GeekBrains

Основы тест-аналитики

Классы эквивалентности

На этом уроке

1. Узнаем, что такое техники тест-дизайна и зачем они нужны
2. Рассмотрим классы эквивалентности
3. Узнаем, как работает pairwise (на примере нелинейных классов эквивалентности)
4. Составим тестовые наборы данных по принципу pairwise с использованием PICT
5. Рассмотрим pairwise и негативное тестирование

Оглавление

Что такое тест-дизайн и зачем он нужен

Тест-дизайн — это этап тестирования ПО, на котором проектируются и создаются тест-кейсы.

Допустим, есть задача: протестировать простейшую форму регистрации с 2 полями: логин и пароль. Сколько тестов нужно, чтобы установить уровень качества функциональности?

Интуитивно можно предположить:
- позитивный сценарий (логин test, пароль test1!)
- негативный сценарий (логин и пароль не заполнены)

Но что будет, если:
- такой логин уже занят?
- логин содержит спецсимволы */% и т.д.?
- логин с пробелами?
- пароль из 1 символа

А сколько еще внештатных ситуаций может быть? И как покрыть их все тест-кейсами?

Вспомним один из принципов тестирования: исчерпывающее тестирование невозможно. При этом надо протестировать достаточно хорошо. Поэтому есть правила составления тестов или подбора тестовых данных. Они вырабатывались на протяжении всего времени существования отрасли информационных технологий и сейчас представляют собой набор лучших практик, которых придерживается хороший тестировщик. Соблюдение этих правил гарантирует оптимальный охват тестируемой функциональности, при которой большая часть дефектов будет обнаружена и не дойдет до пользователя. Правила называются техниками тест-дизайна.

Самые распространенные техники:

- Классы эквивалентности (эквивалентное разделение).
- Граничные значения (анализ граничных значений, метод граничных значений).
- Попарное тестирование (тестовая комбинаторика, Pairwise).
- Тестирование состояний и переходов.
- Таблицы принятия решений.
- Исследовательское тестирование.

Применение техник тест-дизайна позволяет значительно сократить количество тестов не жертвуя качеством а также сконцентрироваться на наиболее уязвимых и важных участках функционала. На этом уроке мы разберемся с первыми тремя техниками.

Классы эквивалентности и граничные значения

Класс эквивалентности — набор входных значений, каждое из которых обрабатывается одинаково и приводит к одному результату. Значения внутри класса обладают одними признаками, что и приводит к идентичной обработке.

Для начала рассмотрим классы эквивалентности на отвлеченном примере: домашних животных. В качестве домашнего питомца можно завести кошку, собаку, попугайчика или рыбку. При этом все кошки имеют общие признаки:

- мурчат, когда их чешут за ушком
- любят охотиться
- делают тыгыдык по ночам

(конечно, мир природы более разнообразен, чем программное обеспечение, поэтому здесь мы делаем некоторое упрощение). Попугайчики имеют другие общие признаки:

- умеют летать
- могут научится разговаривать
- умеют петь

Таким образом, все кошки составляют один класс эквивалентности, а попугайчики - другой класс. Любая кошка будет мурчать, но при этом никогда не сможет летать. С другой стороны, любой попугайчик может петь, но не охотится по ночам. Допустим, перед нами стоит задача измерить уровень шума от разных домашних животных. Зная, что все кошки и все попугайчики производят примерно одинаковые звуки, нам не придется исследовать абсолютно всех кошек и абсолютно всех попугайчиков на планете. Достаточно взять одну любую кошку и одного любого попугайчика, измерить их уровень шума, и мы с большой долей вероятности получим корректные результаты.

> Итак, если известно, что есть группа данных, использование которых приводит систему в одно и то же состояние, то нет необходимости проверять каждое значение из этой группы отдельно. Исключения возможны, но мы не можем проверять все данные, так что приходится прибегать к подобным допущениям.

Классы эквивалентности бывают 2 видов:

- линейные (упорядоченные): значения можно упорядочить и расположить на шкале. У линейных классов эквивалентности всегда есть границы, где заканчивается 1 класс и

начинается другой. Границы могут входить внутрь класса, а могут быть самостоятельными единицами.
- нелинейные (неупорядоченные): значения нельзя упорядочить, граничных значений нет

Тестирование на основе классов эквивалентности — техника тест-дизайна на основе метода черного ящика, в ходе которой все данные делятся на классы эквивалентности, и затем тестирование проводится на одном значении из этого класса.

Тестирование на основе классов эквивалентности основано на двух умозаключениях:

1. Если одно значение из класса выявит ошибку — остальные, скорее всего, тоже это сделают.
2. Если одно значение из класса не выявит ошибку — остальные, скорее всего, тоже этого не сделают.

Исчерпывающее тестирование невозможно и есть вероятность, что отдельные значения в классе эквивалентности поведут себя не как все остальные. Поэтому в описаниях часто встречаются комментарии «скорее всего», «с большой долей вероятности».

На классы эквивалентности можно разбить:

- символы — они могут быть валидными, например использование @ в адресе электронной почты, и невалидными — ?, %,*;
- длину строки — например, валидный класс от 1 до 30 знаков, невалидный — всё остальное, то есть меньше 1 и больше 30;
- объём памяти, который необходим приложению для стабильной работы;
- разрешение экрана — всё, что меньше или больше заявленных требований к разрешению экрана, будет относиться к невалидным классам;
- версии операционных систем, библиотек — также определяются согласно требованиям. Например, приложение должно работать на ОС Windows 7, но поддержка Windows Nt уже не требуется.
- объём передаваемых данных — по требованиям. Например, если мощности сервера не позволяют обработать объём данных больше определённого значения.

Алгоритм

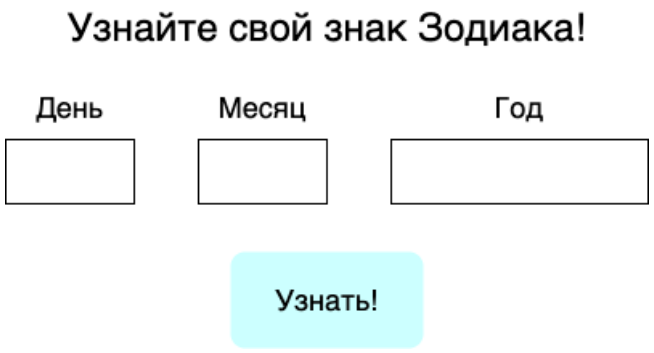
В общем алгоритм работы с классами эквивалентности выглядит следующим образом:

1. на основе анализа выбрать параметры, которые влияют на результат
2. для каждого параметра выделить классы эквивалентности
3. из каждого класса эквивалентности выбрать 1 значение
4. обработать выбранные значение в соответствии с pairwise

Рассмотрим два примера тестирования на основе классов эквивалентности.

Пример 1. Знаки Зодиака

User Story: Я как пользователь хочу определить свой знак Зодиака по дате рождения.



Это макет будущего сайта

Пользовательский сценарий:

Пользователь: Заходит на страницу
Пользователь: Заполняет день, месяц и год рождения
Пользователь: Нажимает кнопку “Узнать”

Система: Проверяет, что значение поля “Месяц” от 1 до 12
Система: Проверяет значение поля “День”
Система: Проверяет значение поля “Год” - високосный или нет

- от 1 до 28, если Месяц = 2 и год не високосный
- от 1 до 29, если Месяц = 2 и год високосный
- от 1 до 30, если Месяц = 4, 6, 9, 11
- от 1 до 31, если Месяц = 1, 3, 5, 7, 8, 10, 12

Дата корректная
Система: Показывает знак Зодиака, соответствующий дате.

Дата некорректная
Система: Показывает сообщение “Указанная дата не существует”

Естественно, мы не будем тратить время и проверять каждый день в году, а воспользуемся техникой классов эквивалентности. Сначала рассмотрим позитивный сценарий, когда пользователь ввел существующую, валидную дату, например, 31.03.2000. Какие классы эквивалентности можно выделить?

Очевидно, поскольку знаков Зодиака 12, то и классов эквивалентности будет столько же, по одному на каждый знак. Присвоим каждому дню года порядковый номер, где 1 - это 1 января, а 365 - это 31 декабря (рассмотрим невисокосный год). Получим следующие классы:

1. 1 - 20 - Козерог

2. 21 - 50 - Водолей
3. 51 - 79 - Рыбы
4. 80 - 110 - Овен
5. 111 - 141 - Телец
6. 142 - 172 - Близнецы
7. 173 - 203 - Рак
8. 204 - 233 - Лев
9. 234 - 266 - Дева
10. 267 - 296 - Весы
11. 297 - 326 - Скорпион
12. 327 - 356 - Стрелец
13. 357 - 365 - Козерог

Из-за того, что гороскоп не совпадает с календарным годом, получилось не 12, а 13 классов. Теперь для тестирования достаточно взять по 1 значению из каждого интервала и проверить его. Например, если мы укажем 30 день года (т.е. 30 января), то результат будет “Водолей”. Итого, для минимального тестового покрытия нам понадобится 13 тестов.

Рассмотрим альтернативный сценарий, когда пользователь ввел невалидную дату. Здесь возможно несколько вариантов:

- 29.02.2001 (год невисокосный, 29 февраля в нем нет)
- 32.03.2000 (32 числа нет ни в одном месяце)
- 31.04.1999 (31 число бывает, но не в апреле)
- 25.14.1995 (нет 14 месяца)
- 00.00.0000 (если речь идет о цифрах, нули проверяем с особым пристрастием)

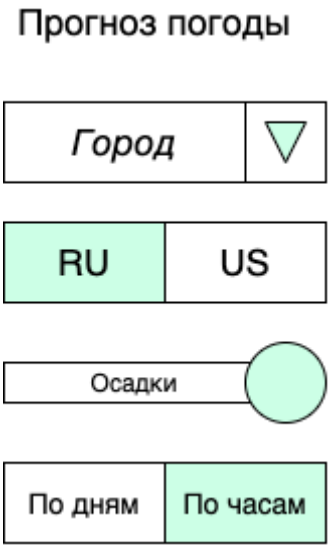
Можно проявить воображение и придумать еще больше негативных проверок, здесь полет фантазии ничем не ограничен.

Итак, для тестирования простейшей странички нам понадобится минимум 18 тестов.

Так как данные классы эквивалентности являются линейными, то у них есть границы. Это элементы повышенного риска, и их тестирование мы рассмотрим в теме “Доменный анализ”. Это еще немного увеличит наш список необходимых проверок.

Пример 2. Прогноз погоды

User story: Я как пользователь хочу узнать прогноз погоды в указанном городе с нужной степенью детализации



Это макет будущего сайта

Пользовательский сценарий:

Пользователь: заходит на сайт
Пользователь: выбирает город из списка
Пользователь: выбирает язык (по умолчанию выбран русский)
Пользователь: выбирает детализацию по осадкам (по умолчанию выключено)
Пользователь: выбирает детализацию по времени (по умолчанию по дням)

Система: показывает пользователю прогноз в соответствии с указанными значениями

Также известно, что используется 3 алгоритма расчета прогноза:

1. для городов федерального значения (Москва, Санкт-Петербург, Севастополь)
2. для городов областного, республиканского, краевого, окружного значения (Волгоград, Казань, Краснодар и т.д.)
3. для городов районного значения (Бологое, Суоярви т. д.)

Выделим классы эквивалентности для всех параметров

Город	**Язык**	**Осадки**	**Детализация**
Федерального значения	Русский	Да	По дням
Областного значения	Английский	Нет	По часам
Районного значения			

Сколько нужно тестов для прогноза погоды? Если протестировать все возможные варианты комбинаций параметров, то получится 3 * 2 * 2 * 2 = 24 теста, что довольно много, и бессмысленно, т.к. количество тестов можно сократить техникой pairwise, или попарным тестированием.

Как работает pairwise

Большинство дефектов возникают при комбинации **только двух** параметров. Например, нет смысла проверять все возможные сочетания параметров. Если ошибка возникнет в сочетании “Москва” + “Русский”, то вероятнее всего, она будет возникать и в различных значениях детализации по осадкам, и со всем детализациями по времени. Такой подход позволяет сократить количество проверок.

Рассмотрим pairwise на практике. Сначала создадим таблицу из 24 строк со всеми возможными комбинациями.

	Город	**Язык**	**Осадки**	**Детализация**
1	Москва	Русский	Да	По дням
2	Москва	Русский	Да	По часам
3	Москва	Русский	Нет	По дням
4	Москва	Русский	Нет	По часам
5	Москва	Английский	Да	По дням
6	Москва	Английский	Да	По часам
7	Москва	Английский	Нет	По дням
8	Москва	Английский	Нет	По часам
9	Волгоград	Русский	Да	По дням
10	Волгоград	Русский	Да	По часам
11	Волгоград	Русский	Нет	По дням
12	Волгоград	Русский	Нет	По часам
13	Волгоград	Английский	Да	По дням
14	Волгоград	Английский	Да	По часам
15	Волгоград	Английский	Нет	По дням
16	Волгоград	Английский	Нет	По часам
17	Бологое	Русский	Да	По дням
18	Бологое	Русский	Да	По часам

19	Бологое	Русский	Нет	По дням
20	Бологое	Русский	Нет	По часам
21	Бологое	Английский	Да	По дням
22	Бологое	Английский	Да	По часам
23	Бологое	Английский	Нет	По дням
24	Бологое	Английский	Нет	По часам

А теперь попробуем создать таблицу таким образом, чтобы перебрать все возможные комбинации каждой пары параметров. Начнем с пары **“Город - Язык”**, а остальные ячейки пока что заполнять не будем

	Город	**Язык**	**Осадки**	**Детализация**
1	Москва	Русский		
2	Москва	Английский		
3	Волгоград	Русский		
4	Волгоград	Английский		
5	Бологое	Русский		
6	Бологое	Английский		

Теперь заполним колонку “Осадки”, чтобы перебрать все возможные пары **“Город - Осадки”**

	Город	**Язык**	**Осадки**	**Детализация**
1	Москва	Русский	Да	
2	Москва	Английский	Нет	
3	Волгоград	Русский	Да	
4	Волгоград	Английский	Нет	
5	Бологое	Русский	Да	
6	Бологое	Английский	Нет	

Далее проверим пару “Язык - Осадки”. Видим, что у нас есть пары “Русский - Да”, “Английский - Нет”, но не хватает пары “Русский - Нет”, “Английский - Да”. Это легко исправить: просто поменяем местами значения осадков в строке 1 и 2. Теперь у нас есть все возможные варианты “Язык - Осадки”.

	Город	Язык	Осадки	Детализация
1	Москва	Русский	**Нет**	
2	Москва	Английский	**Да**	
3	Волгоград	Русский	Да	
4	Волгоград	Английский	Нет	
5	Бологое	Русский	Да	
6	Бологое	Английский	Нет	

Аналогично заполняем колонку “Детализация”. Сначала сопоставляем ее с городом, а затем проверяем, чтобы были все пары “Язык - Детализация”, “Осадки - Детализация”, и при необходимости меняем значения в строках местами (но следим, чтобы не разрушались нужные пары).

	Город	Язык	Осадки	Детализация
1	Москва	Русский	Нет	По дням
2	Москва	Английский	Да	По часам
3	Волгоград	Русский	Да	По часам
4	Волгоград	Английский	Нет	По дням
5	Бологое	Русский	Да	По дням
6	Бологое	Английский	Нет	По часам

Готово! Вместо 24 тестов осталось всего 6, и это в 4 раза сократит время на тестирование.

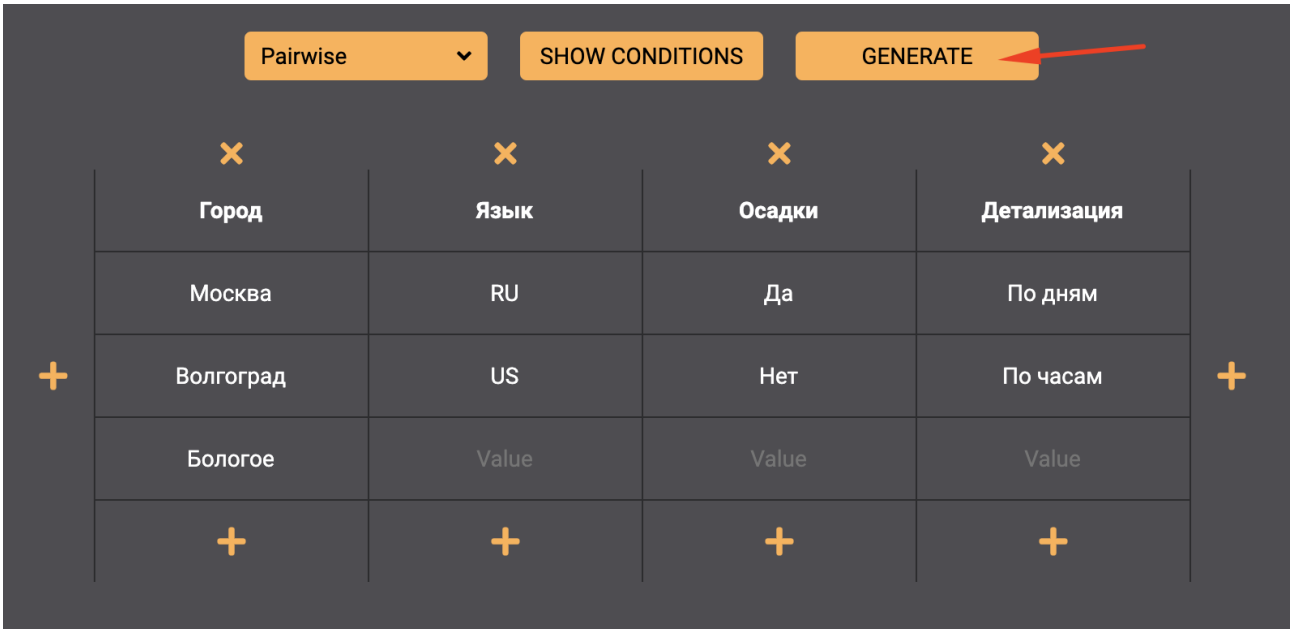
Pairwise используется только для неупорядоченных классов эквивалентности. Для упорядоченных классов эта техника тест-дизайна может дать слишком много проверок на выходе, т.к. необходимо учитывать значения внутри класса и на границах. Особенности тестирования линейных классов эквивалентности рассматривается на уроке “Доменный анализ”.

PICT и другие инструменты

Составлять таблицы попарного тестирования вручную - долго, есть риск ошибок. На радость нам существуют специальные программы, которые сами составляют таблицы из исходных параметров:

- https://slothman.dev/pairwise-generator/
- https://pairwise.yuuniworks.com/

Рассмотрим составление таблицы в slothman. Создадим колонки с нужными значениями и нажмём Generate



В итоге получится файл, похожий на то, что мы сделали сами

	A	B	C	D
1	**Город**	**Язык**	**Осадки**	**Детализация**
2	Волгоград	RU	Да	По дням
3	Бологое	RU	Нет	По часам
4	Волгоград	US	Да	По часам
5	Москва	US	Нет	По дням
6	Бологое	US	Да	По дням
7	Волгоград	US	Нет	По часам
8	Москва	RU	Да	По часам

Обратите внимание, что программа сгенерировала 7 строк, а не 6, как это получилось при создании вручную. Почему так получилось? Дело в том, что в идеале матрица должна быть сбалансирована. Это значит, что если какая-то пара значений двух столбцов встречается несколько раз, то все возможные парные комбинации значений этих столбцов должны встретится столько же раз. Матрица, составленная вручную, не сбалансирована, т.к. пара “Английский - Нет” встречается 2 раза, а “Английский - Да” - 1 раз.

Pairwise и негативное тестирование

Составляя тестовые наборы для попарного тестирования, никогда не учитываются негативные сценарии, т.е. те, которые способны привести к ошибке. В случае с прогнозом погоды - это оставить

пустым значение поля “Город”. Нужно ли добавлять еще одну строку “Поле незаполнено” в генератор pairwise? НЕТ. Невалидные значения никогда не комбинируются друг с другом, т.к. если в тесте будет 2 или более невалидных параметра, то тестировщик не установит причину возникновения ошибки.

Проверки с невалидными данными добавляются в таблицу по принципу “1 невалидный параметр в строке”. Остальные параметры могут быть любыми валидными.

	Город	**Язык**	**Осадки**	**Детализация**
1	Москва	Русский	Нет	По дням
2	Москва	Английский	Да	По часам
3	Волгоград	Русский	Да	По часам
4	Волгоград	Английский	Нет	По дням
5	Бологое	Русский	Да	По дням
6	Бологое	Английский	Нет	По часам
7	Не заполнено	Английский	Нет	По часам
8	Бологое	Китайский	Да	Нет

Контрольные вопросы

1. Какие бывают классы эквивалентности?
2. Приведите примеры, с какими классами эквивалентности вы сталкиваетесь в жизни
3. Что можно разбить на классы эквивалентности?

Глоссарий

Тест-дизайн — это этап тестирования ПО, на котором проектируются и создаются тест-кейсы.

Класс эквивалентности — набор входных значений, каждое из которых обрабатывается одинаково и приводит к одному результату.

Попарное тестирование (pairwise) - техника подбора тестовых данных, основанная на утверждении, что большинство дефектов возникают при комбинации только двух параметров

Домашнее задание

1. Изучите требования к разделу “Проекты”.
2. Выделите классы эквивалентности для создания нового проекта. Обратите внимание на вкладки “Общие настройки” и “Редактор атрибутов”
3. Для неупорядоченных классов определите все допустимые значения. Если возможно, дополните каждый класс недопустимыми значениями.
4. Создайте копию документа. Сформируйте наборы данных из допустимых значений по принципу pairwise на вкладке “Таблица pairwise”.
5. Выделите упорядоченные классы эквивалентности. Добавьте в таблицу соответствующие колонки. Заполните их значениями, которые приведут к положительному результату.
6. Дополните полученную таблицу строками с недопустимыми значениями (если таковые найдутся).
7. Дополните тест-кейсы на вкладке “Тест-кейсы для pairwise” недостающими шагами и допишите ожидаемый результат.
8. Дополните тест-кейсы на вкладке “Тест-кейсы для pairwise” полученными данными.

Дополнительные материалы

1. Статья Pairwise testing. Part 1 - Orthogonal Arrays
2. Статья Метод попарного тестирования
3. Статья Классы эквивалентности для имен
4. Статья Немного о простом. Тест-дизайн