь. Тестирование проводится через интерфейс приложения. Это ручное тестирование без знания, что находится «за кулисами» интерфейса.

Тест-дизайн, основанный на технике чёрного ящика, это написание или отбор тест-кейсов на основе анализа документации без знания внутреннего устройства программы.

Тестирование чёрного ящика находит ошибки:

1. Функции неправильно реализуются или отсутствуют.
2. Интерфейс отличается от макетов.
3. Данные не записываются в базы, или записываются неверно.
4. Недостаточная производительность системы.

Таким образом, тестировщик концентрируется на том, ЧТО программа делает, а не КАК.

Пример:

Тестировщик проводит тестирование веб-сайта, не зная особенностей его реализации. Он использует только предусмотренные разработчиком поля ввода и кнопки. Источник ожидаемого результата — спецификация.

### **Преимущества чёрного ящика**

1. Тестирование производится с позиции пользователя и обнаруживает неточности и противоречия в поведении ПО.
2. Тестировщику необязательно знать языки программирования.
3. Тестирование проводится независимыми специалистами, что помогает избежать предвзятого отношения.
4. Тест-кейсы пишутся, как только готова спецификация.

### **Недостатки чёрного ящика**

1. Тестируется ограниченное количество сценариев.
2. Без чёткой спецификации трудно составить эффективные тест-кейсы.
3. Тесты избыточны, если их уже проверил разработчик на уровне модульного тестирования.

## **Тестирование белого ящика**

**Тестирование белого ящика (white box testing)** — тестирование, основанное на анализе внутренней структуры системы и на знании и понимании исходного кода. К исходному коду тестировщик имеет полный доступ.

Для тестирования методом белого ящика требуется знать язык программирования, на котором написано приложение. Обычно такой вид тестирования применяют разработчики при написании юнит-тестов. Входные значения отбираются на основе кода, который будет их обрабатывать.

Техника белого ящика применяется на разных уровнях тестирования, но главным образом применяется для модульного тестирования компонентов.

### **Преимущества белого ящика**

1. Тестирование производится на ранних этапах: пользовательский интерфейс не требуется.
2. Тестирование более тщательное, с покрытием путей выполнения программы (условий и операторов).

## Недостатки белого ящика

1. Для тестирования требуются специальные знания, в первую очередь — языка программирования.

## Тестирование серого ящика

Тестирование серого ящика (gray box testing) — тестирование в условиях, когда часть внутренней структуры программы известна.

Тестирование методом серого ящика предполагает, что тестировщик работает не с кодом приложения, а с частью его внутренней структуры. Он проверяет запись в базе данных, лог-файлы, а также коды ответа от сервера. Для тестирования веб-приложения методом серого ящика тестировщик использует инструменты разработчика, например, Chrome DevTools.

Техника серого ящика применяется на интеграционном уровне для проверки взаимодействия компонентов программы, например, API-интерфейса и базы данных.

Пример:

Используется при тестировании веб-сайтов на битые ссылки. Если тестер сталкивается с какой-либо проблемой таких ссылок, он может сразу же внести изменения в HTML-код и проверить в реальном времени.

Методы тестирования серого ящика:

1. Матричное тестирование: определение всех переменных, которые есть в программе.
2. Тестирование ортогональных массивов: обеспечивает максимальное покрытие кода с минимальным количеством тестов.
3. Pattern Testing: выполняется на данных истории предыдущих дефектов системы.

## Нефункциональное тестирование

Нефункциональное тестирование (non-functional testing) — анализ свойств компонента или системы, не относящихся к функциональности, то есть проверяется, «как работает система».

Нефункциональное тестирование включает следующие виды тестирования:

1. Тестирование производительности.
  - a. Нагрузочное тестирование.
  - b. Тестирование масштабируемости.
  - c. Объёмное тестирование.

- d. Стрессовое тестирование.
- 2. Тестирование безопасности.
- 3. Инсталляционное тестирование.
- 4. Тестирование интерфейса (GUI/UI-тестирование).
- 5. Тестирование удобства использования.
- 6. Тестирование локализации.
- 7. Тестирование надёжности.

## Тестирование производительности

Тестирование производительности (performance testing) — тестирование для определения работоспособности, стабильности, потребления ресурсов в условиях различных сценариев использования и нагрузок.

Задача системы — обрабатывать требуемое количество данных за установленное время. В случае превышения запланированных объёмов входных данных система восстанавливается после отказа без потери данных.

Например, в требованиях указывается, что система обрабатывает тысячу запросов пользователей в секунду без потери производительности. Чтобы проверить выполнение этого требования, тестировщик формирует тысячу запросов пользователей и направляет их на сервер.

### Нагрузочное тестирование

Нагрузочное тестирование (load testing) — тип тестирования производительности, проводимый с целью оценки поведения системы при возрастающей нагрузке, например, количестве одновременных пользователей или операций, а также для определения нагрузки, которую способны выдержать компонент или система.

Нагрузка повышается до достижения требуемых характеристик, и далее отслеживается поведение на протяжении повышения загрузки системы. При этом происходит:

- измерение времени выполнения операций при определённой интенсивности этих операций;
- определение количества пользователей, одновременно работающих с приложением;
- определение границ приемлемой производительности при увеличении нагрузки, при увеличении интенсивности выполнения этих операций.

## Тестирование масштабируемости

Тестирование масштабируемости (scalability testing) — тестирование программного обеспечения для измерения возможностей вертикального и горизонтального масштабирования, с точки зрения любой из нефункциональных возможностей: увеличение количества пользователей приложения, рост количества транзакций, увеличение объема данных.

Вертикальное масштабирование — это увеличение производительности каждого компонента системы с целью повышения общей производительности. Например, увеличивается объем оперативной памяти на сервере, и тогда он быстрее станет обрабатывать запросы. Это повысит производительность всей системы.

Горизонтальное масштабирование — это разбиение системы на структурные компоненты и разнесение их по отдельным физическим машинам, а также увеличение количества серверов, параллельно выполняющих одну и ту же функцию. Например, увеличивается количество серверов, но каждый выполняет одну и ту же задачу: принимать одни и те же запросы и отвечать на них.

Если разработчики заранее не подумают, как они увеличат его ресурсы при росте популярности, они потеряют значительную часть прибыли.

## Объемное тестирование

Тестирование больших объемов данных. Например, тестируется поведение приложения при попытке загрузить в его базу данных нескольких файлов очень большого размера.

## Стрессовое тестирование

Стрессовое тестирование (stress testing) — вид тестирования производительности, оценивающий систему на граничных значениях рабочих нагрузок или за их пределами, или же в состоянии ограниченных ресурсов — памяти или доступа к серверу.

Например, стандартная нагрузка на сервер приложения — тысяча запросов в секунду. При стрессовом тестировании требуется проверить поведение системы при увеличении нагрузки до 10 тысяч запросов в секунду. Если система не обработает такое количество запросов и отключится, при перезапуске все данные и настройки сохранятся.

## Основные понятия в тестировании производительности

Время задержки (Latency) — временной интервал между запросом и ответом. Если говорят, что у сервиса время задержки составляет 100ms, значит, ему требуется 100 миллисекунд на обработку запроса и ответ. Чем ниже время задержки, тем лучше клиентский опыт.

Время ответа (Response time) — время, требуемое для ответа на запрос.

Пропускная способность (Throughput) — фактическое количество запросов, которое обрабатывает система за определённое время. Метрика пропускной способности показывает объём данных, полученных и обработанных в момент времени.

Важно не отделять показатели времени задержки от пропускной способности. Высокий показатель времени задержки часто напрямую связан с увеличением показателей метрики пропускной способности. Пропускная способность обычно измеряется в rps — количестве запросов в секунду (requests per second).

Ширина пропускания канала (Bandwidth) — максимальное число запросов, обрабатываемых системой. Используется, чтобы измерять максимальный объём, который обрабатывает приложение.

Процент ошибок — отношение невалидных ответов к валидным за промежуток времени.

## Прочие виды нефункционального тестирования

Инсталляционное тестирование (installation testing) — тестирование, направленное на проверку успешной установки и настройки, обновления или удаления приложения при различном программном и аппаратном окружении. Оно позволяет оценить работоспособность системы после завершения работы инсталлятора.

Тестирование интерфейса (GUI/UI testing) — проверка требований к пользовательскому интерфейсу. Например, требований к размещению элементов управления на экране, содержанию и оформлению выводимых сообщений, к форматам ввода, реакции системы на ввод пользователя, ко времени отклика на команды пользователя.

Тестирование удобства использования (usability testing) — проверка того, насколько легко пользователь понимает и осваивает программу, включая не только функциональную составляющую (саму систему), но и её документацию — пользовательские инструкции.

Тестирование локализации (localization testing) — проверка адаптации программного обеспечения для нового места эксплуатации. Включает проверку изменения языка и культурной адаптации.

Например, проверяется, насколько перевод приложения на русский язык корректен, с точки зрения орфографии, грамматики, а также культурных особенностей. Проверка затрагивает все части приложения, в том числе названия кнопок, всплывающие подсказки, надписи над полями. Это касается приложений, которые планируется внедрять в разных странах.

Тестирование безопасности (security testing) — тестирование программного продукта, чтобы определить его защищённость. Основные понятия, охватываемые тестированием: конфиденциальность, целостность и сохранность данных, аутентификация, авторизация и невозможность отказа от авторства (атрибуты качества).



Проводится для тех объектов, в работе которых обеспечение защищённости — одна из важнейших задач.

Тестирование надёжности (reliability testing) — тестирование способности приложения выполнять свои функции в заданных условиях на протяжении заданного времени или установленного количества операций.

Неважно, как долго идёт это тестирование, основная задача — наблюдая за потреблением ресурсов в течение определённого времени, выявить утечки памяти и проследить, чтобы скорость обработки данных или время отклика в начале теста и с течением времени не уменьшалась. В противном случае вероятны сбои в работе продукта и перезагрузки системы.

## Прочие разновидности тестирования

### Виды тестирования, связанные с изменениями в коде

Изменения в код приложения вносятся на протяжении всего процесса разработки: при добавлении новых функций и при исправлении дефектов. В результате тестировщик снова проводит тестирование той части приложения, которая была проверена, но подверглась изменениям. В зависимости от рода этих изменений выделяют регрессионное и повторное тестирование.

Регрессионное тестирование (regression testing) — тестирование уже проверенной функциональности после внесения изменений в код приложения для уверенности, что эти изменения не внесли или не активизировали ошибки в областях, которые не подвергались изменениям.

Повторное или подтверждающее тестирование (re-testing/confirmation testing) — тестирование, во время которого исполняются тестовые сценарии, выявившие ошибки во время последнего запуска, для подтверждения успешности исправления этих ошибок. Это проверка, что дефект исправлен, и приложение после исправления работает в соответствии с требованиями.

Повторное тестирование — обязательный этап тестирования, так как после исправления дефекта разработчиком тестировщик должен это проверить, повторив тот сценарий, который выявил ошибку.

### Статическое и динамическое тестирование

По критерию запуска кода программы на исполнение выделяют два вида тестирования: статическое и динамическое.

Статическое тестирование (static testing) — тестирование системы на уровне спецификации или реализации без исполнения кода программного продукта. Так проводится тестирование:

- документации: требований, схем баз данных, тест-кейсов;

- кода приложения: проверка кода перед запуском специалистом, не участвовавшим в его написании или изменении, то есть аудит кода, или code review;
- параметров настройки среды приложения;
- подготовленных тестовых данных;
- прототипов пользовательского интерфейса.

Статическое тестирование начинается на ранних этапах жизненного цикла ПО и продолжается на протяжении всего процесса разработки.

Динамическое тестирование (dynamic testing) — тестирование, проводимое во время выполнения программного обеспечения, компонента или системы. Проверка — реальное поведение ПО во время его работы.

Для выполнения динамического тестирования требуется, чтобы код программы запустился. При этом тестируется как система в целом, так и отдельные компоненты. Все виды функционального тестирования — динамические.

## Позитивное и негативное тестирование

Есть два вида работы с приложением:

1. Делать всё по инструкции и проверять, что приложение работает согласно требованиям.
2. Намеренно совершать некорректные действия, чтобы проверить, как приложение будет на них реагировать.

На основании этого выделяются два вида тестирования — позитивное и негативное.

Позитивное тестирование (positive testing) — тестирование с использованием только корректных данных и проверка того, что приложение правильно выполняет вызываемые функции. Как правило, проводится в первую очередь для подтверждения работоспособности объекта тестирования.

Тестировщик полностью следует требованиям и инструкции по работе с приложением. Например, при тестировании формы регистрации в приложении заполняет её корректными данными и нажимает кнопку «Зарегистрироваться».

Негативное тестирование (negative testing) направлено на исследование работы приложения в ситуациях, когда с ним выполняются некорректные операции, или используются данные, потенциально приводящие к ошибкам.

Негативное тестирование — это не попытка «сломать» систему, а проверка системы на правильность обработки некорректных действий пользователя.

1. Если пользователь при регистрации укажет email без символа @, приложение выведет сообщение об ошибке.
2. Если же сообщение об ошибке не появится, и пользователь не зарегистрируется, это будет считаться дефектом.
3. Если банковское приложение для выдачи кредита требует, чтобы возраст заёмщика был больше или равен 18, негативным тестом будет проверка возраста 15 лет, а успешным завершением — сообщение о невозможности выдать кредит из-за возраста, не соответствующего требованиям.

Позитивные тесты проводятся в первую очередь, чтобы убедиться в правильной работе приложения. После этого переходим к негативным проверкам.

## Альфа- и бета-тестирование

В зависимости от того, кто выполняет тестирование на последних этапах перед выпуском продукта на рынок, выделяют альфа- и бета-тестирование.

Альфа-тестирование (alpha testing) — внутреннее пробное использование, выполняется внутри организации-разработчика с возможным частичным привлечением пользователей.

Альфа-тестирование проводится после того, как проведены модульное, интеграционное и системное тестирование, и продукт уже частично готов к выпуску на рынок, но требуется его доработка.

Представляет собой имитацию реального использования продукта пользователями, но выполняется либо командой тестирования, либо другими сотрудниками компании-разработчика в тестовой среде, например, на тестовых стендах компании, недоступных внешним пользователям.

Альфа-тестирование — это внутреннее тестирование, по окончании которого выпускается бета-версия продукта и передаётся на бета-тестирование: внешнее или публичное тестирование.

Бета-тестирование (beta testing) — выполняется вне организации-разработчика с активным привлечением пользователей. Обычно представляет собой форму внешнего приёмочного тестирования.

Для проведения бета-тестирования продукт должен быть стабилен, чтобы передать его пользователям. При этом не исключается появление проблем и выявление недостатков. Поэтому сначала доступ открывают для небольшой группы лояльных пользователей, чтобы проверить работоспособность и получить обратную связь.

Часто бета-тестирование применяется при тестировании игр: ОБТ — открытое бета-тестирование. В этом случае привлекаются либо все желающие (например, заполнившие заявку), либо определённые люди, уже имеющие опыт работы с программами или играми подобного типа. Обычно у компаний есть контакты тех, с кем они постоянно сотрудничают при проведении бета-тестирования.

Иногда бета-версия размещается в конкретной стране или регионе для сбора статистики или получения обратной связи перед тем, как её полностью выведут на рынок.

## Тестирование на основе тест-кейсов и исследовательское тестирование

Тестирование на основе тест-кейсов — *scripted testing, test case based testing* — формализованный подход, в котором тестирование производится на основе заранее подготовленных тест-кейсов. Наиболее часто используемый подход на проектах по разработке ПО, так как позволяет структурировать процесс тестирования, сделать его более контролируемым.

Исследовательское тестирование (*exploratory testing*) — частично формализованный подход, когда тестировщик выполняет работу с приложением по выбранному сценарию. Этот сценарий дорабатывается в процессе выполнения для более полного исследования приложения.

Во время исследовательского тестирования неформальные (не созданные заранее) тестовые сценарии разрабатываются, выполняются, анализируются и оцениваются динамически. Результаты тестирования используются для изучения компонента или системы и последующей разработки тестовых сценариев для непокрытых областей.

Исследовательское тестирование проводится сессиями. Сессия — это выделенный промежуток времени, с которым тестировщик исследует программу, ориентируясь на поставленную цель. Например, требуется проверить все поля ввода на странице. Во время сессии ведётся протокол, а тестировщик фиксирует действия и полученные результаты.

Исследовательское тестирование лучше всего подходит в ситуациях, когда документация недостаточная либо вовсе отсутствует, в условиях очень сжатых сроков и как дополнение к другим, более формальным методам тестирования.

Свободное (интуитивное) тестирование — *ad hoc testing* — неформальный подход, где не предполагается использование ни тест-кейсов, ни чек-листов, ни сценариев. Тестировщик полностью опирается на свой профессионализм и интуицию для спонтанного выполнения проверок.

Чаще всего при таком подходе предполагается, что тестировщик плохо знаком с тестируемым приложением. Этот вид тестирования используется редко и исключительно как дополнение к полностью или частично формализованному тестированию, когда для исследования некоторых функций приложения нет тест-кейсов, либо они ещё не написаны.

## Контрольные вопросы

1. В чём преимущества и недостатки тестирования методом чёрного ящика?
2. В чём преимущества тестирования серого ящика перед тестированием чёрного ящика?

3. Какие виды нефункционального тестирования вы знаете?
4. Какие виды тестирования, связанные с изменениями, вы знаете?

## Практическое задание

1. Откройте документ, созданный в практическом задании №2. На вкладке «Тест-кейсы» для каждого тест-кейса определите тип тестирования (выберите из списка). На странице «Баг-репорты» заведите 3 бага по упавшим тестам. Не повторяйте баги, найденные в чек-листе.
2. Тест для самопроверки - <https://coreapp.ai/app/player/lesson/614344ca40a70c676714c756>  
(сдавать не нужно)

## Глоссарий

**Альфа-тестирование** — alpha testing — внутреннее пробное использование, выполняется внутри организации-разработчика с возможным частичным привлечением конечных пользователей.

**Бета-тестирование** — beta testing — выполняется вне организации-разработчика с активным привлечением пользователей. Обычно представляет собой форму внешнего приёмочного тестирования.

**Динамическое тестирование** — dynamic testing — тестирование, проводимое во время выполнения программного обеспечения, компонента или системы.

**Исследовательское тестирование** — exploratory testing — частично формализованный подход, когда тестировщик выполняет работу с приложением по выбранному сценарию, который дорабатывается в процессе выполнения для более полного исследования приложения.

**Нагрузочное тестирование** — load testing — тип тестирования производительности, проводимый с целью оценки поведения системы при возрастающей нагрузке, например, количестве одновременных пользователей или операций. Это тестирование проводится также для определения нагрузки, которую способны выдержать компонент или система.

**Нефункциональное тестирование** — non-functional testing — анализ атрибутов качества компонента или системы, не относящихся к функциональности, то есть проверка работы системы.

**Повторное или подтверждающее тестирование** — re-testing или confirmation testing — тестирование, во время которого исполняются тестовые сценарии, выявившие ошибки во время последнего запуска, для подтверждения успешности исправления этих ошибок.

**Позитивное тестирование** — positive testing — тестирование с использованием только корректных данных и проверка того, что приложение правильно выполняет вызываемые функции.

**Регрессионное тестирование** — regression testing — тестирование уже проверенной функциональности после внесения изменений в код приложения.

**Статическое тестирование** — static testing — тестирование системы на уровне спецификации или реализации без исполнения кода программного продукта. То есть тестируемое приложение не функционирует, или для проведения проверки его запуск не требуется.

**Стрессовое тестирование** — stress testing — вид тестирования производительности, оценивающий систему на граничных значениях рабочих нагрузок или за их пределами, либо в состоянии ограниченных ресурсов — памяти или доступа к серверу.

**Тестирование белого ящика** — white box testing — тестирование, основанное на анализе внутренней структуры компонента или системы, а также на знании и понимании исходного кода, к которому тестировщик (обычно программист) имеет полный доступ.

**Тестирование масштабируемости** — scalability testing — тестирование программного обеспечения для измерения возможностей вертикального и горизонтального масштабирования, с точки зрения любой из нефункциональных возможностей: увеличение количества пользователей приложения, рост количества транзакций, увеличение объёма данных.

**Тестирование на основе тест-кейсов** — scripted testing, test case based testing — формализованный подход, где тестирование производится на основе заранее подготовленных тест-кейсов.

**Тестирование производительности** — performance testing — комплекс видов тестирования, цель которого — определение работоспособности, стабильности, потребления ресурсов и других атрибутов качества приложения в условиях различных сценариев использования и нагрузок.

**Тестирование серого ящика** — gray box testing — тестирование, ориентированное на имитацию работы пользователей в условиях, когда часть внутренней структуры программы известна.

**Тестирование чёрного ящика** — black box testing — тестирование, основанное на анализе функциональной или нефункциональной спецификации системы без знания внутренней структуры.

## Дополнительные материалы

1. Статья [«Ликбез по уязвимостям в веб-приложениях, а также самые частые ошибки разработчиков»](#).
2. Статья [«SQL-инъекции. Проверка, взлом, защита»](#).
3. Статья [«UI controls на русском»](#).
4. Статья [«В чём разница Smoke, Sanity, Regression, Re-test и как их различать?»](#).
5. Статья [«Тестирование пользовательского интерфейса»](#).

6. Статья [«Особенности тестирования чёрного ящика»](#).
7. Статья [«Тестирование локализации»](#).
8. Статья [«Тестирование безопасности: изнутри и снаружи»](#).
9. Статья [«Тестирование установки \(Installation Testing\)»](#).