ТЕСТИРОВЩИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

КУРС «РУЧНОЕ ТЕСТИРОВАНИЕ»

9. ТЕХНИКИ ТЕСТ-ДИЗАЙНА

Основные понятия, цели тест-дизайна.

Основные техники тест-дизайна. Правила, плюсы и минусы.

Тест-дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи, в соответствии с определёнными ранее критериями качества и целями тестирования.

Техники тест дизайна - это рекомендации, советы и правила по которым стоит разрабатывать тест для проведения тестирования приложения. Это не образцы тестов, а только рекомендации к применению. В частности, различные инженеры могут работая под одним и тем же проектом создать различный набор тестов. Правильным будет считаться тот набор тестов, который за меньшее количество проверок обеспечит более полное покрытие тестами.

ОСНОВНЫЕ ПОНЯТИЯ

Тестовое покрытие - это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.

Существуют следующие подходы к оценке и измерению тестового покрытия:

- **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.

ОСНОВНЫЕ ПОНЯТИЯ

Основная цель тест-дизайна — структурировать процедуры тестирования, чтобы было легче отслеживать покрытие требований тест-кейсами. Благодаря тест-дизайну мы:

- создаем тесты, помогающие выявлять серьезные ошибки
- вдумчиво подходим к тестированию и не тратим ресурсы впустую
- сводим к минимуму количество тестов, необходимых для тестирования продукта.

ЦЕЛИ ТЕСТ- ДИЗАЙНА

Задачи:

- Проанализировать требования к продукту
- Оценить риски возможные при использовании продукта
- Написать достаточное минимальное количество тестов
- Разграничить тесты

ЗАДАЧИ ТЕСТ- ДИЗАЙНА

Тест дизайн скиллы профессионала:

- Декомпозиция приложения — разбивание тестируемой системы на компоненты
- Навыки и способности поиска информации для приложения
- Расстановка приоритетов
- Грамотная речь и верный вектор мыслительного процесса
- Знание техник тест дизайна
- Отточенное мастерство применения техник тест дизайна

ТЕСТ- ДИЗАЙН, НЕОБХОДИМЫЕ НАВЫКИ

«А теперь давайте еще раз, но простыми словами…**техники тест-дизайна** – это совокупность правил, позволяющих правильно определить список проверок для тестирования. И самое важное, это ~~использовать эти правила всегда и везде :)~~ уметь на интуитивном уровне применять данные правила. Именно умение «проводить аналитику в голове» отличает хорошего тестировщика!»

ПРОМЕЖУТОЧНОЕ ЗАКЛЮЧЕНИЕ

- Эквивалентное Разделение (Equivalence Partitioning – EP).
- Анализ Граничных Значений (Boundary Value Analysis – BVA).
- Таблица принятия решений (Decision Table)
- Попарное тестирование (Pairwise Testing - PT)
- Причина / Следствие (Cause/Effect - CE).
- Предугадывание ошибки (Error Guessing - EG).

ОСНОВНЫЕ ТЕХНИКИ ТЕСТ-ДИЗАЙНА

Метод эквивалентного разбиения позволяет минимизировать число тестов, не создавая сценарий для каждого возможного значения, а выбрав только одно значение из целого класса и приняв за аксиому, что для всех значений в данной группе результат будет аналогичным.

Эквивалентное разделение, алгоритм использования техники:

1. Необходимо определить класс эквивалентности. Это главный шаг техники. От него во многом зависит эффективность её применения.
2. Затем нужно выбрать одного представителя от каждого класса. На этом шаге из каждого эквивалентного набора тестов мы выбираем один тест.
3. Нужно выполнить тесты. На этом шаге мы выполняем тесты от каждого класса эквивалентности.

ЭКВИВАЛЕНТНОЕ РАЗБИЕНИЕ

Если мы выбираем в качестве техники тест-дизайна эквивалентное разделение, это означает, что мы будем тестировать только несколько значений из каждого класса элементов. Помните, что это не гарантирует отсутствия ошибок в остальных значениях, не охваченных тестами. Мы лишь *предполагаем*, что использование нескольких элементов из каждой группы будет достаточно показательным.

Эквивалентное разделение — хорошее решение для случаев, когда вы имеете дело с большим объемом входящих данных или множеством одинаковых вариантов ввода. В противном случае, возможно, имеет смысл более тщательно охватить продукт тестами

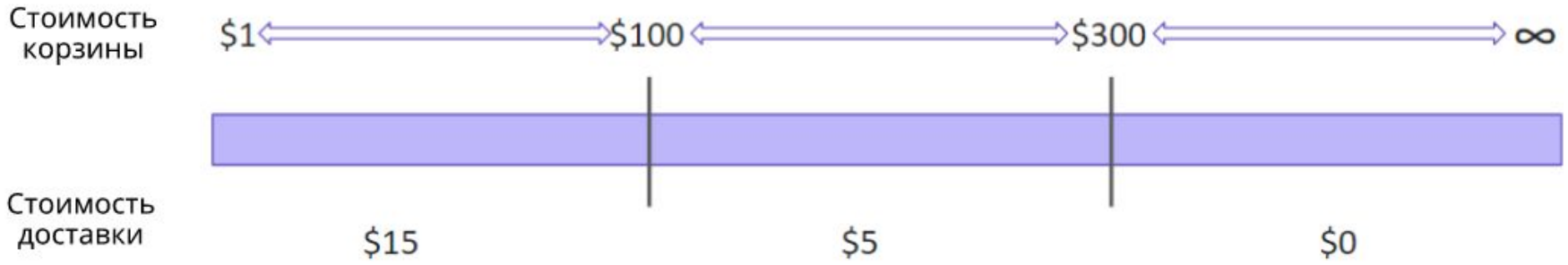
ЭКВИВАЛЕНТНОЕ РАЗБИЕНИЕ

Допустим, есть интернет-магазин, который предлагает разные тарифы на доставку в зависимости от стоимости корзины:

- Стоимость доставки для заказов на сумму менее $100 составляет $15.
- Стоимость доставки для заказов на сумму более $100 составляет $5.
- При заказе от $300 долларов доставка бесплатна.

У нас есть следующие ценовые категории для работы:

- от $1 до $100.
- от $100 до $300.
- $300 и выше.



При использовании техники эквивалентного разделения мы получаем три набора данных для тестирования:

I. От $1 до $100:

- допустимые значения: любая цена в диапазоне от 1 до 99,99
- недопустимые значения: любая цена ниже 1 или выше 99,99

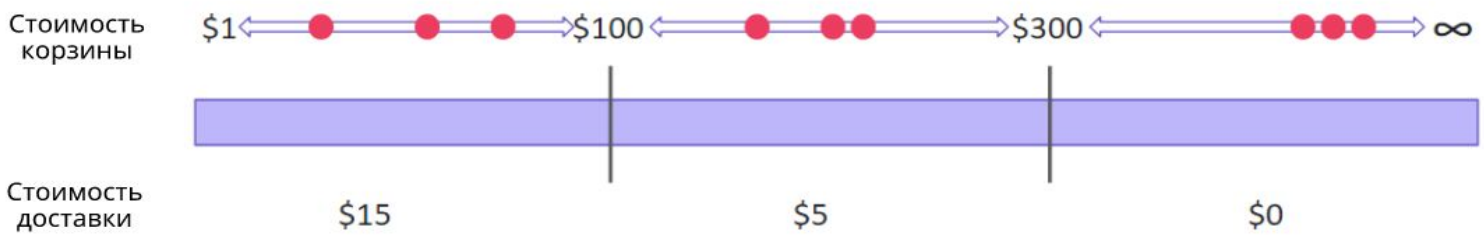
2. От $100 до $300:

- допустимые значения: любая цена в диапазоне от 100 до 299,99
- недопустимые значения: любая цена ниже 100 или выше 299,99

3. От $300 и выше:

- допустимые значения: любая цена выше 299,99;
- недопустимые значения: любая цена ниже 300.

Таким образом, мы можем выбрать несколько чисел из каждого диапазона цен и предположить, что остальные числа из этих диапазонов будут давать такие же результаты.



Плюсы и минусы техники анализа классов эквивалентности

- К плюсам можно отнести заметное сокращение времени и улучшение структурированности тестирования.
- К минусам можно отнести то, что, при неправильном использовании техники, мы рискуем потерять баги.

ЭКВИВАЛЕНТНОЕ РАЗБИЕНИЕ

Анализ граничных значений в чем-то похож на эквивалентное разделение. Можно даже сказать, что оно лежит в основе анализа граничных значений. Но есть некоторые отличия.

При анализе граничных значений мы тоже группируем данные по эквивалентным классам, но проверяем не значения из определенного класса, а граничные значения — те, которые находятся на «границах» классов.

Эта логика применяется для интеграционного тестирования. Мы проверяем отдельные элементы во время юнит-тестирования, а на следующем уровне ошибки, скорее всего, появятся на «стыках» юнитов.

ТЕХНИКИ АНАЛИЗА ГРАНИЧНЫХ ЗНАЧЕНИЙ

ПРИМЕР: Возьмем предыдущий сценарий с различными тарифами на доставку. У нас те же данные, но другой подход к их использованию. Предполагая, что ошибки наиболее вероятны на границах диапазонов, мы тестируем только «граничные» числа:

От $1 до $100:

- допустимые граничные значения: 1,00, 1,01, 99,99
- недопустимые граничные значения: 0,99, 100,00, 100,01

$300 и выше:

- допустимые граничные значения: 300,00, 300,01;
- недопустимые граничные значения: 299,99.

От $100 до $300:

- допустимые граничные значения: 100,00, 100,01, 299,99;
- недопустимые граничные значения: 99,99, 300,00;

Другое название метода – матрица принятия решений. Эта техника подходит для более сложных систем, например – двухфакторной аутентификации. Предположим, чтобы войти в систему, пользователю нужно ввести сначала логин и пароль, а затем еще подтвердить свою личность присланным в смс кодом.

Какие возможны сценарии:

1. Правильный логин и правильный пароль.
2. Правильный логин, неправильный пароль.
3. Неправильный логин, правильный пароль.
4. Неправильный логин, неправильный пароль.

ТАБЛИЦА ПРИНЯТИЯ РЕШЕНИЙ

Первый из этих сценариев сопровождается либо правильным, либо неправильным вводом смс-кода, итого у нас получается 5 тестов. При этом только один из сценариев приведет к положительному результату (пользователь успешно авторизуется), а остальные закончатся неудачей.

Однако, может быть так, что система выдает разные сообщения в зависимости от того, на каком этапе была допущена ошибка, скажем: invalid login, invalid password. Соответственно, групп потребуется больше, а таблица станет обширнее.

Этот метод хорош тем, что он показывает сразу все возможные сценарии в форме, понятной даже неспециалисту.

ТАБЛИЦА ПРИНЯТИЯ РЕШЕНИЙ

Условия	Значения	Правила				
Логин	Верный, Неверный	Верный	Верный	Верный	Неверный	Неверный
Пароль	Верный, Неверный	Верный	Верный	Неверный	Неверный	Верный
Код	Верный, Неверный, Не пришел	Верный	Неверный	Не пришел	Не пришел	Не пришел
Действия	Пользователь вошел на сайт Пользователь не авторизован	Пользователь вошел на сайт	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован

Условия	Значения	Правила			
Логин	Верный, Неверный	Верный	Неверный	Верный, Неверный	Верный
Пароль	Верный, Неверный	Верный	Верный, Неверный	Неверный	Верный
Код	Верный, Неверный, Не пришел	Верный	Верный, Неверный, Не пришел	Верный, Неверный, Не пришел	Неверный
Действия	Пользователь вошел на сайт Пользователь не авторизован	Пользователь вошел на сайт	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован

ПРИМЕР ТАБЛИЦЫ ПРИНЯТИЯ РЕШЕНИЙ

Суть этого метода, также известного как **pairwise testing**, в том, что каждое значение каждого проверяемого параметра должно быть протестировано на взаимодействие с каждым значением всех остальных параметров. После составления такой матрицы мы убираем тесты, которые дублируют друг друга, оставляя максимальное покрытие при минимальном необходимом наборе сценариев.

Попарное тестирование позволяет обнаружить максимум ошибок без избыточных проверок.

ПОПАРНОЕ ТЕСТИРОВАНИЕ

Допустим, есть сеть пекарен, продающих яблочные пироги и чизкейки онлайн. Каждый товар доступен в трех размерах – маленьком, среднем и большом. Пекарня предлагает доставку, как немедленную, так и к определенному времени, а также возможность самовывоза. Пекарня работает в трех городах – Нью-Йорке, Лос-Анджелесе и Чикаго. Также пользователь может заказать до трех товаров одновременно.

Заказ	Размер	Город	Количество	Доставка	Время
Яблочный пирог	Маленький	Нью-Йорк	1	Адресная	Немедленно
Чизкейк	Средний	Лос-Анджелес	2	Самовывоз	К указанному времени
	Большой	Чикаго	3		

ПОПАРНОЕ ТЕСТИРОВАНИЕ. ПРИМЕР

ПОПАРНОЕ ТЕСТИРОВАНИЕ. ПРИМЕР

Если вы захотите протестировать все возможные варианты, у вас будет 2x3x3x3x2x2=216 комбинаций. Но проверять каждую нет смысла. Вместо этого вы можете выстроить переменные таким образом, чтобы охватить максимум сценариев.

Для этого вам нужно сгруппировать переменные или использовать какой-нибудь инструмент, который сделает это за вас. Например, воспользовавшись Pairwise Tool, мы получили 17 сценариев, способных охватить все 216 комбинаций. Вы можете увидеть список комбинаций

	Заказ	Размер	Город	Количество	Доставка	Время
1	Яблочный пирог	Большой	Чикаго	3	Адресная	Немедленно
2	Чизкейк	Большой	Нью-Йорк	2	Адресная	К указанному времени
3	Чизкейк	Маленький	Лос-Анджелес	1	Самовывоз	Немедленно
4	Чизкейк	Средний	Нью-Йорк	2	Адресная	К указанному времени
5	Чизкейк	Маленький	Лос-Анджелес	3	Самовывоз	Немедленно
6	Яблочный пирог	Большой	Лос-Анджелес	2	Самовывоз	Немедленно
7	Яблочный пирог	Маленький	Нью-Йорк	3	Адресная	К указанному времени
8	Яблочный пирог	Маленький	Чикаго	2	Самовывоз	К указанному времени
9	Яблочный пирог	Средний	Нью-Йорк	1	Адресная	Немедленно
10	Чизкейк	Средний	Чикаго	1	Самовывоз	К указанному времени
11	Чизкейк	Средний	Лос-Анджелес	3	Адресная	К указанному времени
12	Чизкейк	Большой	Нью-Йорк	1	Самовывоз	Немедленно
13	Яблочный пирог	Средний	Нью-Йорк	3	Самовывоз	Немедленно
14	Яблочный пирог	Маленький	Лос-Анджелес	1	Адресная	К указанному времени
15	Яблочный пирог	Средний	Нью-Йорк	2	Самовывоз	Немедленно
16	Яблочный пирог	Большой	Лос-Анджелес	1	Адресная	К указанному времени
17	Яблочный пирог	Маленький	Чикаго	2	Адресная	Немедленно

Простая проверка базовых действий и их результата. Например, если нажать крестик в правом верхнем углу окна (причина), оно закроется (следствие), и т.д. Этот метод позволяет проверить все возможности системы, а также обнаружить баги и улучшить техническую документацию продукта.

Примерный алгоритм использования техники:

1. Выделяем причины и следствия в спецификациях.
2. Связываем причины и следствия.
3. Учитываем «невозможные» сочетания причин и следствий.
4. Составляем «таблицу решений», где в каждом столбце указана комбинация входов и выходов, т.е. каждый столбец – это готовый тестовый сценарий.
5. Расставляем приоритеты.

ПРИЧИНА И СЛЕДСТВИЕ

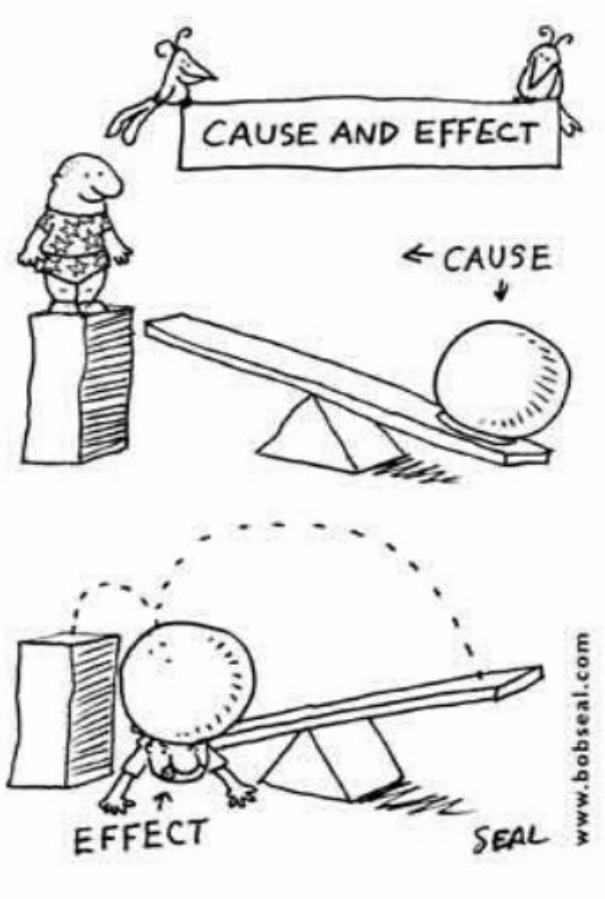
Эта техника помогает:

- Определить минимальное количество тестов для нахождения максимума ошибок.
- Выяснить все причины и следствия – таким образом, мы убедимся, что на любые манипуляции с системой у системы будет ответ.
- Найти возможные недочеты в логике описания приложения (что, в свою очередь, поможет улучшить документацию).

ПРИЧИНА И СЛЕДСТВИЕ

ПРИЧИНА И СЛЕДСТВИЕ. ПРИМЕР

Например, QA-специалист тестирует приложение типа “записная книжка”. После ввода всех данных нового контакта и нажатия кнопки Создать (причина) приложение должно автоматически создать карточку с номером телефона, фотографией и ФИО человека (следствие). Тесты покажут, можно ли оставлять одно или несколько полей пустыми, распознает ли система кириллицу, латиницу или оба алфавита, а также другие параметры.



Предугадывание ошибок обычно применяется вместе с другими техниками тест-дизайна. Суть этой техники в том, что тестировщик предугадывает, где могут появиться ошибки, опираясь на свой опыт, знание системы и требования к продукту. Таким образом он выявляет места, где могут накапливаться ошибки, и может уделить этим областям повышенное внимание.

ПРЕДУГАДЫВАНИЕ ОШИБОК

Пример предугадывания ошибок

Как правило, тестировщики начинают с тестирования на распространенные ошибки:

- ввод пробелов в текстовые поля
- нажатие кнопки Submit без ввода данных
- ввод неверных параметров (адрес электронной почты вместо номера телефона и т.д.)
- загрузка файлов, превышающих максимально допустимый размер
- … и так далее.

Чем больше опыта у тестировщика, тем больше сценариев предугадывания ошибок он сможет быстро придумать

ПРЕДУГАДЫВАНИЕ ОШИБОК. ПРИМЕР

Преимущества:

1. Эта проверка эффективна в качестве дополнения к другим техникам.
2. Выявляет тестовые случаи, которые “никогда не должны случиться”.

Недостатки:

1. Техника в значительной степени основана на интуиции.
2. Необходим опыт в тестировании подобных систем.
3. Малое покрытие тестами.

ПРЕДУГАДЫВАНИЕ ОШИБОК

ИТОГ

- Этот список далеко не полон и дает только самое общее представление о принципах тестирования и техниках тест-дизайна.
- Правильно подобранная техника тест-дизайна помогает разумно использовать ресурсы QA. Очень часто тестировщикам приходится комбинировать несколько техник тест-дизайна для обеспечения наиболее эффективного покрытия тестами. Правильная комбинация всегда зависит от конкретного проекта.

ТЕСТИРОВЩИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

КУРС «РУЧНОЕ ТЕСТИРОВАНИЕ»

8. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Изучение инструментов баг-трекинга

Оформление баг-репорта.

Разберем пример : использование техник тест-дизайна на поле «загрузка фото».

1. Первичный анализ:

- Для чего создавалась фича? Кто ее будет использовать? На каких окружениях?
- Какие форматы картинки нам необходимы? Размеры? Разрешение? Соотношение сторон важно? Цветность фотографии важна? Ширина / высота в пикселях?

2. Выберем параметры для проверки - выделим «эквивалентные классы», указанные в требованиях (допустим они есть)

- Форматы: jpeg, png, ico, tiff, jpg, pdf, pnm, bmp, gif, raw, heic
- Разрешение фотографии (количество точек): 320*240, 320*480, 640×480, 800×600, 1024×768, 1280×720, 1600×900, 1920*1080
- Соотношение сторон: 4:3, 5:3, 2:3, 5:4, 16:9, 8:5, 256:135
- Размер файла: 1 b, 1kb, 1mb, 10mb, 100 mb,
- Цветность фотографии: black and white, multi, specific, transparent (казалось бы причем здесь цвета)
- Стандарты размера фотографии (при печати, пожалуй очень спорный пункт но оставим для массовки): 9×13, 10×15, 15×20, 15×30, 20×30, 30×40

Возьмем за основу, что все остальное будет эквивалентный класс негативных тестов и будет приводить к одной и той же ошибке

3. Следующим шагом, стоит остановиться и подумать, а какие вещи/параметры я еще упустил. \

4. Совершенствуем тесты с помощью техники тест-дизайна (попробуем избавится от параметров, которые или не могут быть применены либо являются не граничными, либо уже эквивалентные относительно одного параметра.)

- Из форматов изображений сложно что-то убрать, уж слишком специфический у нас набор получился и любое исключение может привести к багу. А вот разрешение вполне подходит под один эквивалентный класс и взять можно граничные значения 320*240 и 1920*1080.

- Форматы: jpeg, png, ico, tiff, jpg, pdf, pnm, bmp, gif, raw, heic
- Разрешение фотографии (количество точек): 320*240, 320*480, 640×480, 800×600, 1024×768, 1280×720, 1600×900, 1920*1080
- Соотношение сторон: 4:3, 5:3, 2:3, 5:4, 16:9, 8:5, 256:135
- Размер файла: 1 b, 1kb, 1mb, 10mb, 100 mb,
- Цветность фотографии: black and white, multi, specific, transparent (казалось бы причем здесь цвета)
- Стандарты размера фотографии (при печати, пожалуй очень спорный пункт но оставим для массовки): 9×13, 10×15, 15×20, 15×30, 20×30, 30×40

- В соотношении сторон тоже все понятно, а файл если отработает самый маленький и самый большой, то будет все окей.
- Цветность фотографий в условиях переборки большинства форматов, тоже не волнует (хоть и вряд ли повлияет на работу загрузки и отображения).
- Стандартные размеры фотографии для печати не так важны, но могут не верно отображаться, а значит стоит протестировать. Пожалуй, стоит рискнуть и все же убрать несколько форматов в целях уменьшения тестов (при минимальном наборе можно было бы оставить — но в дальнейшем точно нужно будет избавится), предположим, что мы это выяснили у БА — избавимся от jpg (есть jpeg), pnm/bmp (не столь популярен) и ico (формат хранения файлов значков).

- Форматы: jpeg, png, ico, tiff, jpg, pdf, pnm, bmp, gif, raw, heic
- Разрешение фотографии (количество точек): 320*240, 320*480, 640×480, 800×600, 1024×768, 1280×720, 1600×900, 1920*1080
- Соотношение сторон: 4:3, 5:3, 2:3, 5:4, 16:9, 8:5, 256:135
- Размер файла: 1 b, 1kb, 1mb, 10mb, 100 mb,
- Цветность фотографии: black and white, multi, specific, transparent (казалось бы причем здесь цвета)
- Стандарты размера фотографии (при печати, пожалуй очень спорный пункт но оставим для массовки): 9×13, 10×15, 15×20, 15×30, 20×30, 30×40

ИСПОЛЬЗОВАНИЕ ТЕХНИК ТЕСТ-ДИЗАЙНА

Немного поработав во всех направлениях, как с техниками тест-дизайна, так и БА/DEV, мы получаем минимальный набор тестов:

Test-case	I	II	III	IV	V	VI	VII
Формат	jpeg	png	pdf	tiff	gif	raw	heic
Разрешение фотографии	320*240	any	1920*1080	800×600	any	any	1280×720
Соотношение сторон	2:3	5:3	16:9	5:4	8:5	256:135	any
Размер файла	1kb	1b	any	1mb	10mb	100mb	4,7mb
Цветность фотографи	black and white	transparent	any	transparent	multi	specific	any
Стандарты размера фотографии	10×15,	15×30	20×30	15×20	30×40	any	9×13

Который с какой-то высокой долей вероятности покроет большой процент возможных кейсов.

Там, где мы добавили *any* — это возможность для большей фантазии и использования параметров, не включенных в требования.

Таким образом, потратив совсем немного времени, наш маленький большой QA сможет проведя около 7 тестов — спать спокойно сегодня вечером.

Вместо выводов:

Как можно было заметить мы использовали еще одну интересную технику тест дизайна — мы непросто тестировали каждое значение в отдельности, мы применили комбинации значений и соединили их, чтобы снизить количество тестов до минимально достаточного набора. В конечном случае, если один из тестов даст ошибку, хорошему тестировщику нужно не просто зарепортить баг, нужно еще и отыскать какой именно из параметров дал сбой.

Для того чтобы провести более полноценное тестирование, не исчерпывающее ❗, необходимо сделать “немного” больше тестов с использованием комбинации различных параметров друг с другом.