



ТЕСТИРОВАНИЕ ПО. МНОГООБРАЗИЕ ТЕСТИРОВАНИЯ



АНАСТАСИЯ ШАРИКОВА



АНАСТАСИЯ ШАРИКОВА

QA Lead в Bookmate



shharikova@gmail.com



[@shharikova](https://www.instagram.com/shharikova)



ПЛАН ЗАНЯТИЯ

1. [О тест дизайне](#)
2. [Разбор основных методов тест дизайна](#)
3. [Параметры принятия софта на тест и немного про альфа- и бета-тестирование](#)

О ТЕСТИРОВАНИИ



Тестирование программ может использоваться для демонстрации наличия ошибок, но оно никогда не покажет их отсутствие.

Дейкстра, 1970 г.

ЗАЧЕМ НУЖНО ТАК МНОГО РАЗНОГО ТЕСТИРОВАНИЯ?

Ответ кроется в окружающем мире. Все мы на самом деле каждый день сталкиваемся с багами, особенно из-за того, что нас все больше окружают технологии.

Приведу пример случая, который встретился мне на днях. Тут явно упустили из виду тестирование методом эквивалентных значений:

воскресенье, Сегодня

Финал ЧМ по хоккею на экране вашего смартфона! Болеем за наших! Матч Россия-<наименование команды> смотрите [сегодня](#) на <наименование канала> канале в [21:15](#) в приложении МТС ТВ совершенно бесплатно. Трафик не расходуется. Скачайте: mtstv.ru/promo

20:14

ПРО ISTQB

Каждому, кто работает с обеспечением качества, надо знать эту аббревиатуру: **ISTQB® – International Software Qualifications Board.**

Что такое ISTQB?

ISTQB® - некоммерческая организация, занимающаяся определением различных принципов развития сферы тестирования ПО, таких как структура и правила аккредитации, сертификации и т.п.

Во-первых, из их глоссария и программы можно узнавать определения разных терминов, во-вторых большое количество фирм во всем мире (но не в РФ) берет на работу преимущественно сертифицированных по этой программе специалистов.



О ТЕСТ ДИЗАЙНЕ

ЧТО ТАКОЕ ТЕСТ ДИЗАЙН?

Тест дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования (определение по ISTQB). В русском языке еще часто встречается название «проектирование тестов».

Задача тест дизайна – понять, как тестировать и что тестировать.



ЦЕЛИ ТЕСТ ДИЗАЙНА

Обеспечить покрытие функционала приложения тестами:

- Тесты должны покрывать весь функционал;
- Тестов должно быть минимально достаточно.

Важно замечание: тесты должны покрывать весь функционал, но это не значит, что *всегда* надо проверять *все* тесты.



ТЕСТОВОЕ ПОКРЫТИЕ

Тестовое Покрытие (Test Coverage) — это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.



ТЕСТОВОЕ ПОКРЫТИЕ

- Покрытие требований (Requirements Coverage) — оценка покрытия тестами функциональных и нефункциональных требований к продукту путем построения матриц трассировки (traceability matrix).
- Покрытие кода (Code Coverage) — оценка покрытия исполняемого кода тестами, путем отслеживания непроверенных в процессе тестирования частей программного обеспечения.
- Тестовое покрытие на базе анализа потока управления — оценка покрытия, основанная на определении путей выполнения кода программного модуля и создания выполняемых тест кейсов для покрытия этих путей.



ЗАДАЧИ ТЕСТ ДИЗАЙНА

Когда мы тестируем, важно:

- Проанализировать требования к продукту;
- Понять, какие нас ждут риски и особенности использования продукта;
- Написать достаточное минимальное количество тестов (те, которые покрывают основной функционал);
- Разделить тесты на приемочные, критические, расширенные наборы.

ЧТО ВАЖНО ПРИ НАПИСАНИИ ТЕСТ КЕЙСОВ И ТЕСТ ПЛАНОВ?

- Грамотный и понятный русский язык;
- Писать тесты так, чтобы человек, который их видит впервые, все понял;
- Не забывать о тестовых учетках и подробном описании тестового окружения, нужного для прохождения;
- Умение декомпозировать сервис на компоненты – например «настройки», «авторизация» и так далее;
- Расстановка приоритетов: например, проверить оплату и авторизацию обычно важнее, чем смену фоновой картинки;
- Умение применить нужные техники тест дизайна.



МЕТОДИКИ ТЕСТ ДИЗАЙНА

Методики тест дизайна — это рекомендации, советы и правила, по которым стоит разрабатывать тест для проведения тестирования приложения.

Даже на одном и том же проекте два тестировщика могут применять к одной задаче разные методики, в зависимости от того, какую сочли более уместной.

Методики тест дизайна включают в себя многочисленные методы, о которых я расскажу далее.

ВЫБОР МЕТОДА ПРОЕКТИРОВАНИЯ ТЕСТОВ

Выбор конкретного метода проектирования тестов зависит от множества факторов, включая:

- Тип системы или компонента и ее сложность;
- Нормативные документы;
- Требования пользователей или контракта;
- Уровни и типы рисков;
- Задачи тестирования;
- Знания и опыт тестировщиков;
- Доступные инструменты и документацию;
- Ресурсы времени и бюджет;
- Особенности жизненного цикла разработки;
- Ожидаемое использование системы;
- Прошлый опыт использования методов проектирования тестов для компонента или системы;
- Ожидаемые типы дефектов компонента или системы.



РАЗБОР ОСНОВНЫХ МЕТОДОВ ТЕСТ ДИЗАЙНА



МЕТОДЫ ТЕСТ ДИЗАЙНА

1. Метод черного ящика:
 - Эквивалентные значения;
 - Граничные значения;
 - Прочие методики.
2. Метод белого ящика.
3. На основе опыта:
 - Метод предположения об ошибках;
 - Метод исследовательского тестирования;
 - Тестирование на основе чек-листов.



ТЕСТИРОВАНИЕ МЕТОДОМ ЧЕРНОГО ЯЩИКА

Методы черного ящика (поведенческие методы или методы, основанные на поведении) основываются на анализе соответствующего базиса тестирования (формальных требований, спецификаций, сценариев использования, пользовательских историй или бизнес-процессов).

Эти методы применимы как для функционального, так и для нефункционального тестирования. Методы черного ящика сосредотачиваются на связи входных данных и выходных результатов объекта тестирования, а не на его внутренней структуре.

Дальше мы рассмотрим примеры таких методов.



ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ

Эквивалентное разбиение, которое применяется в одном из методов, делит данные на группы (классы эквивалентности), которые обрабатываются схожим образом. Области эквивалентности могут быть позитивными и негативными.

Пример:


Если в форму на сайте можно ввести возраст от 1 до 99 лет, то для проверки по эквивалентным значениям мы выберем 0 и 50 для ввода.

ЭКВИВАЛЕНТНЫЕ ЗНАЧЕНИЯ

Разберем на примере из реального таска:

Description

Если пользователь попытался войти больше допустимого количество раз — сервер начнет отвечать статусом **429 (Too many requests)** и заголовком **Retry-After: <delay-seconds>**, где **delay-seconds** — это количество секунд, через которое можно будет повторить запрос.

Необходимо обрабатывать такой ответ и объяснить пользователю что происходит, возможно, дать возможность восстановить пароль. Тут  допишет подробности.

ГРАНИЧНЫЕ ЗНАЧЕНИЯ

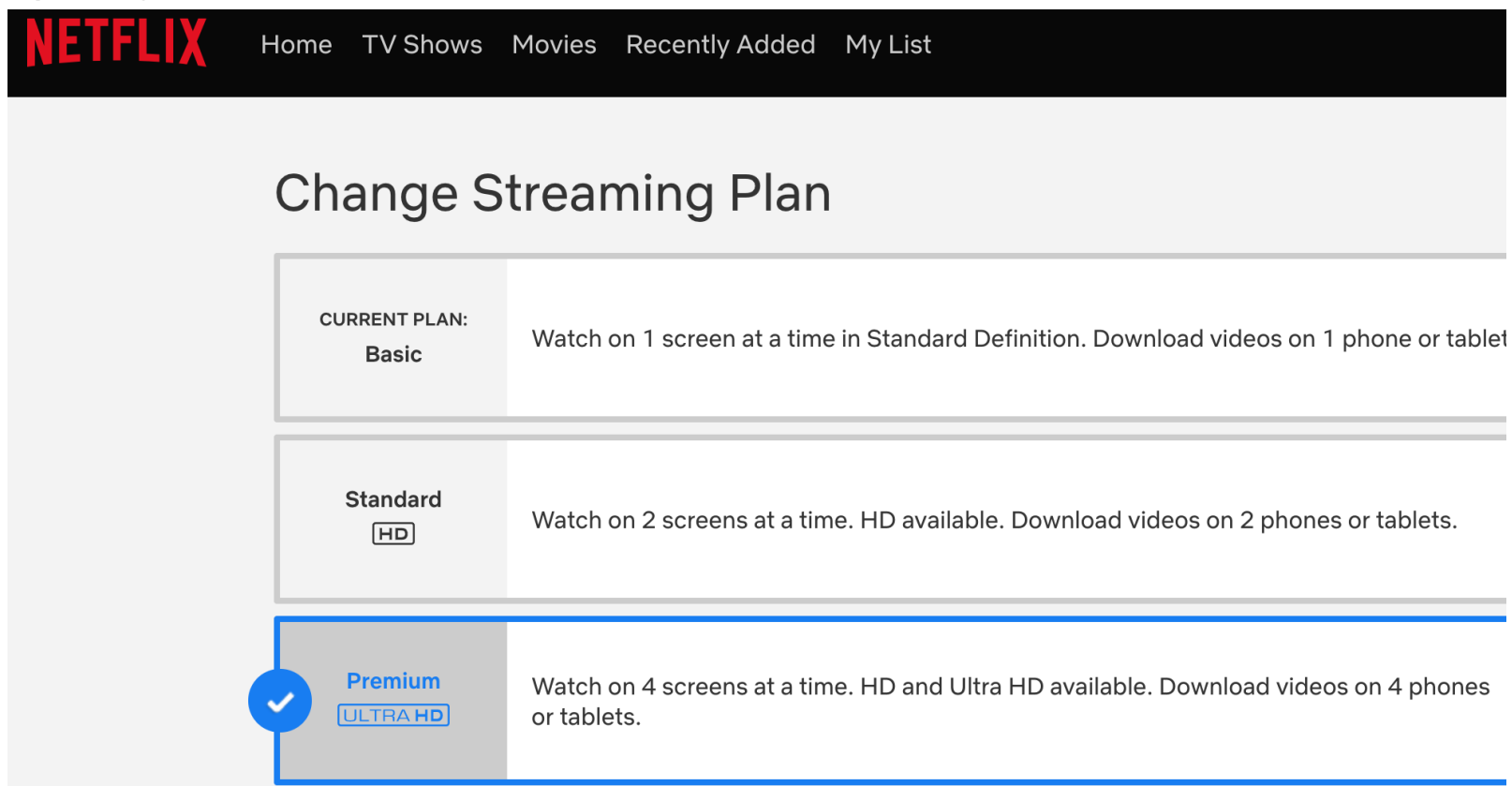
Метод анализа граничных значений является продолжением метода эквивалентного разбиения, но может быть применим, только если классы состоят из упорядоченных числовых значений.

Максимальное и минимальное значения класса являются его границами.

Пример:


В случае формы из предыдущего примера этими значениями будет 0, 1, 2 и 98, 99, 100.

Реальный пример: как бы мы протестировали ограничение устройств на премиум плане Netflix?



NETFLIX Home TV Shows Movies Recently Added My List

Change Streaming Plan

CURRENT PLAN: Basic	Watch on 1 screen at a time in Standard Definition. Download videos on 1 phone or tablet
Standard HD	Watch on 2 screens at a time. HD available. Download videos on 2 phones or tablets.
 Premium ULTRA HD	Watch on 4 screens at a time. HD and Ultra HD available. Download videos on 4 phones or tablets.



ПРОЧИЕ МЕТОДЫ

- Таблица альтернатив;
- Таблица переходов;
- Тестирование с помощью вариантов использования (Use Case);

МЕТОД БЕЛОГО ЯЩИКА

Методы белого ящика (структурные методы или методы, основанные на структуре) основаны на анализе архитектуры, детального проектирования, внутренней структуры или кода компонента либо системы. В отличие от методов черного ящика методы белого ящика сосредотачиваются на структуре и обработке внутри объекта тестирования.

Более актуальны для опытных ручных тестировщиков и автоматизаторов, но важно о них знать.

БЕЛЫЙ ЯЩИК: ТЕСТИРОВАНИЕ И ПОКРЫТИЕ ОПЕРАТОРОВ И УСЛОВИЙ

Проверка методом тестирования операторов* направлена на проверку исполняемых операторов в коде. Фактически, суть в том, что в тесте должен быть задействован каждый оператор кода хотя бы раз.

Тестирование условий направлено на проверку логических условий в коде, а также кода, выполняемого в зависимости от исхода условия.

*Оператор — это элемент языка, задающий полное описание действия, которое необходимо выполнить. Каждый оператор представляет собой законченную фразу языка программирования и определяет некоторый вполне законченный этап обработки данных.

НА ОСНОВЕ ОПЫТА



Knowledge gained through experience is far superior and many times more useful than bookish knowledge

Mahatma Gandhi

Методы, основанные на опыте, используют опыт разработчиков, тестировщиков и пользователей для проектирования, реализации и выполнения тестов. Их часто совмещают с методами черного и белого ящиков.

Далее рассмотрим некоторые из них.

МЕТОД ПРЕДПОЛОЖЕНИЯ ОБ ОШИБКАХ

Предположение об ошибках — это способ предотвращения багов, основанный на знаниях тестировщика, включающих:

- Историю работы приложения в прошлом;
- Наиболее вероятные типы дефектов, допускаемых при разработке;
- Типы дефектов, которые были обнаружены в схожих приложениях;
- Данные технического задания.

Например, в ТЗ написано, что “Когда с сервера приходит картинка, мы ее показываем”. Мы в таком случае проверяем, а что будет, если она не придет или придет не картинка, а видео.

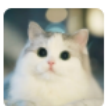
МЕТОД ПРЕДПОЛОЖЕНИЯ ОБ ОШИБКАХ

Реалистичный пример: когда разработчики сделали сборку для релиза и уверены, что раз там исправили два маленьких бага, то они ничего серьезного не сломали. Но опытный тестировщик знает, что обычно в половине случаев все ломается после сборки, и все равно все будет проверять очень внимательно.



Tina Kropaneva 11:58 AM

думаю, два краша они смогли норм вмерджить))



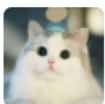
Margarita Nazarova 11:58 AM

нууу

я б не была так уверена



Tina Kropaneva 11:58 AM



Margarita Nazarova 11:58 AM

много чего могло пойти не так

МЕТОД ИССЛЕДОВАТЕЛЬСКОГО ТЕСТИРОВАНИЯ/AD-НОС ТЕСТИРОВАНИЕ

Во время исследовательского тестирования мы решаем, что проверяем динамически во время выполнения тестов.

Этот вариант особенно подходит в условиях отсутствия документации или при сжатых сроках на тестирование.

Пример: вышел новый закон о пользовательских данных и из-за этого надо срочно добавить новую галочку в регистрации — в этом случае мы будем тестировать методом исследовательского тестирования, а потом уже писать нормальные тесты после релиза.



ТЕСТИРОВАНИЕ НА ОСНОВЕ ЧЕК-ЛИСТОВ

При тестировании по чек-листам тестировщик проектирует, реализует и выполняет тесты, покрывающие тестовые условия, указанные в чек-листе.

В качестве составной части анализа тестировщики могут создавать новые чек-листы, расширять их либо использовать готовые чек-листы, не меняя их.

Такие списки могут быть построены на опыте, на исторических данных об ошибках, на информации о приоритетах для пользователей и понимании, как и почему происходят отказы в программе.

ТЕСТИРОВАНИЕ НА ОСНОВЕ ЧЕК-ЛИСТОВ

Пример чек-листа: реальный чек-лист к задаче, в которой делали новые пейволлы (страница в приложении, на которой пользователю предлагают купить подписку).

	ios уже посмотрели	ios ок баров нет	android уже посмотрели	andr. баров нет	web посмотрели
аналитика	+	+			+
переводы защитные в клиент					
переводы приходящие с сервера					
пейвол на онбординге	+	+	+	-	-
пейвол на входе	+	+	+	+	-
пейвол в книге	+	+	+	+ -	начали смотреть
пейвол со всеми подписками / страница со всеми подписками (веб)					+
обновление с предыдущей версии					-
партнерские пейволлы	+	+			
новые юзеры	+	+	+	+	+
старые без подписки					+
старые с обычной на книги					
старые с премиумом					
старые с аудио					
старые с комбинациями					
старые на более старых версиях					-
динамические лейблы					
соответствие макетам по дизайну					
цены верные					
планы ок					
все виды оплат как ранее, работают					

ПРИМЕР ИСПОЛЬЗОВАНИЯ ТЕСТ ДИЗАЙНА

На скриншоте список тест кейсов (внутри они заполнены по шагам и ожидаемому поведению), в которых в основном проверяются основные пользовательские сценарии, т.е. минимальный набор в случае приемочного теста:

Авторизация через tw, у пользователя есть приложение tw (пользователь там залогинен)

Авторизация через tw (web сайт) - приложения tw нет - пользователь не авторизован на tw через web

Повторная авторизация на tw

Отказ доступа к аккаунту на tw

Авторизация через fb, у пользователя есть приложение fb (пользователь там залогинен)

Авторизация через fb (web сайт) - приложения fb нет - пользователь авторизован на fb через web

Повторная авторизация на fb

Отказ доступа к аккаунту на fb

Авторизация через vk, у пользователя есть приложение vk (пользователь там залогинен)

Авторизация через vk (web сайт) - приложения vk нет - пользователь не авторизован на vk через web

Повторная авторизация на vk

Отказ доступа к аккаунту на vk

Авторизация через соц. сети без интернет соединения

ПАРАМЕТРЫ ПРИНЯТИЯ СОФТА НА ТЕСТ И НЕМНОГО ПРО АЛЬФА- И БЕТА-ТЕСТИРОВАНИЕ

АЛЬФА-ТЕСТИРОВАНИЕ И БЕТА-ТЕСТИРОВАНИЕ

Альфа-тестирование и бета-тестирование обычно используются разработчиками уже готового продукта, которые хотят получить обратную связь от потенциальных или существующих пользователей, как только продукт будет выставлен в коммерческую продажу.

Цели альфа- и бета-тестирования:

- получение уверенности в том, что система работает, до ее запуска;
- обнаружение багов, связанных с условиями и средой эксплуатации;
- привлечение пользователей, заинтересованных в эксклюзивности (например, в играх);
- проверка нужности некоего функционала в целом.

ПАРАМЕТРЫ ПРИНЯТИЯ ПРОДУКТА НА ТЕСТИРОВАНИЕ

Затронем тему, которую часто не упоминают: когда мы можем начать тестирование продукта?

Говорим мы о новом функционале или о исправленных багах — важно контролировать, чтобы у нас были все доступные данные по нашей задаче. Это нужно для того, чтобы в полной мере понимать, какие методики нам применять и что именно от нас нужно в данный момент.


Пример, того что стоит знать по задаче:


- Подробное описание того, как должен работать функционал;
- Описание окружения, на котором мы можем проверить сейчас и на котором будет работать потом;
- Связанные задачи, если они есть;
- Тестовые данные, если они нужны, например, учетки или некие данные.

ПАРАМЕТРЫ ПРИНЯТИЯ ПРОДУКТА НА ТЕСТИРОВАНИЕ

Пример, как **НЕ** надо

Всё проверить

 Edit

 Comment

Assign

Reopen Issue

To Idea

Workflow ▾

Description

Когда  сверстают игру, проверить тексты и вообще всё.

Мой совет на будущее — контролируйте, чтобы вся нужная информация появилась на самом начальном этапе, еще до начала тестирования.

КРАТКИЕ ВЫВОДЫ

При всем многообразии вариантов тестирования, важно помнить одно – наша задача, как специалиста, не найти баги, а удостовериться в том, что все работает, как задумано. И почаще напоминать об этом разработке, конечно ;)

Помимо этого, вопрос, который чаще всего поможет понять, в каком направлении стоит двигаться – это **«А что, если?»**

Например, если написано, что в поле надо ввести возраст, нам нужно подумать – а что, если ввести пустое значение? Или букву? Или иероглиф? И так далее.

[illegible]



ПОЛЕЗНЫЕ ССЫЛКИ

- [Официальный сайт ISTQB](#)
- [Баги, которые не будут исправлены](#)
- [Видео про важность информации для тестирования](#)
- [Видео о том, как нам жить вместе с разработкой \(с примерами!\)](#)
- [О валидации и верификации](#)
- [О авторизации, аутентификации и индентификации](#)

ДОМАШНЕЕ ЗАДАНИЕ

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задаем в чате в Slack.
- Задания можно сдавать по частям.
- Также пройдите тестирование в личном кабинете Нетологии.
- Зачет по домашней работе проставляется после того, как приняты **все задания**



Спасибо за внимание!

Время задавать вопросы ☐

АНАСТАСИЯ ШАРИКОВА



shharikova@gmail.com



[@shharikova](https://www.instagram.com/shharikova)