

Техники тест-дизайна

Коновалова Елизавета

Повторение

Виды тестирования Тестовая модель

Сегодня

- Зачем нужны техники тест-дизайна
- Ориентация кейсов, тестовые данные
- Техники тест-дизайна
- Вспомогательные инструменты
- Эффективность тестов
- Мутационное тестирование
- Домашнее задание

Зачем нужны технические тест-

дизайна
001

Тест дизайн

Этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.

Техники тест дизайна – это приемы, которые позволяют создавать эффективные тест кейсы.

Зачем нужны техники тест-дизайна



**Задача:
проверить содержимое**

Задача: проверить содержимое



Гипотезы:

Телефон

Ничего

Мяч

Кот

Задача: проверить содержимое



Гипотезы:

Телефон

Ничего

Мяч

Кот

Тесты:

Открыть

Послушать

Потрясти

Просветить

Задача: проверить содержимое



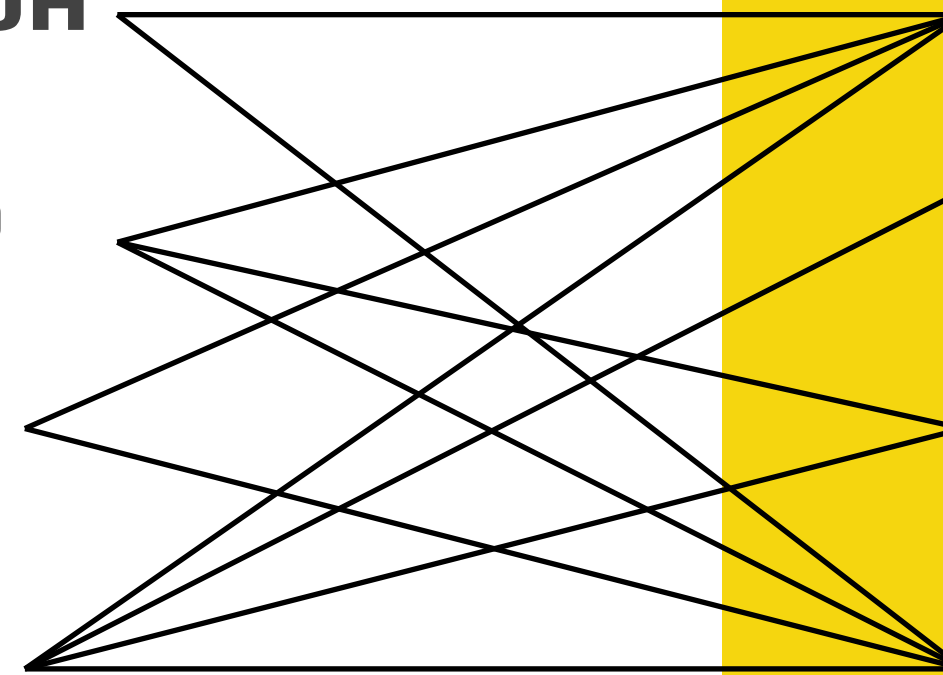
Гипотезы:

Телефон

Ничего

Мяч

Кот



Тесты:

Открыть

Послушать

Потрясти

Просветить

Задача: проверить содержимое



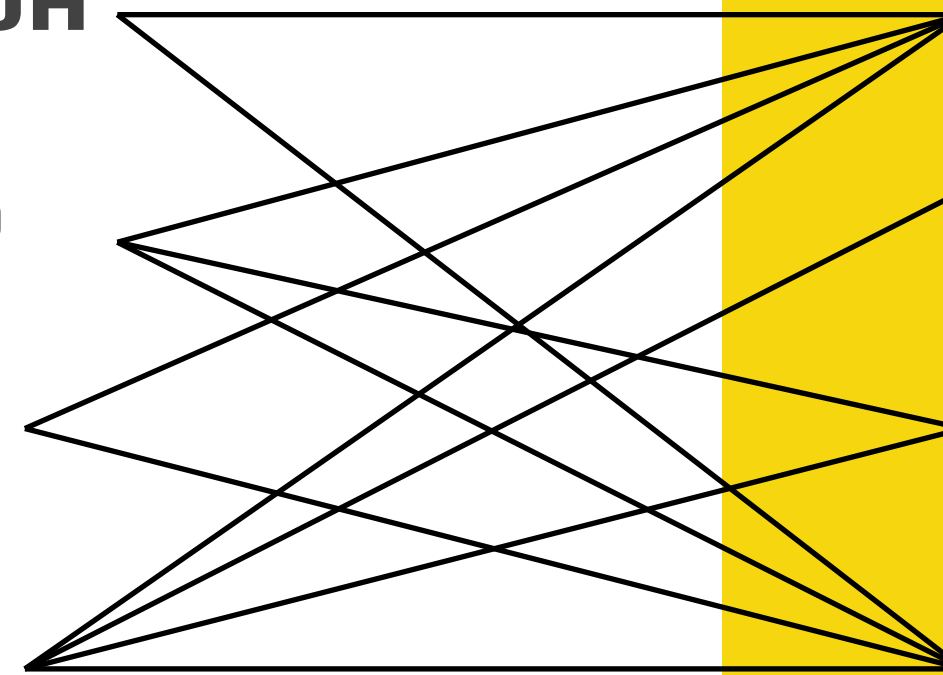
Гипотезы:

Телефон

Ничего

Мяч

Кот



Тесты:

Открыть

Послушать

Потрясти

Просветить

Задача: проверить содержимое



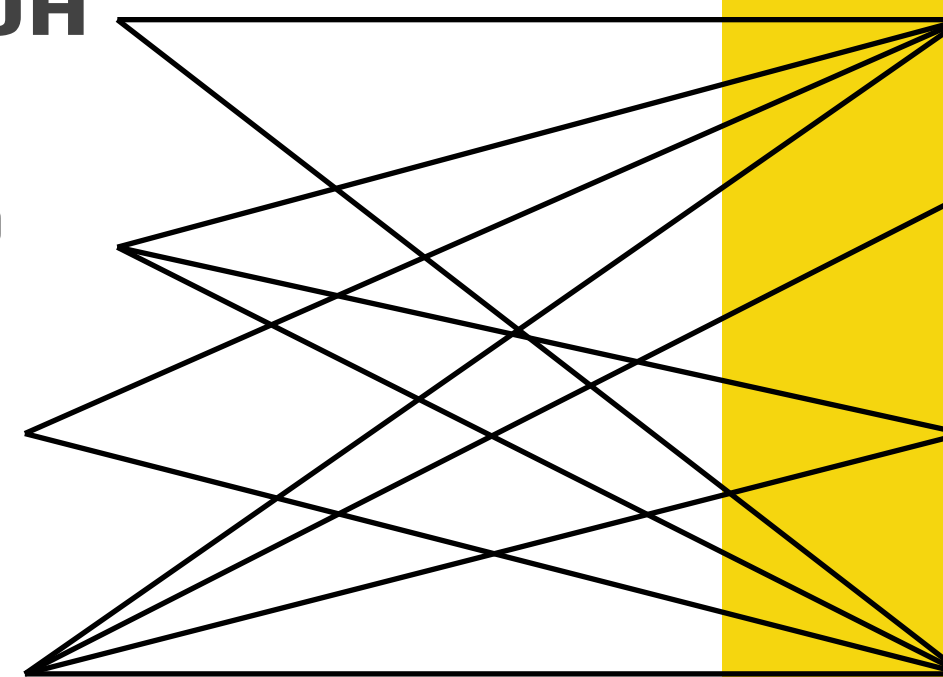
Гипотезы:

Телефон

Ничего

Мяч

Кот



Тесты:

Открыть

Послушать

Потрясти

Просветить

Зачем нужны техники тест-дизайна

- Чтобы не дублировать (не делать лишнее)
- Чтобы экономить время (ресурсы)
- Чтобы обеспечивать качество в разных условиях

Зачем нужны техники тест-

ирования



Исчерпывающее тестирование

Это процесс тестирования,
включающий все возможные проверки, состояния
системы и их сочетания.

Особенность: оно невозможно

Исчерпывающее тестирован



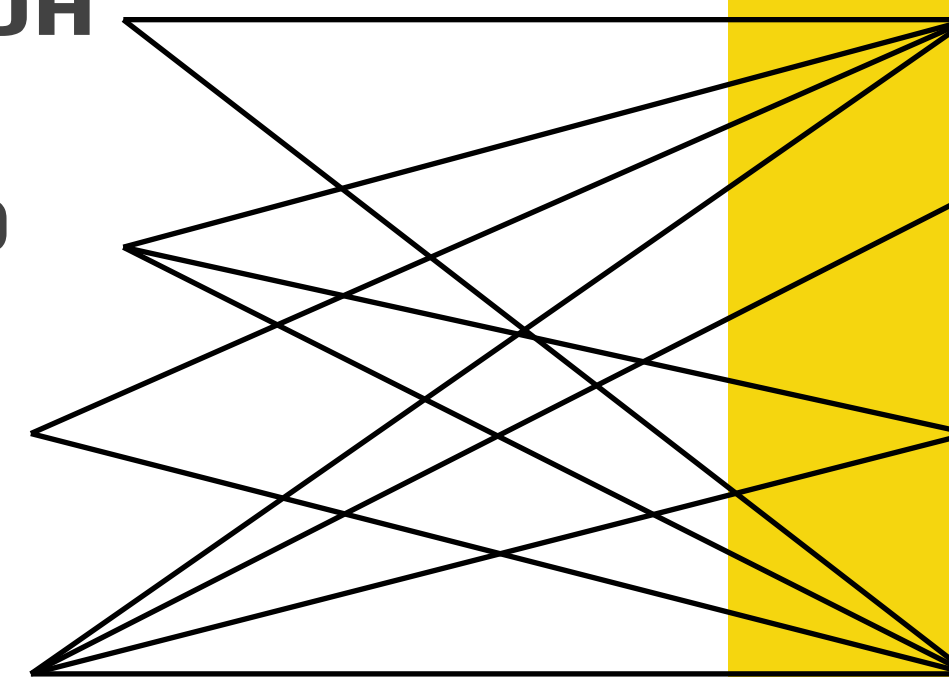
Гипотезы:

Телефон

Ничего

Мяч

Кот



Тесты:

Открыть

Послушать

Потрясти

Просветить

Запомнить!

Техники тест-дизайна:

- **Нужны, чтобы** создавать качественные тест-кейсы при минимальных затратах ресурсов
- Помогают проверить максимум функционала при минимальном количестве тест-кейсов (времени)
- Помогают избегать дублей, избыточности, однотипных проверок

Ориентация кейсов, тестовые

данные
002

Позитивные и негативные кейсы

Позитивные сценарии – сценарии, которые описывают нормальное (штатное, ожидаемое) поведение системы.

Негативные сценарии – сценарии, которые соответствуют внештатному поведению тестируемой системы.

Позитивные и негативные кейсы

meowle



Введите часть имени

🔍 Найти имя коту

Запомнить!

- Позитивные кейсы приоритетнее негативных
- Негативных кейсов больше и они чаще падают

Тестовые данные

Это данные, на которых проводятся тесты.

На разных данных одни и те же шаги дают разные результаты

Тестовые данные

meowle



Введите часть имени

🔍 Найти имя коту

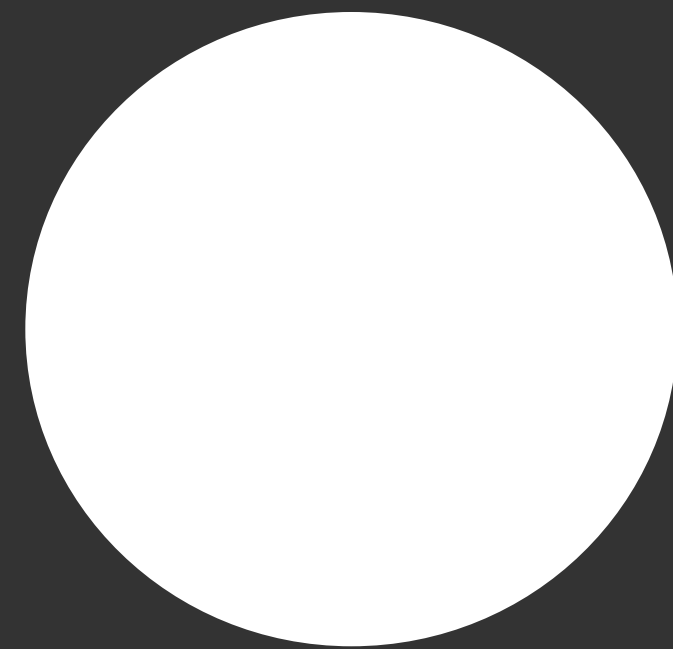
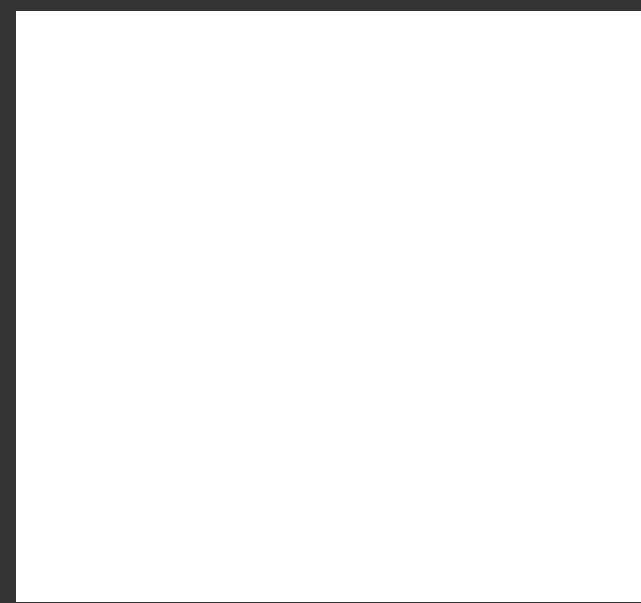
Запомнить!

- Тестовые данные не менее важны, чем шаги теста
- Разные наборы тестовых данных группируются в разные тест-кейсы
- Техники тест-дизайна применяются не только к тестам, но и к тестовым данным

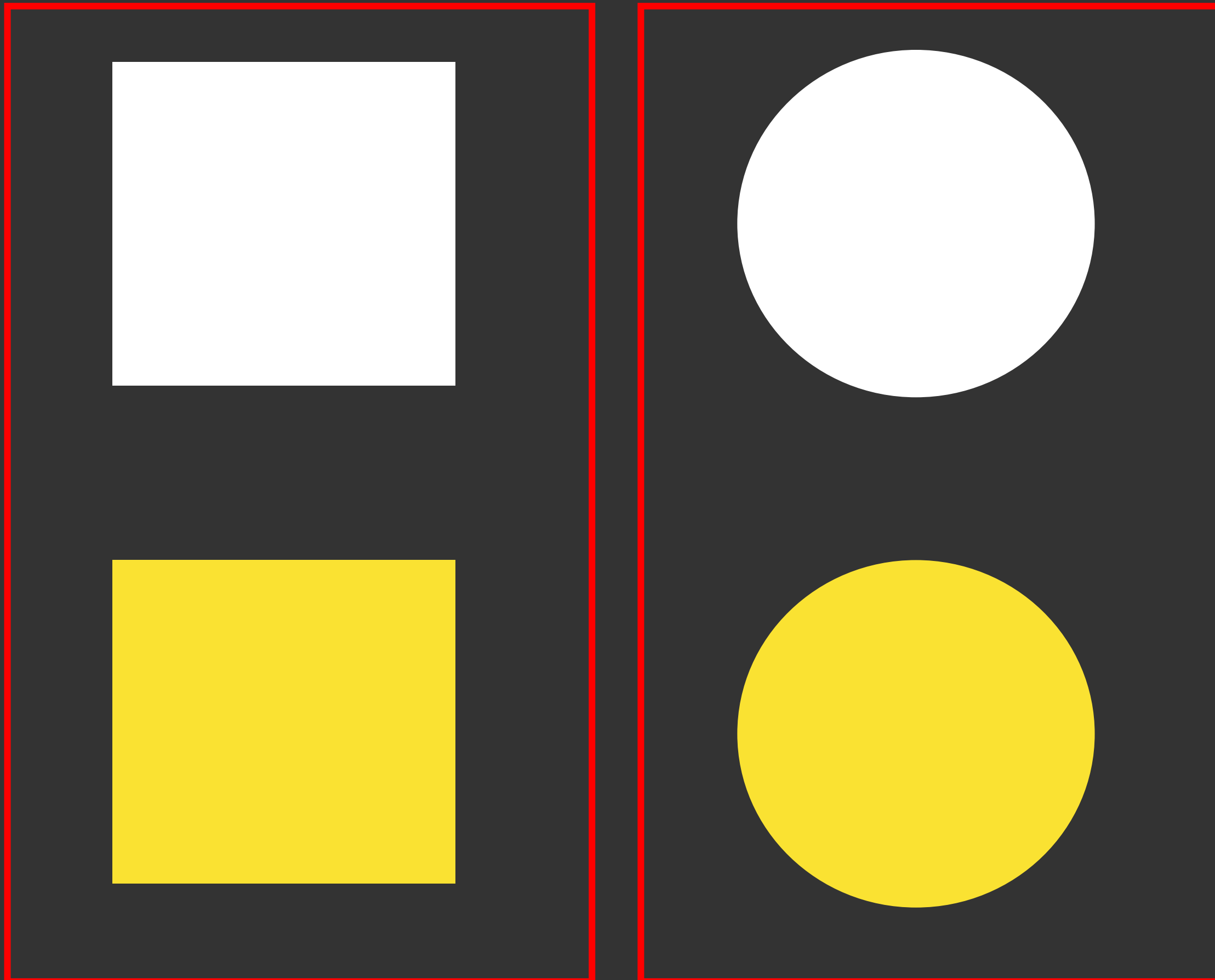
Классы эквивалентности и граничные значения

003

Классы эквивалентности



Классы эквивалентности



Классы эквивалентности



Классы эквивалентности

Два класса считаются эквивалентными, если в их основе лежит одна логика и наборы тестовых данных настолько похожи, что проверять их в полном объёме бессмысленно

Классы эквивалентности

meowle



Введите часть имени

🔍 Найти имя коту

Граничные значения

Это входное значение или данные, которое находится на грани эквивалентной области или на наименьшем расстоянии от обеих сторон грани, например, минимальное или максимальное значение области.

Граничные значения

Допустимые
значения

До 3 символов

От 3 до 50 символов

Больше 50 символов

Граничные
значения

Граничные значения

Алгоритм определения граничных значений:

1. Определить диапазон значений
2. Определить границы диапазона (чаще всего совпадают с границами классов эквивалентности)
3. Проверить следующие значения:
 - Минимальное
 - Максимальное
 - На 1 ниже границы
 - На 1 выше границы

Запомнить!

- Наиболее распространенные техники тест-дизайна – разбиение на классы эквивалентности и проверка граничных значений
- Эти техники можно применить почти к любому ПО, и к документации тоже

Причина - следствие

004

Причина-следствие

Причина	Следствие
То, что вызывает действие (каскад действий или изменений)	Результат действий

Причина-следствие

Причина	Следствие
То, что вызывает действие (каскад действий или изменений)	Результат действий
Ввести данные «Новый кот» Нажать кнопку «Добавить»	Добавился новый кот

Предугадывание ошибки

005

Предугадывание ошибки



Предугадывание ошибки

Это техника создания гипотез о проблемных местах системы на основании документации и знаний о системе.

Обычно с ее помощью создаются негативные кейсы

Предугадывание ошибки

Предугадать можно

- По опыту
- По документации
- По статистике (багов, тестов, использования)
- С помощью знаний о работе системы
- С помощью знаний о работе ПО

Предугадывание ошибки

meowle



Введите часть имени

🔍 Найти имя коту

Запомнить!

- Предугадывание ошибки работает тем эффективнее, чем больше и разнообразней опыт QA
- Для предугадывания ошибки можно пользоваться чужим опытом, например, статистикой

Попарное тестирование

006

Попарное тестирование

Метод парного тестирования основан на идее, что подавляющее большинство багов выявляется тестом, проверяющим один параметр, либо сочетание двух параметров.

Попарное тестирование

Все кейсы

Пол	Злость	Пушистость
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Достаточный набор

Пол	Злость	Пушистость
1	1	1
1	0	0
0	1	0
0	0	1

Ортогональная матрица

Попарное тестирование

Все кейсы

Пол	Злость	Пушистость
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Достаточный набор

Пол	Злость	Пушистость
1	1	1
1	0	0
0	1	0
0	0	1

Ортогональная матрица

Попарное тестирование

Все кейсы

Пол	Злость	Пушистость
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Достаточный набор

Пол	Злость	Пушистость
1	1	1
1	0	0
0	1	0
0	0	1

Ортогональная матрица

Попарное тестирование

Все кейсы

Пол	Злость	Пушистость
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Достаточный набор

Пол	Злость	Пушистость
1	1	1
1	0	0
0	1	0
0	0	1

Ортогональная матрица

Попарное тестирование

Все кейсы

Пол	Злость	Пушистость
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Достаточный набор

Пол	Злость	Пушистость
1	1	1
1	0	0
0	1	0
0	0	1

Ортогональная матрица

Запомнить!

- Многие придумано до и для вас, гуглите
- Пользуйтесь вспомогательными инструментами для создания тестов
- Будьте аккуратны с уровнем абстракции

Вспомогательны е инструменты

007

Майнд-карта

- Майнд-карта (*интеллектуальная карта*) – вид записи материалов в виде структуры, постепенно разветвляющийся на наиболее мелкие части.
- Ее цель: категоризировать, структурировать основную информацию.
- Особенность: не содержит никаких описаний

Майнд-карта



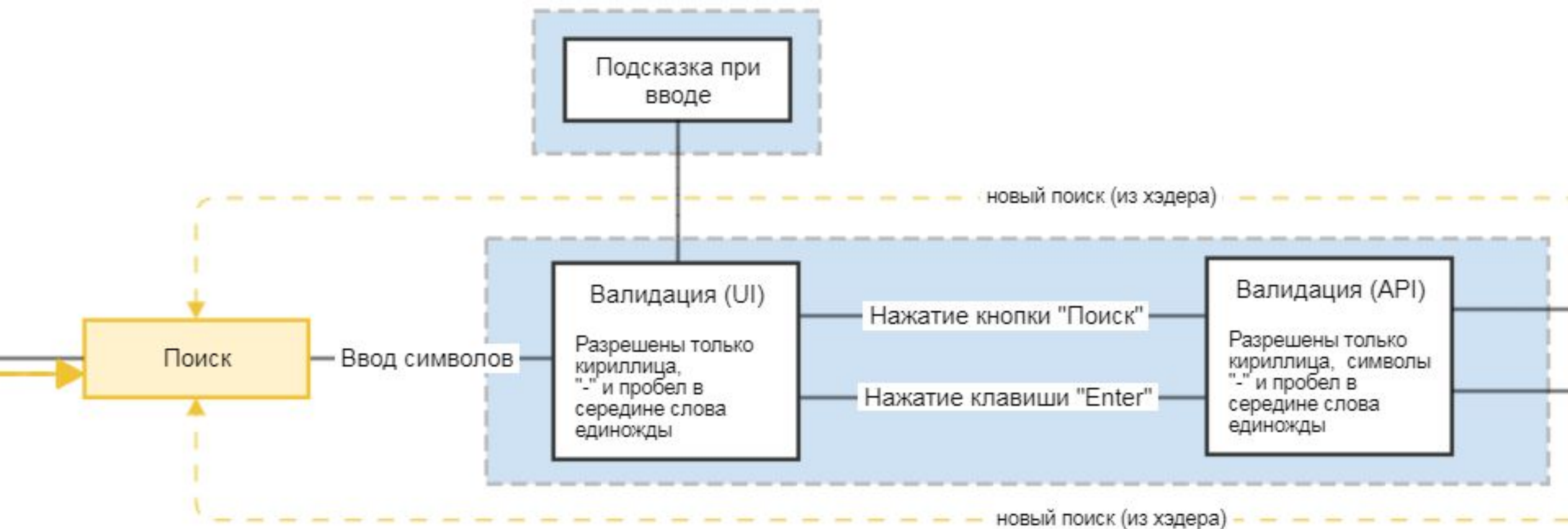
Майнд-карта

- Xmind
- MindMeister
- MindManager
- iMindMap
- Coggle
- ...

Блок-схема

- **Блок-схема – схема, описывающая алгоритмы или процессы.**
- **Ее цель: категоризировать, структурировать основную информацию.**
- **Особенность: не содержит никаких описаний**

Блок-схема



Блок-схема

- **Visio**
- **Dia**
- **draw.io**
- **Google Docs**
- **...**

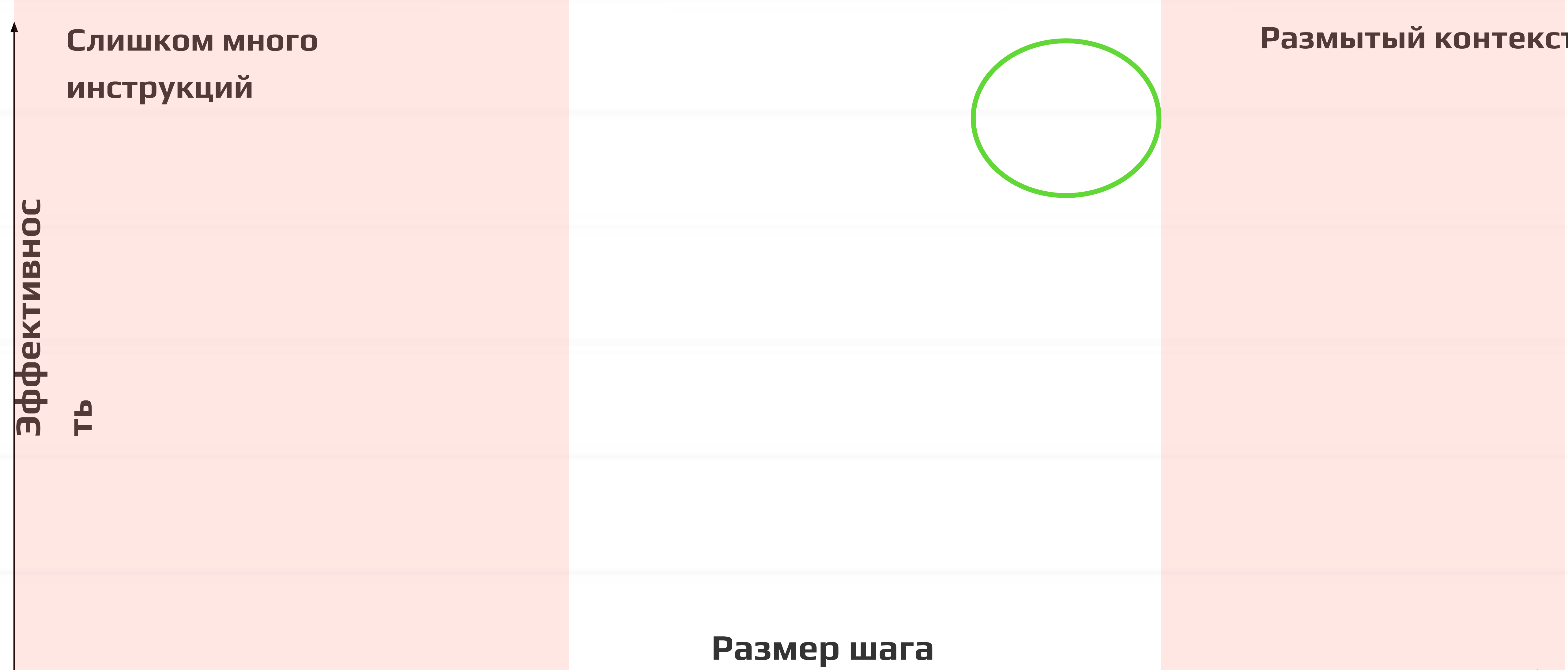
Запомнить!

- Чтобы создать оптимальную тестовую модель, можно использовать разные способы визуализации
- От способа представления информации напрямую зависит ее восприятие

Эффективность тест-кейсов

008

Эффективность тест-кейсов



Запомнить!

- Пользуйтесь вспомогательными инструментами и техниками
- Больше – не значит лучше
- Удаляйте всю бюрократию, которой команда не пользуется

Мутационное тестирование

009

Мутационное тестирование

Код



Passed

Мутационное тестирование

Код



Passed

Код



Passed

Мутационное тестирование

Код



Passed

Код



Passed

Мутант

Мутационное тестирование

Код



Passed

Код



Passed

Мутант



Passed

Мутационное тестирование

Код



Passed

Код



Passed

Мутант



Мутационное тестирование

Код



Passed

Код



Passed

Мутант



!

Мутационное тестирование

Код



Passed

Код



Passed

Мутант



! Skipped

Мутационное тестирование

Это метод тестирования программного обеспечения, который включает небольшие изменения кода программы.

Если набор тестов не в состоянии обнаружить такие изменения, то он рассматривается как недостаточный

Домашнее задание!

- Создать в <https://docs.google.com/spreadsheets/> целостную структуру тестовой модели своей фичи №1
- Наполнить ее эффективными тестами

Важная информация

- Командный зачет - соревнование багов, итоги – каждый календарный месяц
- Хороший баг содержит версию приложения! Можно узнать версию приложения вот тут <https://meowle.testops.ru/versions>

Спасибо!

Коновалова

Близарета
[@eakonovalova](#)