

Техники тест- дизайна



Анализ тестирования



(test analysis) - деятельность, которая направлена на определение тестовых условий путем анализа базиса тестирования



“

Проектирование теста

(test design) - активность по получению и определению тест-кейсов в соответствии с условиями тестирования.

”





Эквивалентное разбиение

Предположим, что для расчета скидки в онлайн-магазине, пользователь должен ввести в поле свой возраст. Допускается ввод только чисел. Расчет скидки производится следующим образом:

Возраст покупателя от 0 до 17 лет - скидка 50%

Возраст покупателя от 18 до 54 лет - скидка 25%

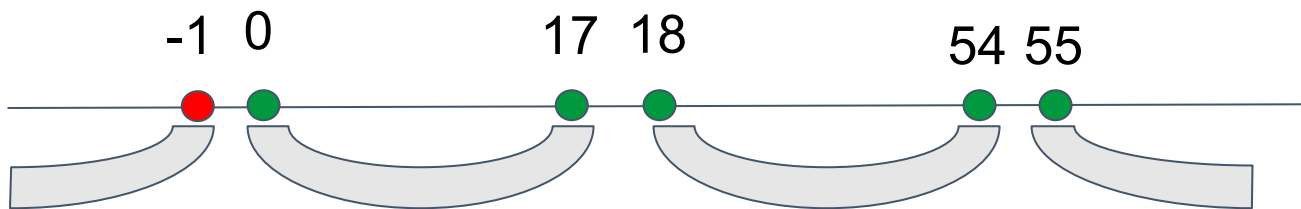
Возраст покупателя от 55 и выше - скидка 75%

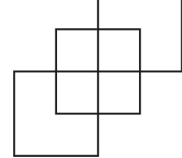
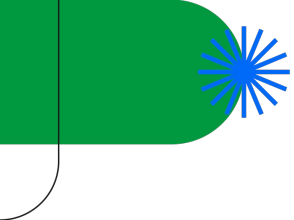
Эквивалентное разбиение

1. Определяем классы эквивалентности и граничные значения.

Класс эквивалентности (equivalence class) — набор данных, обрабатываемых одинаковым образом и приводящих к одинаковому результату

Граничное условие (border condition, boundary condition) — значение, находящееся на границе классов эквивалентности.





Эквивалентное разбиение

Негативный класс:

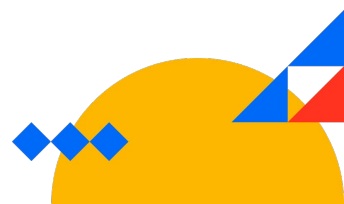
Возраст покупателя до 0 лет - сообщение об ошибке

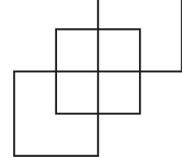
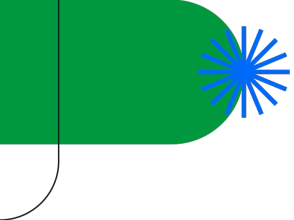
Позитивные классы:

Возраст покупателя от 0 до 17 лет - скидка 50%

Возраст покупателя от 18 до 54 лет - скидка 25%

Возраст покупателя от 55 и выше - скидка 75%





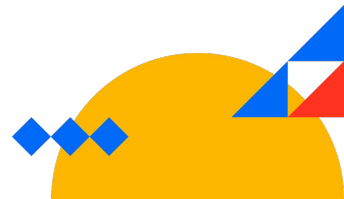
Эквивалентное разбиение

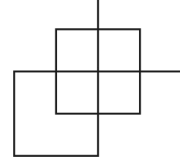
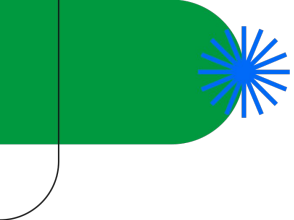
Негативный класс:

Ввод букв, спецсимволов, кода

Дополнительно проверяют пустое значение или 0, даже если оно входит в существующий класс.

Стратегия тестирования зависит от природы самих данных: например, для имени нужно проверять короткие, мужские, женские, распространенные имена и так далее. Но это уже дополнительные классы и не всегда есть на это время.



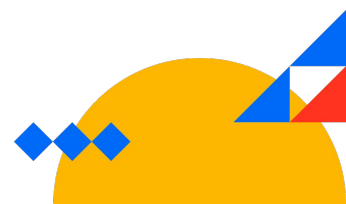


Эквивалентное разбиение

Во избежание маскирования дефектов негативные классы эквивалентности в тестовых сценариях следует использовать по отдельности, то есть избегать комбинаций одних негативных классов с другими. Дефекты могут быть маскированы, если при наличии нескольких дефектов обнаруживается только один из них.

В будущем мы рассмотрим этот принцип при создании тест-кейсов, в которых такое объединение тоже запрещено.

Данная формулировка отсутствует в версии 2023



Эквивалентное разбиение

2. Тестируем хотя бы одно значение из каждого класса. Так как код пишется таким образом, что все правила к одному из представителей класса применяются ко всем другим.

```
If (applicantAge >= 0 && applicantAge <=16)
    hireStatus="NO";
If (applicantAge >= 16 && applicantAge <=18)
    hireStatus="PART";
If (applicantAge >= 18 && applicantAge <=55)
    hireStatus="FULL";
If (applicantAge >= 55 && applicantAge <=90)
    hireStatus="NO";
```

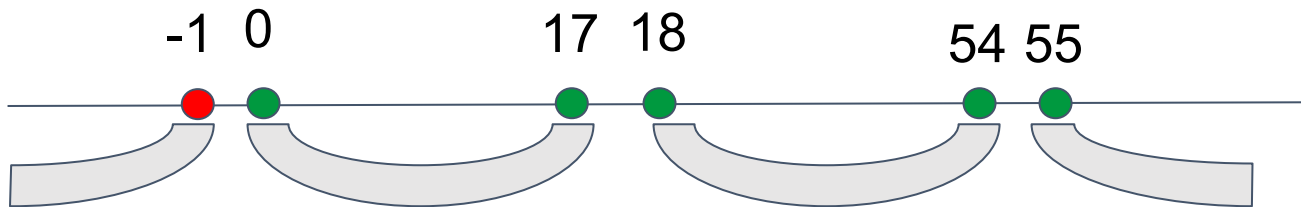
Эквивалентное разбиение

Позитивные классы	Представители класса	Ожидаемый результат
0 - 17	8	скидка 50%
18 - 54	27	скидка 25%
55 - +беск	100	скидка 75%
Негативные классы		
от -беск до -1	-5	сообщение об ошибке
Буквы	dmf	сообщение об ошибке
спецсимволы	\$&!@%	сообщение об ошибке

Анализ граничных значений

На практике одного эквивалентного разбиения недостаточно.

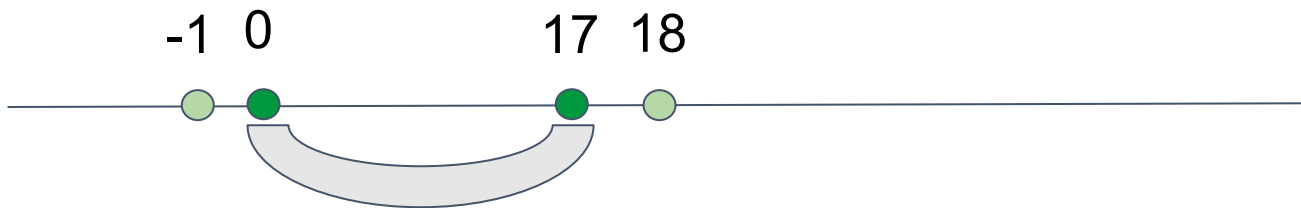
Метод анализа граничных значений является продолжением метода эквивалентного разбиения, но может быть применим, только если классы состоят из **упорядоченных числовых значений (в версии 2023 - упорядоченных классов)**. Максимальное и минимальное значение класса являются его границами [Beizer 1990]. Некорректное поведение более вероятно на границах класса, чем внутри класса.



Анализ граничных значений

1. Определяем остальные граничные значения. Существует два метода. Рассмотрим первый [Craig, 2002; Myers, 2011] :

- а) Тестирование нижней границы и значения до нее
- б) Тестирование верхней границы и значения после нее





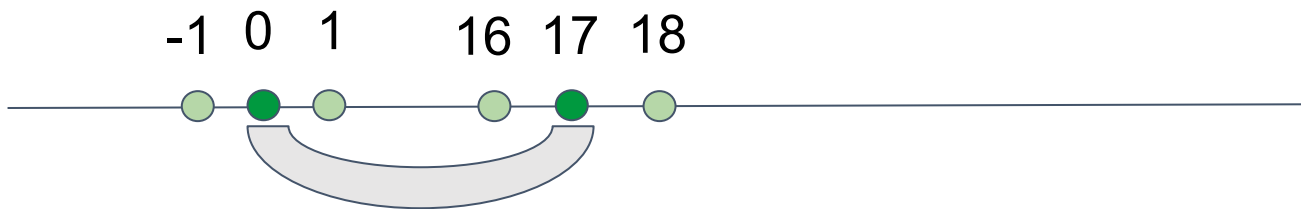
Эквивалентное разбиение

Позитивные классы	Граничные значения	Ожидаемый результат
0 - 17	-1, 0 и 17, 18	скидка 50%
18 - 54	17, 18 и 54, 55	скидка 25%
55 - +беск	54, 55, для +беск нет	скидка 75%
Негативные классы		
от -беск до -1	для -беск нет и -1, 0	сообщение об ошибке
Буквы	нельзя определить	сообщение об ошибке
спецсимволы	нельзя определить	сообщение об ошибке

Анализ граничных значений

Второй метод [Koomen, 2006; O'Regan, 2019]:

- а) Тестирование нижней границы и значения как до нее, так и после нее
- б) Тестирование верхней границы и значения как до нее, так и после нее



Эквивалентное разбиение

Позитивные классы	Граничные значения	Ожидаемый результат
0 - 17	-1, 0, 1 и 16, 17, 18	скидка 50%
18 - 54	17, 18, 19 и 53, 54, 55	скидка 25%
55 - +беск	54, 55, 56 для +беск нет	скидка 75%
Негативные классы		
от -беск до -1	для -беск нет и -2, -1, 0	сообщение об ошибке
Буквы	нельзя определить	сообщение об ошибке
спецсимволы	нельзя определить	сообщение об ошибке



Дополнительная информация

1. В случае тестирования десятичных дробей, граничные значения будут зависеть не только от целой, но и от дробной части.

Например, допустимые значения от 0,01 до 10.00.

Граничные значения для этого класса: 0.00, 0.01, 0.02 и 9.99, 10.00, 10.01

2. Значения, которые повторяются, второй раз тестирование не нужно. При выполнении тестовых заданий рекомендую оставлять комментарий о том, что они были удалены
3. Лучше указывать не конкретные примеры представителей, а их привязку к количеству символов и типу данных. Например, “абвг” заменить на 4 буквы.



Дополнительная информация

1. Существуют разные типы границ: технологические, логические и произвольные. О них можно прочитать [здесь](#).
2. Не всегда на проектах точно описаны все границы, но надо уметь их определять и тестировать. Как это сделать, можно узнать в этой [статье](#).
3. Логическая ловушка: умение различать представителей класса и представителей граничных значений из смежных классов. Например, для позитивного класса от 0 до 17 граничными значениями для 17 будут 16,17,18, где 18 будет представителем негативного смежного класса. Это не делает его представителем позитивного класса, но в таблицах и кейсах оно будет фигурировать как граничное значение позитивного класса.



Дополнительная информация

1. Классы эквивалентности можно использовать и для других аспектов, например, для формирования выборки устройств для тестирования (разрешение экрана, операционные системы)
2. Тестирование классов эквивалентности в равной степени применимо на модульном, интеграционном, системном и приемочном уровнях тестирования. Все это требует входных или выходных значений, которые могут быть разделены на основе системных требований. [Lee Copeland]



“ **Тестирование таблицы решений** ”

(decision table testing) - техника тестирования методом черного ящика, в которой тестовые сценарии проектируются для проверки комбинаций условий и действий, отраженных в таблице решений



Таблица принятия решений

Таблицы альтернатив – хороший способ записи сложных бизнес-правил, которые должны быть реализованы в системе. В процессе создания таблицы, тестировщик определяет условия (входы) и результирующие действия системы (выходы). Пары условий и действий образуют строки таблицы, при этом условия указываются сверху, а действия – снизу.

<i>Условие 1</i>	
<i>Условие 2</i>	
Действие 1	
Действие 2	

Таблица принятия решений

	Правило 1	Правило 2	Правило 3	Правило 4
<i>Наличие высшего образования</i>	Нет	Есть	Нет	Есть
<i>Опыт работы</i>	Нет	Нет	Есть	Есть
Оклад	500	600	600	700
Премия	5%	10%	15%	20%

Таблица принятия решений

	Правило 1	Правило 2	Правило 3	Правило 4	Правило 5	Правило 6	Правило 7	Правило 8
Условия								
Верный идентификатор ?	Нет	Нет	Нет	Нет	Да	Да	Да	Да
Верное количество?	Нет	Нет	Да	Да	Нет	Нет	Да	Да
Достаточно средств?	Нет	Да	Нет	Да	Нет	Да	Нет	Да
Действия								
Можно купить?	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Да

Таблица 5-8: Таблица решений для заказа на покупку на сайте "Браун и Дональдсон"

Таблица принятия решений

Правило 1 Правило 2 Правило 3 Правило 4 Правило 5

Условия

Верный идентификатор?	Нет	Да	Да	Да	Да
Верное количество?	НВ	Нет	Нет	Да	Да
Достаточно средств?	НВ	Нет	Да	Нет	Да

Действия

Можно купить?	Нет	Нет	Нет	Нет	Да
---------------	-----	-----	-----	-----	----

Таблица 5-9: Схлопнутая таблица решений, исключая "не важные" условия.

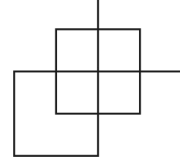
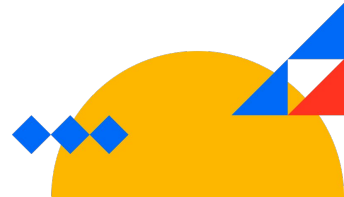


Таблица принятия решений

Применение: тестирование таблиц принятия решений может быть использовано, когда система должна реализовывать сложные бизнес-правила, когда эти правила могут быть представлены в виде комбинации условий и когда эти условия имеют дискретные действия, связанные с ними.

Задача для закрепления:

В дождливую погоду ты надеваешь дождевик, когда идет снег - надеваешь пальто, когда светит солнце - надеваешь солнечные очки, когда нет дождя и снега, ты носишь плащ. Считать, что может идти только один вид осадков (дождь и снег не идут одновременно).



“

”

Попарное тестирование
(pairwise testing) - техника тестирования методом черного ящика, при которой тестовые сценарии разрабатываются таким образом, чтобы выполнить тестирование для всех комбинаций параметр-значение






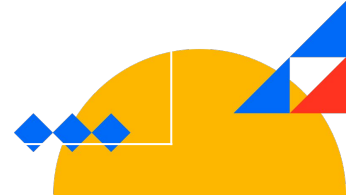
Попарное тестирование

Сайт должен работать в 8 браузерах - Internet Explorer 5.0, 5.5 и 6.0, Netscape 6.0, 6.1 и 7.0, Mozilla 1.1, и Opera 7;

Используя различные плагины - RealPlayer, MediaPlayer, без плагинов;

Запускаться на различных клиентских операционных системах - Windows 95, 98, ME, NT, 2000, и XP;

Получать страницы от разных веб-серверов - IIS, Apache, и WebLogic; работать с различными серверными операционными системами - Windows NT, 2000 и Linux.

- 8 браузеров, 3 плагина, 6 клиентских операционных систем, 3 сервера, 3 серверных операционных системы
 - 1296 комбинаций
- 
- 

Попарное тестирование

Существует две техники определения кейсов для попарного тестирования: ортогональные массивы (на практике не встречал) и алгоритм All Pairs

Ортогональный массив - это двумерный массив, с таким интересным свойством - выберите любые два столбца в массиве. В каждой паре столбцов будут встречаться все комбинации значений этих столбцов. Рассмотрим массив $L^4(2^3)$:

	1	2	3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

Попарное тестирование

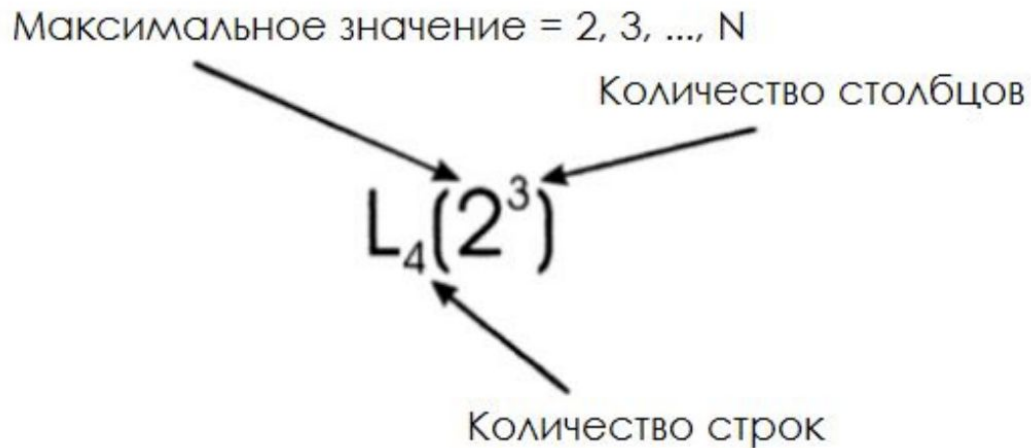


Рисунок 6-1: обозначение ортогонального массива



Попарное тестирование

Процесс использования ортогональных массивов для выбора попарного подмножества для тестирования заключается в следующем:

1. Определите переменные.
2. Определите количество значений, которое может принимать каждая переменная.
3. Определите ортогональный массив, у которого будет столбец для каждой переменной (каждый столбец ортогонального массива имеет столько же вариантов значений, сколько имеет ваша переменная).
4. Спроецируйте задачу тестирования на ортогональный массив.
5. Постройте тест-кейсы.

Попарное тестирование

	1	2	3	4	5
1	1	1	1	1	1
2	1	4	3	4	4
3	1	4	2	4	4
4	1	1	4	1	1
5	1	3	5	3	3
6	1	2	7	2	2
7	1	2	6	2	2
8	1	3	8	3	3
9	3	4	1	3	3
10	3	1	3	2	2
11	3	1	2	2	2
12	3	4	4	3	3
13	3	2	5	1	1
14	3	3	7	4	4
15	3	3	6	4	4
16	3	2	8	1	1
17	2	3	1	2	1
18	2	2	3	3	4
19	2	2	2	3	4
20	2	3	4	2	1
21	2	1	5	4	3
22	2	4	7	1	2
23	2	4	6	1	2
24	2	1	8	4	3
25	4	2	1	4	3
26	4	3	3	1	2
27	4	3	2	1	2
28	4	2	4	4	3
29	4	4	5	2	1
30	4	1	7	3	4
31	4	1	6	3	4
32	4	4	8	2	1

33	5	2	1	4	2
34	5	3	3	1	3
35	5	3	2	1	3
36	5	2	4	4	2
37	5	4	5	2	4
38	5	1	7	3	1
39	5	1	6	3	1
40	5	4	8	2	4
41	7	3	1	2	4
42	7	2	3	3	1
43	7	2	2	3	1
44	7	3	4	2	4
45	7	1	5	4	2
46	7	4	7	1	3
47	7	4	6	1	3
48	7	1	8	4	2
49	6	4	1	3	2
50	6	1	3	2	3
51	6	1	2	2	3
52	6	4	4	3	2
53	6	2	5	1	4
54	6	3	7	4	1
55	6	3	6	4	1
56	6	2	8	1	4
57	8	1	1	1	4
58	8	4	3	4	1
59	8	4	2	4	1
60	8	1	4	1	4
61	8	3	5	3	2
62	8	2	7	2	3
63	8	2	6	2	3
64	8	3	8	3	2



Попарное тестирование

1. *Определяем переменные и количество значений для них*

8 браузеров, 3 плагина, 6 клиентских операционных систем, 3 сервера, 3 серверных операционных системы

2. *Определите ортогональный массив, у которого будет столбец для каждой переменной*

Идеальным размером для ортогонального массива будет $8^1 6^3 1^3$ (один столбец от 1 до 8, один столбец от 1 до 6 и три столбца от 1 до 3). Такого массива не существует. Но есть альтернатива $L_{64}(8^2 4^3)$



Попарное тестирование

4. Спроецируйте задачу тестирования на ортогональный массив.

Значения браузера у нас проецируются на первый столбец ортогонального массива/ Ячейка, содержащая "1", будет представлена значением "IE 5.0"; содержащая "2" - значением "IE 5.5"; содержащая "3" - значением "IE 6.0" и т.д.

5. Необходимо проделать этот шаг для каждой переменной.

6. После заполнить недостающие значения любыми значениями переменной для поддержки ортогональности.

Попарное тестирование

	Browser	2	3	4	5
1	IE 5.0	1	1	1	1
2	1	4	3	4	4
3	1	4	2	4	4
4	1	1	4	1	1
5	1	3	5	3	3
6	1	2	7	2	2
7	1	2	6	2	2
8	1	3	8	3	3
9	IE 6.0	4	1	3	3
10	3	1	3	2	2
11	3	1	2	2	2
12	3	4	4	3	3
13	3	2	5	1	1
14	3	3	7	4	4
15	3	3	6	4	4
16	3	2	8	1	1
17	IE 5.5	3	1	2	1
18	2	2	3	3	4
19	2	2	2	3	4
20	2	3	4	2	1
21	2	1	5	4	3
22	2	4	7	1	2
23	2	4	6	1	2
24	2	1	8	4	3
25	Net 6.0	2	1	4	3
26	4	3	3	1	2
27	4	3	2	1	2
28	4	2	4	4	3
29	4	4	5	2	1
30	4	1	7	3	4
31	4	1	6	3	4

Попарное тестирование

	Browser	Plug-in	3	4	5
1	IE 5.0	None	1	1	1
2	IE 5.0	4	3	4	4
3	IE 5.0	4	2	4	4
4	IE 5.0	None	4	1	1
5	IE 5.0	MediaPlayer	5	3	3
6	IE 5.0	RealPlayer	7	2	2
7	IE 5.0	RealPlayer	6	2	2
8	IE 5.0	MediaPlayer	8	3	3
9	IE 6.0	4	1	3	3
10	IE 6.0	None	3	2	2
11	IE 6.0	None	2	2	2
12	IE 6.0	4	4	3	3
13	IE 6.0	RealPlayer	5	1	1
14	IE 6.0	MediaPlayer	7	4	4
15	IE 6.0	MediaPlayer	6	4	4
16	IE 6.0	RealPlayer	8	1	1
17	IE 5.5	MediaPlayer	1	2	1
18	IE 5.5	RealPlayer	3	3	4
19	IE 5.5	RealPlayer	2	3	4
20	IE 5.5	MediaPlayer	4	2	1
21	IE 5.5	None	5	4	3
22	IE 5.5	4	7	1	2
23	IE 5.5	4	6	1	2
24	IE 5.5	None	8	4	3
25	Net 6.0	RealPlayer	1	4	3
26	Net 6.0	MediaPlayer	3	1	2
27	Net 6.0	MediaPlayer	2	1	2
28	Net 6.0	RealPlayer	4	4	3
29	Net 6.0	4	5	2	1
30	Net 6.0	None	7	3	4
31	Net 6.0	None	6	3	4

Lee Copeland, «A practitioner's guide to software test design»

Попарное тестирование

	Browser	Plug-in	Client OS	Server	Server OS
1	IE 5.0	None	Win 95	IIS	Win NT
2	IE 5.0	4	Win ME	4	4
3	IE 5.0	4	Win 98	4	4
4	IE 5.0	None	Win NT	IIS	Win NT
5	IE 5.0	MediaPlayer	Win 2000	WebLogic	Linux
6	IE 5.0	RealPlayer	7	Apache	Win 2000
7	IE 5.0	RealPlayer	Win XP	Apache	Win 2000
8	IE 5.0	MediaPlayer	8	WebLogic	Linux
9	IE 6.0	4	Win 95	WebLogic	Linux
10	IE 6.0	None	Win ME	Apache	Win 2000
11	IE 6.0	None	Win 98	Apache	Win 2000
12	IE 6.0	4	Win NT	WebLogic	Linux
13	IE 6.0	RealPlayer	Win 2000	IIS	Win NT
14	IE 6.0	MediaPlayer	7	4	4
15	IE 6.0	MediaPlayer	Win XP	4	4
16	IE 6.0	RealPlayer	8	IIS	Win NT
17	IE 5.5	MediaPlayer	Win 95	Apache	Win NT
18	IE 5.5	RealPlayer	Win ME	WebLogic	4
19	IE 5.5	RealPlayer	Win 98	WebLogic	4
20	IE 5.5	MediaPlayer	Win NT	Apache	Win NT
21	IE 5.5	None	Win 2000	4	Linux
22	IE 5.5	4	7	IIS	Win 2000
23	IE 5.5	4	Win XP	IIS	Win 2000
24	IE 5.5	None	8	4	Linux
25	Net 6.0	RealPlayer	Win 95	4	Linux
26	Net 6.0	MediaPlayer	Win ME	IIS	Win 2000
27	Net 6.0	MediaPlayer	Win 98	IIS	Win 2000
28	Net 6.0	RealPlayer	Win NT	4	Linux
29	Net 6.0	4	Win 2000	Apache	Win NT
30	Net 6.0	None	7	WebLogic	4
31	Net 6.0	None	Win XP	WebLogic	4
32	Net 6.0	4	8	Apache	Win NT
33	Net 6.1	RealPlayer	Win 95	4	Win 2000
34	Net 6.1	MediaPlayer	Win ME	IIS	Linux
35	Net 6.1	MediaPlayer	Win 98	IIS	Linux

Попарное тестирование

	Browser	Plug-in	Client OS	Server	Server OS
1	IE 5.0	None	Win 95	IIS	Win NT
2	IE 5.0	None	Win ME	IIS	Win NT
3	IE 5.0	None	Win 98	IIS	Win NT
4	IE 5.0	None	Win NT	IIS	Win NT
5	IE 5.0	MediaPlayer	Win 2000	WebLogic	Linux
6	IE 5.0	RealPlayer	Win 95	Apache	Win 2000
7	IE 5.0	RealPlayer	Win XP	Apache	Win 2000
8	IE 5.0	MediaPlayer	Win 98	WebLogic	Linux
9	IE 6.0	None	Win 95	WebLogic	Linux
10	IE 6.0	None	Win ME	Apache	Win 2000
11	IE 6.0	None	Win 98	Apache	Win 2000
12	IE 6.0	None	Win NT	WebLogic	Linux
13	IE 6.0	RealPlayer	Win 2000	IIS	Win NT
14	IE 6.0	MediaPlayer	Win 95	IIS	Win NT
15	IE 6.0	MediaPlayer	Win XP	IIS	Win NT
16	IE 6.0	RealPlayer	Win 98	IIS	Win NT
17	IE 5.5	MediaPlayer	Win 95	Apache	Win NT
18	IE 5.5	RealPlayer	Win ME	WebLogic	Win NT
19	IE 5.5	RealPlayer	Win 98	WebLogic	Win NT
20	IE 5.5	MediaPlayer	Win NT	Apache	Win NT
21	IE 5.5	None	Win 2000	IIS	Linux
22	IE 5.5	None	Win 95	IIS	Win 2000
23	IE 5.5	None	Win XP	IIS	Win 2000
24	IE 5.5	None	Win 98	IIS	Linux
25	Net 6.0	RealPlayer	Win 95	IIS	Linux
26	Net 6.0	MediaPlayer	Win ME	IIS	Win 2000
27	Net 6.0	MediaPlayer	Win 98	IIS	Win 2000
28	Net 6.0	RealPlayer	Win NT	IIS	Linux
29	Net 6.0	None	Win 2000	Apache	Win NT
30	Net 6.0	None	Win 95	WebLogic	Win NT
31	Net 6.0	None	Win XP	WebLogic	Win NT
32	Net 6.0	None	Win 98	Apache	Win NT
33	Net 6.1	RealPlayer	Win 95	IIS	Win 2000
34	Net 6.1	MediaPlayer	Win ME	IIS	Linux
35	Net 6.1	MediaPlayer	Win 98	IIS	Linux

Lee Copeland, «A practitioner's guide to software test design»



Попарное тестирование






Алгоритм All Pairs позволяет генерировать все пары автоматически. Самым популярным инструментом для этого является PICT, но существует еще несколько десятков других решений. На практике используют именно этот метод.

Выходные данные полученные этим способом могут отличаться от данных, полученных в результате применения ортогональных массивов.



Попарное тестирование

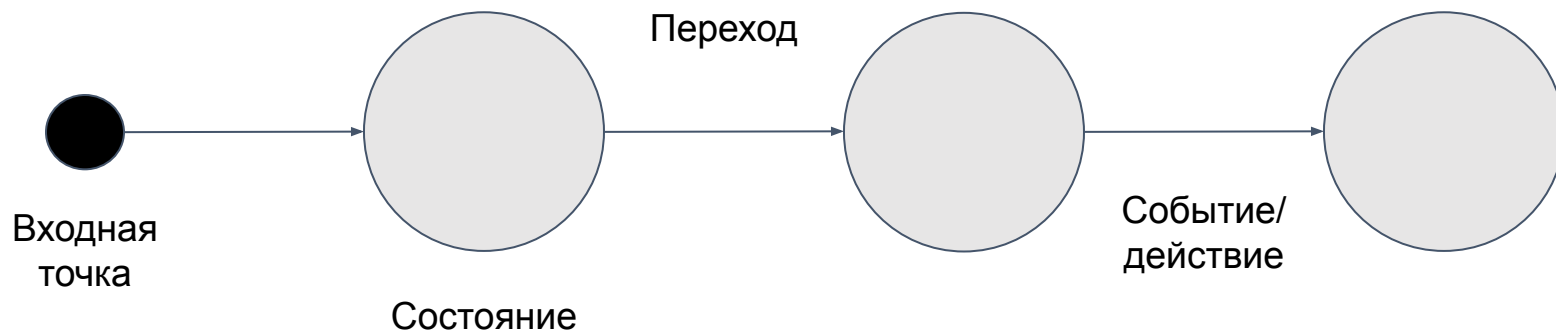
Применение: на всех уровнях тестирования. Всё, что требуется - это комбинации входных точек (параметров), каждый из которых принимает различные значения до такой степени, что результат может привести к комбинационному взрыву, так как для тестирования существует слишком много комбинаций.

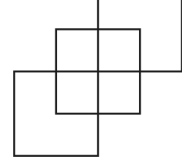
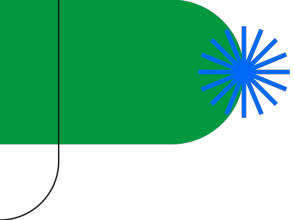


“Тестирование таблицы переходов
(state transition testing) - разработка
тестов методом черного ящика, при
котором сценарии тестирования
строятся на основе модели
переходов состояний.”

Тестирование состояний и переходов

Диаграмма состояний и переходов включает в себя состояние, переход, событие, действие и входную точку. Всегда включает в себя только один объект.





Тестирование состояний и переходов

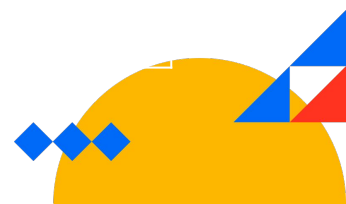
Состояние - это состояние, в котором система ожидает возникновения одного или нескольких событий.

Переход - это изменение состояния из одного в другое, произошедшее благодаря какому-то событию.

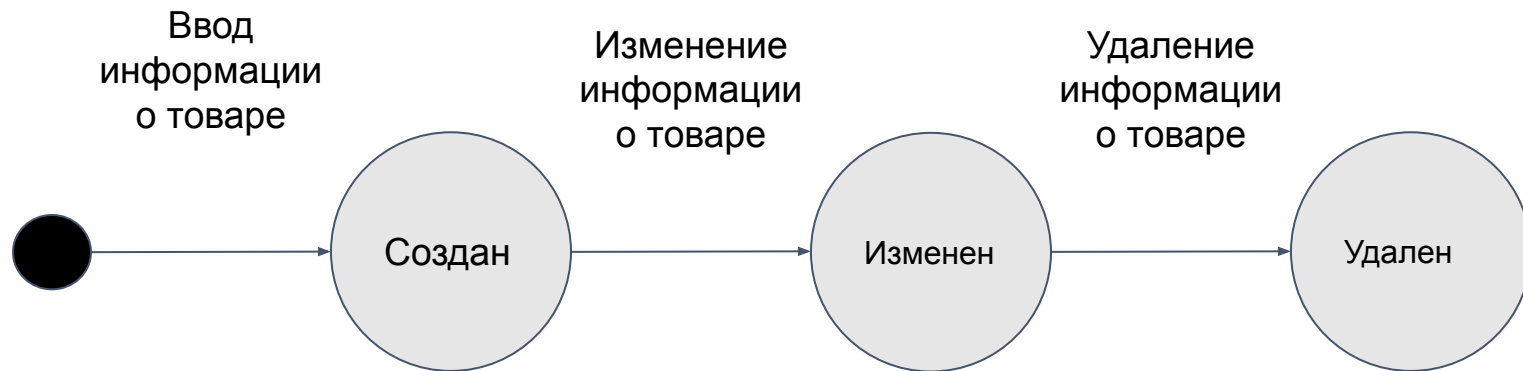
Событие - что-то, что вызывает изменение состояния системы.

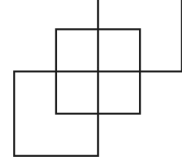
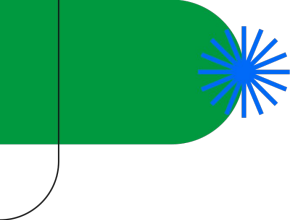
Действие - это операция, которая вызвана изменением состояния.

Входная точка на диаграмме показана черной точкой, а точка выхода показана в виде значка "бычьего глаза".



Тестирование состояний и переходов

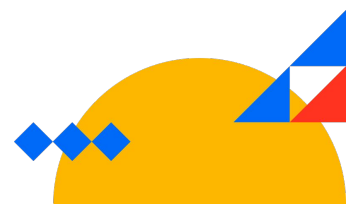




Тестирование состояний и переходов

Применение: тестирование с помощью таблицы переходов наиболее распространено в сфере встроенного ПО и при тестировании программных меню. Метод также подходит для моделирования бизнес-сценариев, имеющих конкретные состояния, или для тестирования переходов по экранным формам.

В версии 2023 этот метод расписан подробнее с учетом различных покрытий.



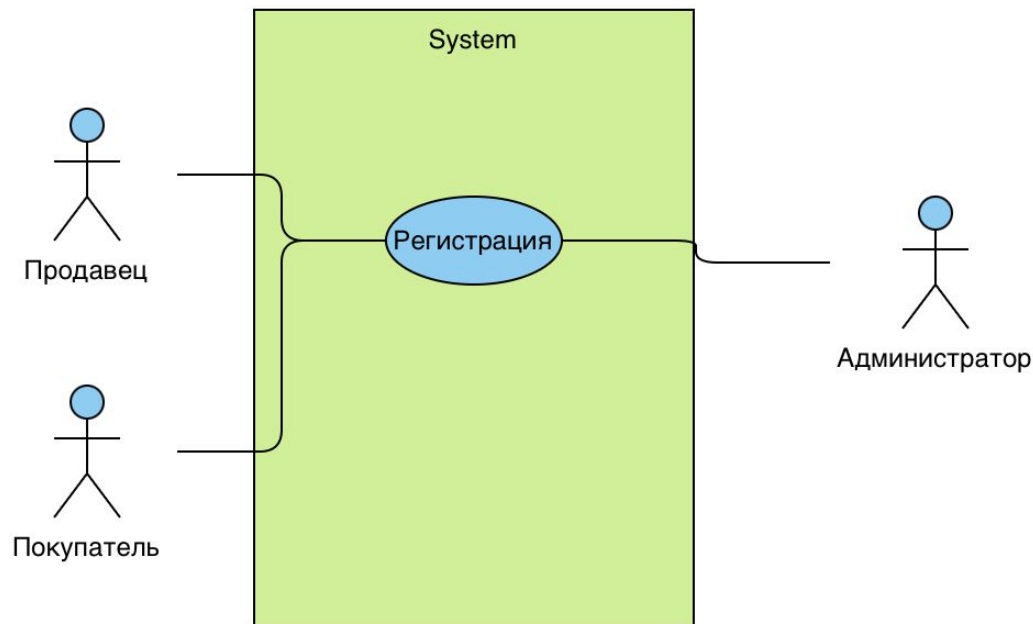
“

Тестирование по сценариям использования (use case testing) - метод тестирования черного ящика, в котором сценарии тестирования разрабатываются на основе выполнения сценариев использования.

”



Тестирование по сценариям использования



Тестирование по сценариям использования

Название use case	Регистрация
Описание use case	Пользователь регистрируется в системе, чтобы получить доступ к работе в ней
Акторы	Продавцы, Покупатели, Администратор
Предусловия	Система должна быть подсоединена к сети
Постусловия	После успешного регистрации пользователю высылается письмо о подтверждении на email

Тестирование по сценариям использования

Основные сценарии	Номер	Шаги
Акторы/пользователи	1	Ввод логина Ввод пароля
	2	Проверить логин и пароль
	3	Зарегистрировать пользователя
Расширения	1a	Логин не прошел валидацию Система выводит сообщение об ошибке
		...



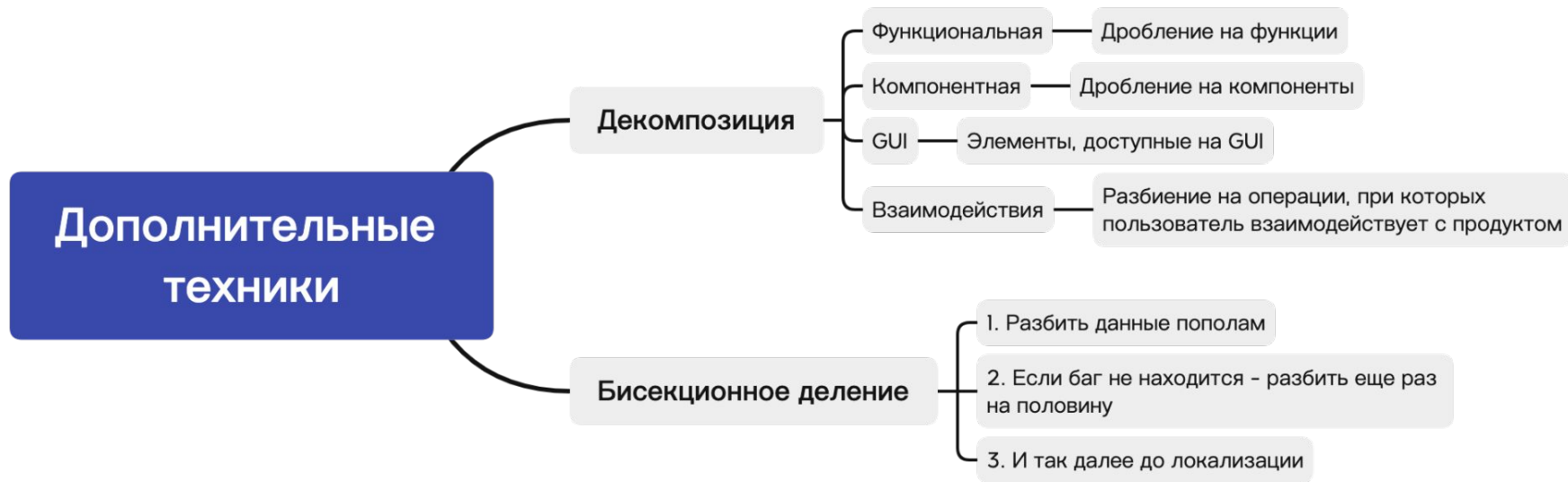
Дополнительные техники

Предположение об ошибках (error guessing) - техника тестирования, в которой тесты получены на основе знаний тестировщика о ранее обнаруженных сбоях или общих знаниях о типах отказов

Отображение причинно-следственных связей (cause-effect graphing) - метод тестирования черного ящика, при котором тестовые сценарии разрабатываются на основе диаграмм причинно-следственных связей. (в версии 2024 отсутствует)

Диаграмма причинно-следственных связей - графическое представление логических отношений между входными данными (причинами) и связанными с ними выходными данными (следствиями).

Дополнительные техники



<https://qaschool.ru/blog/dekompozirui-eto/>

<https://habr.com/ru/post/468087/>