ТЕХНИКА АНАЛИЗА КЛАССОВ ЭКВИВАЛЕНТНОСТИ

Класс эквивалентности (equivalence class) — одно или несколько значений ввода, к которым программное обеспечение применяет одинаковую логику.

Техника анализа классов эквивалентности

это техника, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений. Такое разделение помогает убедиться в правильном функционировании целой системы — одного класса эквивалентности, проверив только один элемент этой группы. Эта техника заключается в разбиении всего набора тестов на классы эквивалентности с последующим сокращением числа тестов.

Техника анализа классов эквивалентности

Техника рекомендует проведение тестов для всех классов эквивалентности, хотя бы по одному тесту для каждого класса. Техника анализа классов эквивалентности стремится *не только сокращать количество тестов*, но и *сохранять приемлемое тестовое покрытие*.

Признаки эквивалентности тестов:

- направлены на поиск одной и той же ошибки;
- если один из тестов обнаруживает ошибку, другие скорее всего, тоже её обнаружат;
- если один из тестов не обнаруживает ошибку, другие, скорее всего, тоже её не обнаружат;
- тесты используют схожие наборы входных данных;
- для выполнения тестов мы совершаем одни и те же операции;
- тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние;
- все тесты приводят к срабатыванию одного и того же блока обработки ошибок;
- ни один из тестов не приводит к срабатыванию блока обработки ошибок.

Техника анализа классов эквивалентности алгоритм использования:

- *Определить классы эквивалентности.*

Это главный шаг техники, т.к. во многом от него зависит эффективность её применения.

2. *Выбрать одного представителя от каждого класса эквивалентности.*

На этом этапе следует выбрать один тест из эквивалентного набора тестов.

3. *Выполнение тестов.*

На этом шаге следует выполнить тесты от каждого класса эквивалентности.

Если есть время, можно протестировать еще несколько представителей от каждого класса эквивалентности. Следует иметь ввиду, при правильном определение классов эквивалентности дополнительные тесты скорее всего будут избыточными и дадут такой же результат.

Техника анализа классов эквивалентности классический пример:
Есть поле ввода с диапазоном допустимых значений от 1 до 100.

Сами понимаете, что на 95 тестов на допустимые значения и на несметное количество тестов на недопустимые значения уйдет очень много времени. И здесь нам помогут классы эквивалентности.

Исходя из того, с одной стороны, все допустимые значения могут влиять на поле ввода одинаково, следовательно все числа от 1 до 100 можно смело считать

эквивалентными. С другой стороны, все недопустимые значения должны одинаково влиять на поле ввода (в идеале не должно быть возможности ввода этих значений в поле). Таким образом, есть уже несколько классов эквивалентности:

1. *допустимые значения (от 1 до 100);*

недопустимые значения:

1. *от — ∞ до 0;*
2. *от 101 до + ∞;*
3. *специальные символы (# @ + — / _ : ; “ ‘ и т.д.);*
4. *буквы.*

Используя классы эквивалентности можно протестировать поле ввода минимум из 5 тестов.

На практике классы эквивалентности обязательны при тестировании всевозможных форм и полей ввода.

Плюсы и минусы техники анализа эквивалентных классов

К плюсам можно отнести отсеивание огромного количества значений ввода, использование которых просто бессмысленно.

К минусам можно отнести неправильное использование техники, из-за которого есть риск упустить баги.

Техника анализа классов эквивалентности пример из реального проекта:
проект — Btrack.com

протестировать поля ввода — в Manage->Users Add user поля Имя/Фамилия

Условие: в поля ввода можно внести русские/английские буквы; заглавные и строчные буквы; т.к. имена и фамилии могут быть составными наличие символа “-” (дефис)

Исходя из условий, разбиение на классы эквивалентности происходит следующим образом:

Допустимые значения (блок позитивных тестов):

1. *все буквы русского (от А до Я)/английского (A-Z);*
2. *заглавные и строчные буквы;*
3. *допускается написание имени/фамилии через дефис.*

Недопустимые значения (блок негативных тестов):

1. *цифры (от -∞ до +∞);*
2. *специальные символы, отличные от дефиса и угловых скобок (< >).*

Используя данные классы можно протестировать поля ввода с помощью 5 тестов.

Из блока допустимых значений:

1. *буква П/F;*
2. *р и P/g и G$*
3. *Салтыков-Щедрин.*

Из блока недопустимых значений:

1. *+1026;*
2. *%$#/ и < >.*

Подытожим — Техника анализа классов эквивалентности одна из нескольких часто применяемых техник при планировании и разработке тестов. Значительно сокращает количество тестов необходимых для проверки функционала и время, с другой стороны в не опытных руках может стать инструментом который не только не поможет найти дефекты, но и сложит ошибочное представление о покрытие приложения тестами.

ТЕХНИКА АНАЛИЗА ГРАНИЧНЫХ ЗНАЧЕНИЙ

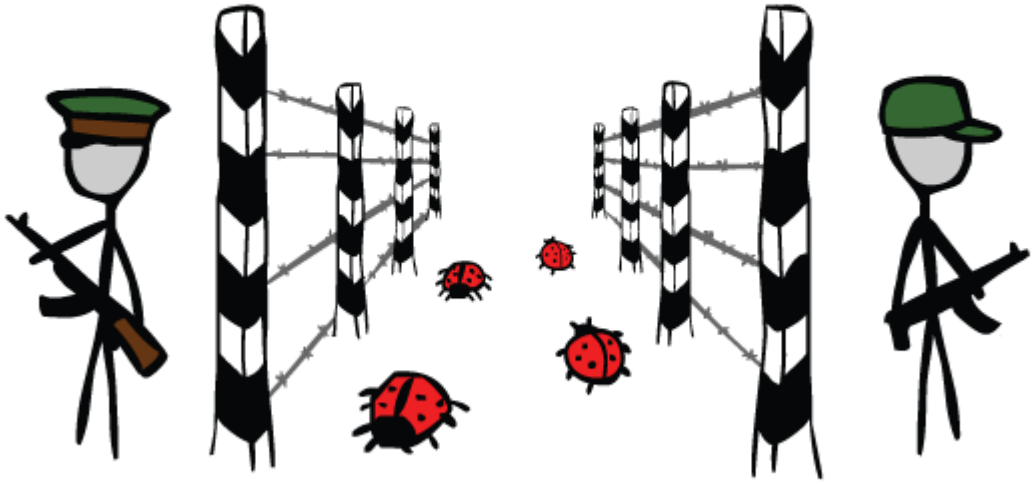
Граничные значения — это те места, в которых один класс эквивалентности переходит в другой.

Техника анализа граничных значений

Это техника проверки поведения продукта на крайних (граничных) значениях входных данных. Граничное тестирование также может включать тесты, проверяющие поведение системы на входных данных, выходящих за допустимый диапазон значений. При этом система должна определённым (заранее оговоренным) способом обрабатывать такие ситуации. Например, с помощью исключительной ситуации или сообщения об ошибке.

!!!Граничные значения очень важны и их обязательно следует применять при написании тестов, т.к. именно в этом месте чаще всего и обнаруживаются ошибки.

Баги водятся на границах



Техника анализа граничных значений

На каждой границе диапазона следует проверить по три значения:

1. граничное значение;
2. значение перед границей;
3. значение после границы.

Цель этой техники — найти ошибки, связанные с граничными значениями.

Алгоритм использования техники граничных значений:

- *выделить классы эквивалентности*;

Как и в предыдущей технике, этот шаг является очень важным и от того, насколько правильным будет разбиение на классы эквивалентности, зависит эффективность тестов граничных значений.

- *определить граничные значения* этих классов;
- нужно *понять, к какому классу будет относиться каждая граница*;
- нужно *провести тесты по проверке значения до границы, на границе и сразу после границы*.

Количество тестов для проверки граничных значений будет равен количеству границ, умноженному на 3. Рекомендуется проверять значения *вплотную к границе*. К примеру, есть диапазон целых чисел, граница находится в числе 100. Таким образом, будем проводить тесты с числом 99 (до границы), 100 (сама граница), 101 (после границы).

Техника анализа граничных значений. Пример использования на существующем проекте:

Проект — Btrack.com

Протестировать поля ввода — Dev-Est/Qa-est

Условие — в поля ввода можно внести только целые числа от 0 до 10 000.

Определяемся с существующими границами — так как в условии все значения от 0 до 10 000 приведут к одному и тому же результату, то границы две: нижняя и верхняя.
Первое граничное значение — 0
Второе граничное значение — 10 000
Добавляем к ним, стоящие рядом значения (опять же, если бы, числа были дробными, то пришлось бы, для начала, определится с количеством знаков после запятой. Например, если десятые, то значения были бы: -0,1 0 0,1 и 9999,9 10000 10000,1):

1. *-1, 0, 1*
2. *9 999, 10 000, 10 001*

Данная Техника анализа граничных значений хорошо и очень хорошо сочетается в работе:

Таблица принятия решений

(таблица решений) — способ компактного представления модели со сложной логикой; инструмент для упорядочения сложных бизнес требований, которые должны быть реализованы в продукте. Это взаимосвязь между множеством условий и действий.
В таблицах решений представлен набор условий, одновременное выполнение которых должно привести к определённому действию.
Таблица принятия решений

Таблица принятия решений, как правило, разделяется на 4 квадранта:

Условия	Варианты выполнения действий
Действия	Необходимость действий

Условия — список возможных условий.
Варианты выполнения действий — комбинация из выполнения и/или невыполнения условий этого списка.
Действия — список возможных действий.
Необходимость действий — указание надо или не надо выполнять соответствующее действие для каждой из комбинаций условий.

Рассмотрим *таблицу принятия решений* на примере страницы регистрации нового пользователя сервиса KUKU.io
Используем понятия “корректные” и “некорректные” данные.
Чтобы регистрация прошла успешно, необходимо заполнить корректными оба поля. Если поля заполняются некорректными данными, то система должна выдать ошибку: “Введены невалидные данные”.

Условие	Значения 1	Значения 2	Значения 3	Значения 4
Ввод корректных данных в поле E-mail	+	—	+	—
Ввод корректных данных в поле Password	+	—	—	+
Ввод некорректных данных в поле E-mail	—	+	—	+
Ввод некорректных данных в поле Password	—	+	+	—
Действия				
Регистрация прошла успешно	+	—	—	—
Выдается ошибка: “Введены невалидные данные”	—	+	+	+

Значения 2, 3, 4 приводят к одному и тому же результату с разными входными значениями

Скаля́рная величина́ (от лат. *scalaris* «ступенчатый») в физике — величина, каждое значение которой может быть выражено одним (как правило, действительным) числом[1]. То есть скалярная величина определяется только значением, в отличие от вектора, который, кроме значения, имеет направление. К скалярным величинам относятся длина, площадь, время, температура, электрический заряд, работа, статистический вес

Транзитивность — свойство бинарного отношения. Бинарное отношение R на множестве X называется транзитивным, если для любых трёх элементов множества a,b,c выполнение отношений aRb и bRcbRc влечёт выполнение отношения aRc (запись aRb означает отношение a к b, bRc —b к c, aRc —a к c.

Те́нзор (от лат. tensus, «напряжённый») — применяемый в математике и физике вид линейного многокомпонентного алгебраического объекта (объекта линейной алгебры), заданного на векторном пространстве {\displaystyle V}V конечной размерности {\displaystyle n}n. В физике в качестве {\displaystyle V}V обычно выступает физическое трёхмерное пространство или четырёхмерное пространство-время, а компонентами тензора являются координаты (проекции) взаимосвязанных физических величин. Использование тензоров в физике позволяет глубже понять физические законы и уравнения, упростить их запись (за счет сведения многих связанных физических величин в один тензор), а также записывать уравнения в так называемой общековариантной форме, не зависящей от выбранной системы отсчета.

Отношение эквивалентности — бинарное **отношение** между элементами данного множества, свойства которого сходны со свойствами **отношения** равенства.

Фактормножество — множество всех классов эквивалентности для заданного отношения эквивалентности ~ на множестве X, обозначается X/~. Разбиение множества на классы эквивалентных элементов называется его факторизацией.

Бина́рное (**двухме́стное**) **отноше́ние** (соответствие[1][2]) — отношение между двумя множествами и , то есть всякое подмножество декартова произведения этих множеств:

Свойства отношений [править | править код]

Бинарное отношение R на некотором множестве M может обладать различными свойствами, например:

- рефлексивность: $\forall x \in M : (xRx)$,
- антирефлексивность (иррефлексивность): $\forall x \in M : \neg(xRx)$,
- корефлексивность: $\forall x, y \in M : (xRy \Rightarrow x = y)$,
- симметричность: $\forall x,\ y \in M : (xRy \Rightarrow yRx)$,
- антисимметричность: $\forall x,\ y \in M : (xRy \wedge yRx \Rightarrow x = y)$,
- асимметричность: $\forall x,\ y \in M : (xRy \Rightarrow \neg(yRx))$,
- транзитивность: $\forall x,\ y,\ z \in M : (xRy \wedge yRz \Rightarrow xRz)$,
- евклидовость: $\forall x, y, z \in M : (xRy \wedge xRz \Rightarrow yRz)$,
- полнота (или связность[7]): $\forall x, y \in M : (xRy \vee yRx)$,
- связность рус. (англ.) (или слабая связность[7]): $\forall x, y \in M : (x \neq y \Rightarrow xRy \vee yRx)$,
- трихотомия рус. (англ.): $\forall x, y \in M$ верно ровно одно из трех утверждений: xRy, yRx или $x = y$.

Симметричное отношение

Материал из Википедии — свободной энциклопедии [править | править код]

В математике бинарное отношение R на множестве X называется **симметричным**, если для каждой пары элементов множества (a, b) выполнение отношения $a\,R\,b$ влечёт выполнение отношения $b\,R\,a$.

Формально, отношение R симметрично, если $\forall a, b \in X,\ a\,R\,b \Rightarrow b\,R\,a$.

Антисимметричность отношения не является антонимом симметричного отношения. Оба свойства для некоторых отношений выполняются одновременно, а для некоторых не выполняется ни одно. Можно считать антонимом асимметричное отношение, так как единственное бинарное отношение, одновременно симметричное и асимметричное — это пустое отношение.

Примеры

Рефлексивное отношение в математике — бинарное отношение R на множестве X, при котором всякий элемент этого множества находится в отношении R с самим собой[1].

Формально, отношение R рефлексивно, если $\forall x \in X :\ (xRx)$.

Свойство рефлексивности при заданных отношениях матрицей характеризуется тем, что все диагональные элементы матрицы равняются 1; при заданных отношениях графом каждый элемент x имеет петлю — дугу (x, x).

Бинарное отношение R на множестве X является рефлексивным тогда и только тогда, когда его подмножеством является тождественное отношение id_X на множестве X ($\mathrm{id}_X = \{(x, x) | x \in X\}$), то есть $\mathrm{id}_X \subseteq R$.

Если aRa не имеет смысла, то отношение R называется **антирефлексивным** (или **иррефлексивным**)[1].

Если **антирефлексивное отношение** задано матрицей, то все диагональные элементы являются нулевыми. При задании такого отношения графом каждая вершина не имеет петли — нет дуг вида (x, x).

Формально антирефлексивность отношения R определяется как: $\forall x \in X :\ \neg(xRx)$.

Если условие рефлексивности выполнено не для всех элементов множества X, говорят, что отношение R **нерефлексивно**.

Мно́жество — одно из ключевых понятий математики; представляющее собой набор, **совокупность** каких-либо (вообще говоря любых) объектов — **элементов** этого множества[1]. Два множества равны тогда и только тогда, когда содержат в точности одинаковые элементы[2].

Изучением общих свойств множеств занимаются теория множеств, а также смежные разделы математики и математической логики. Примеры: множество жителей заданного города, множество непрерывных функций, множество решений заданного уравнения. Множество может быть пустым и непустым, упорядоченным и неупорядоченным, конечным и бесконечным, бесконечное множество может быть счётным или несчётным. Более того, как в наивной, так и в аксиоматической теориях множеств любой объект обычно считается множеством. Понятие множества позволяет практически всем разделам математики использовать общую идеологию и терминологию.

Существуют два основных способа задания множеств: перечислением элементов и их описанием.

Перечисление[править | править код]

Первый способ требует задать (перечислить) все элементы, входящие в множество.

Описание[править | править код]

Второй способ применяется, когда множество нельзя или затруднительно задать перечислением (например, если множество содержит бесконечное число элементов). В таком случае его можно описать свойствами принадлежащих ему элементов.

Мо́да — значение во множестве наблюдений, которое встречается наиболее часто. (Мода = типичность.) Иногда в совокупности встречается более чем одна мода *(например: 6, 2, 6, 6, 8, 9, 9, 9, 0; мода — 6 и 9)*. В этом случае можно сказать, что совокупность мультимодальна. Из структурных средних величин только мода обладает таким уникальным свойством. Как правило, мультимодальность указывает на то, что набор данных не подчиняется нормальному распределению.

Диспе́рсия случа́йной величины́ — мера разброса значений случайной величины относительно

её математического ожидания. Обозначается в русской литературе и (англ. *variance*) в зарубежной.

Корреля́ция (от лат. *correlatio* «соотношение»), или **корреляцио́нная зави́симость** — статистическая взаимосвязь двух или более случайных величин (либо величин, которые можно с некоторой допустимой степенью точности считать таковыми). При этом изменения значений одной или нескольких из этих величин сопутствуют систематическому изменению значений другой или других величин[1].

Сколько тест кейсов надо написать что бы протестировать функцию?

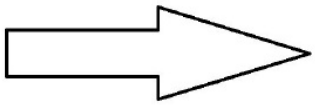
1 тест кейс – Три Браузера, 3 ОС, 3 сценария

1. С помощью pairwiseTool произведите выборку комбинаций исходных параметров и скиньте ссылку на скриншот результата в комментарии. Берётся условный сайт, который должен открываться на Win 7, Win 8 и Win 10. Поддерживаемые браузеры — Google Chrome, Opera, Microsoft Edge, Mozilla Firefox, Яндекс.Браузер. Пользователи могут использовать или не использовать AdBlock.
2. С помощью программы PICT сделайте выборку комбинаций исходных параметров и скиньте ссылку на скриншот результата в комментариях. Необходимо провести конфигурационное тестирование со следующими комплектующими:

- видеокарты: GeForce GT 730, GeForce GT 1030, GeForce GTX 1080, GeForce RTX 2070;
- процессоры: Intel Core i5, Intel Core i7, AMD Ryzen 7, Intel Core i9;

- память: 8GB, 16GB.

№	Переменная 1	Переменная 2	Переменная 3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1



№	OS	Mode	Light type
1	iOS	Night	Simple light
2	iOS	Day	Strobe light
3	Android	Night	Strobe light
4	Android	Day	Simple light

Контрпример — пример, опровергающий верность некоторого утверждения.

Построение контрпримера — обычный способ опровержения гипотез. Если имеется утверждение типа «Для любого *X* из множества *M* выполняется свойство *A*», то контрпримером для этого утверждения будет: «*Существует* объект X_0 из множества *M*, для которого свойство *A* не выполняется».

Контрпримером можно опровергнуть утверждение(в том числе в коде)

Квадрат Эйлера перешел в Pairwise testing

При сверхбольших обьемах данных нужно использовать Data-science или продвинутую статистику (например метод доверительных интервалов)

Severity	High Priority	Medium Priority	Low priority	Very Low priority
High	First	Fourth	eight	twelve
Medium	Second	Fifth	nine	Thirdteen
Low	Third	Six	ten	Fourteen
Very Low	seven	eleven	Fifteen	Sixteen