1. **Что такое тест-дизайн?**

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

В процессе тест-дизайна необходимо получить ответ на 2 вопроса:

1) Что тестировать?

2) Как тестировать?

Чтобы получить ответ на эти вопросы, необходимо, используя различные стратегии и техники тест-дизайна, создать набор тестовых случаев, обеспечивающий оптимальное тестовое покрытие тестируемого приложения.

Как запороть все?

1. Найти неверные входные значения
2. Пропустить самые популярные комбинации
3. Упустить взаимодействие переменных

2. Цели тест-дизайна.

Можно выделить главные цели тест-дизайна:

1. *Придумать тесты, которые обнаружат наиболее серьезные ошибки продукта в наибольшем количестве.*

Конечно, можно придумывать тесты, которые находят несерьезные ошибки, но тогда тестирование будет неэффективным. Также нам необходимо найти как можно больше серьезных дефектов.

2. *Минимизировать количество тестов, необходимых для нахождения большинства серьезных ошибок.*

Мы можем придумать столько тестов, сколько не в состоянии будем выполнить. Поэтому перед разработчиками тестов всегда стоит задача – сохранить эффективность тестов (то есть их способность обнаруживать серьезные ошибки) без увеличения их числа.

3. Хороший тест дизайн

Хороший тест-дизайн: минимум кейсов -> максимум багов.

Много тестов ≠ максимально протестированный продукт

Наличие большого количества тестов не может гарантировать, что продукт будет максимально протестирован!

В дело вступает реальность: в большинстве случаев провести достаточное количество тестов невозможно

Чтобы протестировать в сжатые сроки, от каких-то тестов придётся отказаться. Каждый из них может найти потенциальный дефект, и получается, что, отказываясь от проведения того или иного теста, мы повышаем вероятность пропуска дефекта.
Главная задача тестировщика в этом случае - выбрать, какие тесты мы не будем проводить.
Тест-дизайн позволяет нам построить как раз такие тесты, которые позволяют нам с достаточной степенью уверенности сказать, что мы проверили продукт максимально полно и завершили тестирование.

4. План работы над тест-дизайном

- **Анализ имеющихся проектных артефактов: документация (спецификации, требования, планы), модели, исполняемый код и т.д.**

Откуда получить информацию тестировщику?

- бизнес-требования, спецификации, тех.дизайн, РИПВ, диаграммы процессов и т.д
- беседа с аналитиком, менеджером, системным архитектором, заказчиком.
- беседа с разработкой.

Лучше использовать все 3 источника информации для анализа.

- Оценка и анализ рисков (то есть проблем, которые могут быть при использовании продукта)

- Написание тестовой документации
- Проектирование и создание тестовых случаев test cases

5. Методики черного ящика

5.1 Эквивалентное разбиение (Equivalence partitioning)
5.2 Анализ граничных значений (Boundary Value Analysis)
5.3 Тестирование таблицы решений (Decision table testing)
5.4 Тестирование таблицы переходов (State transition testing)
5.5 Тестирование по сценариям использования (Use case testing)
5.6 Предугадываение ошибок (Error Guessing - EG)
5.7 Техника причина/следствие (Cause/Effect - CE)
5.8 Исследовательское тестирование (EXPLORATORY TESTING - ET)
5.9 Исчерпывающее тестирование (Exhaustive Testing - ET)
5.1 Парное (PAIRWISE)

5.1 Эквивалентное разбиение

Подход заключается в следующем: входные данные для программного обеспечения или системы разбиваются на группы (классы), от которых ожидается сходное поведение, то есть они должны обрабатываться аналогичным образом. Таким образом,

нет необходимости тестировать все возможные входные данные, необходимо проверить по отдельно взятому представителю класса.

Таким образом, Эквивалентные классы – классы или объединения элементов, к которым ПО применяет одинаковую логику.

Тестовые сценарии эквиваленты, если

- ❖ Они тестируют одно и тоже;
- ❖ Если один из них выявляет ошибку, то и остальные выявят ее;
- ❖ Если одни из них не выявляет ошибку, то и остальные не выявят.

Если определен класс эквивалентности, необходимо протестировать только одно-два значения из него.

Примерный алгоритм использования техники:

1. Определить классы эквивалентности. Это главный шаг техники. От него во многом зависит эффективность её применения.

2. Выбрать одного представителя от каждого класса. На этом шаге из каждого эквивалентного набора тестов мы выбираем один тест.

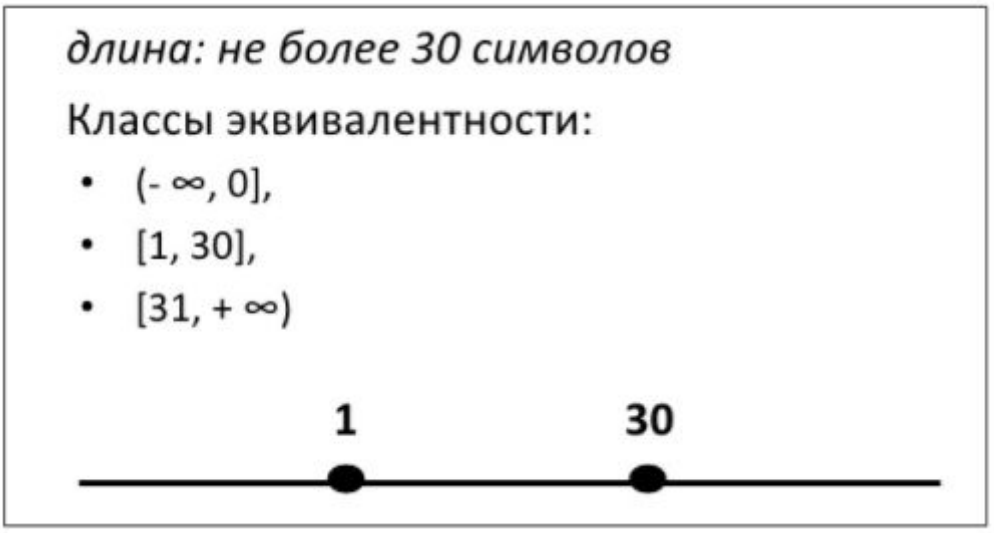
3. Выполнить тесты. На этом шаге мы выполняем тесты от каждого класса эквивалентности.

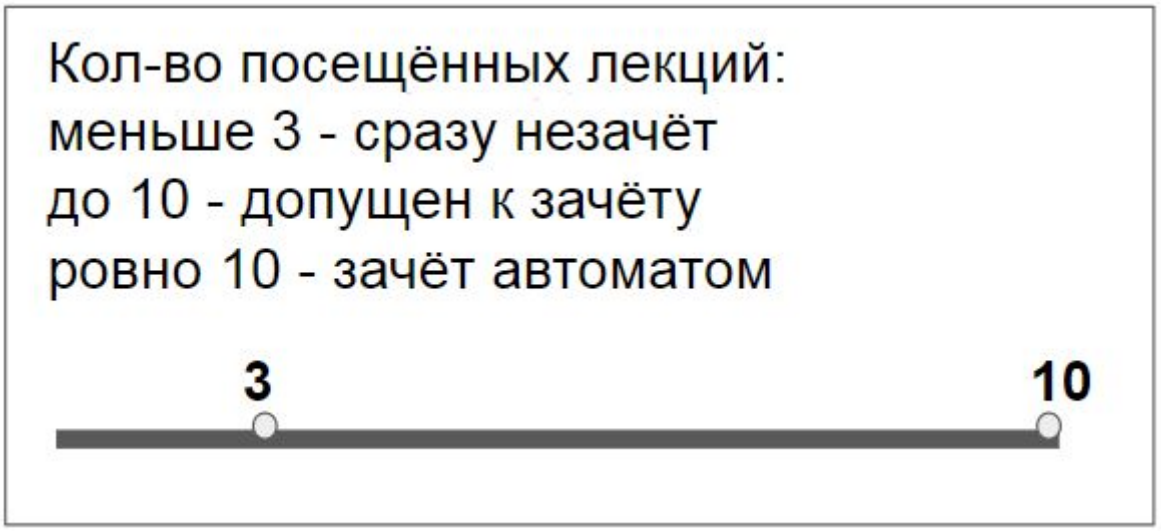
Эквивалентные области (или классы) могут быть определены как для валидных, так и для невалидных данных.

К плюсам можно отнести заметное сокращение времени и улучшение структурированности тестирования.

К минусам можно отнести то, что при неправильном использовании техники мы рискуем не обнаружить баги.

Примеры:





Система просит ввести в поле арабскую цифру.

Класс эквивалентности = [0,1,2,3,4,5,6,7,8,9]

Пример 1

Подсчет комиссии при отмене бронирования авиабилетов.

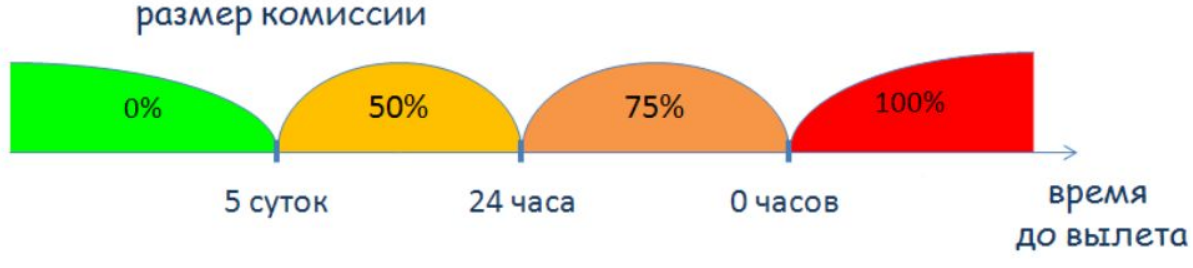
Условие: Размер комиссии зависит от времени до вылета, когда совершена отмена:

- За 5 суток до вылета комиссия составляет 0%
- Меньше 5 суток, но больше 24 часов – 50%
- Меньше 24 часов до вылета – 75%
- После вылета – 100%

Решение:

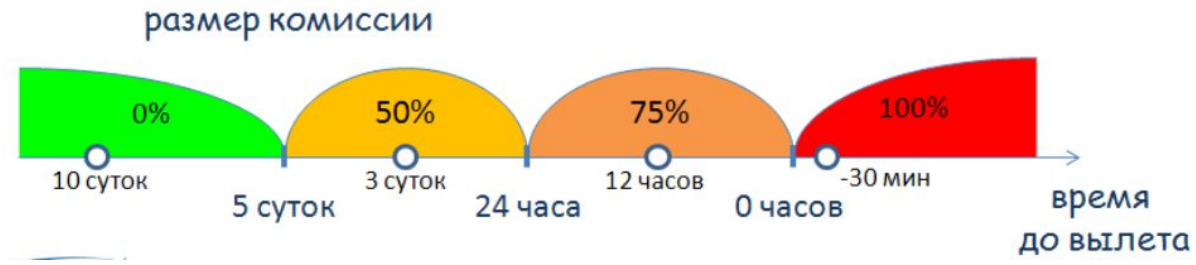
1. Определим классы эквивалентности (для каждого теста из этих классов мы ожидаем получить одинаковый результат):

- 1 класс: время до вылета > 5 суток
- 2 класс: 24 часа < время до вылета < 5 суток
- 3 класс: 0 часов < время до вылета < 24 часа
- 4 класс: время до вылета < 0 часов (вылет уже состоялся)



2. Выберем представителя от каждого класса. Здесь мы можем поступить, как нам хочется, и выбрать любые значения из класса. Ведь, если предположить, что мы правильно разбили на классы эквивалентности, то нет разницы, какое значение из диапазона мы выберем.

- ❖ время до вылета = 10 суток (тест из 1-го класса)
- ❖ время до вылета = 3 суток (тест из 2-го класса)
- ❖ время до вылета = 12 часов (тест из 3-го класса)
- ❖ время до вылета = -30 мин (тест из 4-го класса)



Выполним тесты:

1. Отменим бронь за 10 суток до вылета и проверим, что комиссия составила 0%.
2. Отменим бронь за 3 суток до вылета и проверим, что комиссия составила 50%.
3. Отменим бронь за 12 часов до вылета и проверим, что комиссия составила 75%.
4. Отменим бронь через 30 мин после вылета и проверим, что комиссия составила 100%.

5.2 Анализ граничных значений.

Этот подход тесно связан с разбиением на классы эквивалентности, т.к. именно классы являются источником этих самых границ.

Если техника анализа классов эквивалентности ориентирована на тестовое покрытие, то техника анализа граничных значений основана на рисках.

Эта техника начинается с идеи о том, что программа может сломаться в области граничных значений:

❖ При разработке большое число проблем возникает на границах входных переменных.

❖ Даже если эквивалентные классы найдены правильно, то граничные значения могут быть ошибочно отнесены к другому классу.

Примерный алгоритм использования техники анализа граничных значений:

1. Во-первых, нужно выделить классы эквивалентности. Опять же, это очень важный шаг, и от правильности разбиения на классы эквивалентности зависит эффективность тестов граничных значений.

2. Далее нужно определить граничные значения этих классов.

3. Нам нужно понять, к какому классу будет относиться каждая граница.

4. Для каждой границы нам нужно провести тесты по проверке значения до границы, на границе, и сразу после границы.

Количество тестов для проверки граничных значений будет равно количеству границ, умноженному на 3. Рекомендуется проверять значения вплотную к границе.

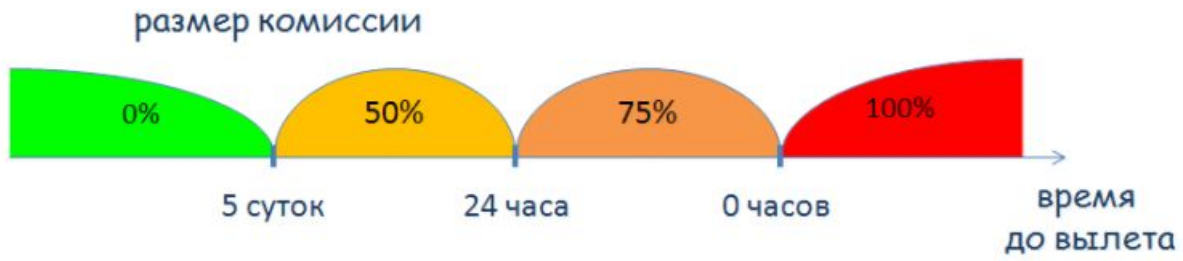
Подсчет комиссии при отмене бронирования авиабилетов.

Размер комиссии зависит от времени до вылета, когда совершена отмена:

- ❖ Более и ровно за 5 суток до вылета комиссия 0%
- ❖ Меньше 5 суток, но больше 24 часов – 50%
- ❖ Меньше 24 часов до вылета – 75%
- ❖ После вылета – 100%

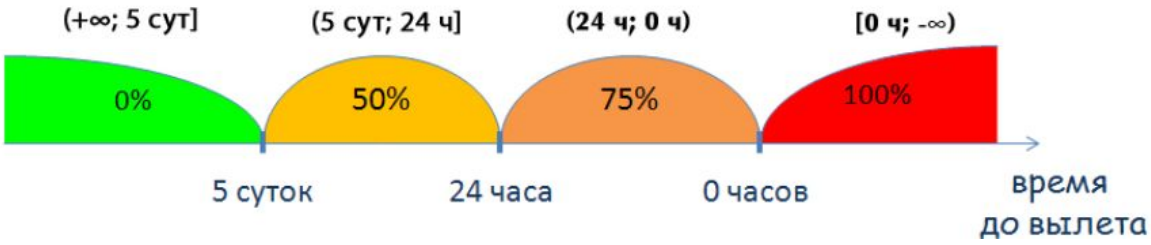
Определим границы:

- 5 суток
- 24 часа
- 0 часов



3. Определим, к какому классу относятся границы:

- 5 суток – к 1-му классу
- 24 часа – к 2-му классу
- 0 часов – к 4-му классу
- 1 класс: время до вылета >= 5 суток
- 2 класс: 24 часа =< время до вылета < 5 суток
- 3 класс: 0 часов < время до вылета < 24 часа
- 4 класс: время до вылета =< 0 часов (вылет уже состоялся)



4. Протестируем значения на границах, до и после них:

1. Отменим бронь за 5 суток + 1 секунду до вылета (или просто постараемся выполнить бронь как можно ближе к границе, но слева от нее) и проверим, что комиссия равна 0%.

2. Отменим бронь ровно за 5 суток до вылета и проверим, что комиссия равна 0%.
3. Отменим бронь за 5 суток – 1 секунду до вылета и проверим, что комиссия равна 50%.
4. Отменим бронь за 24 часа + 1 секунду до вылета и проверим, что комиссия равна 50%.
5. Отменим бронь ровно за 24 часа до вылета и проверим, что комиссия равна 50%.
6. Отменим бронь за 24 часа - 1 секунду до вылета и проверим, что комиссия равна 75%.
7. Отменим бронь за 1 секунду до вылета и проверим, что комиссия равна 75%.
8. Отменим бронь ровно во время вылета и проверим, что комиссия равна 100%.
9. Отменим бронь спустя 1 секунду после вылета и проверим, что комиссия равна 100%.

Мы получили 9 тестов, по 3 теста на каждую границу.

Если мы имеем диапазон целых значений, и граница у нас находится в числе 10, то мы будем проводить тесты:

- с числом 9 (вплотную до границы),
- 10 (саму границу)
- 11 (сразу после границы)

... 3 4 5 6 7 8 **9 10 11** 12 13...

Если речь о времни, и граница 13:00:00, то будем проводить тесты:

- 12:59:59 (вплотную *до границы*),
- 13:00:00 (*саму границу*)
- 13:00:01 (сразу *после границы*).

..12:59:58 **12:59:59 13:00:00**
13:00:01 13:00:02...

Если речь о дате (день рождения, праздник и тд) и граница 15.05.2017, то будем проводить тесты:

- 14.05.2017 (вплотную *до границы*),
- 15.05.2017 (*саму границу*),
- 16.05.2017 (сразу *после границы*).

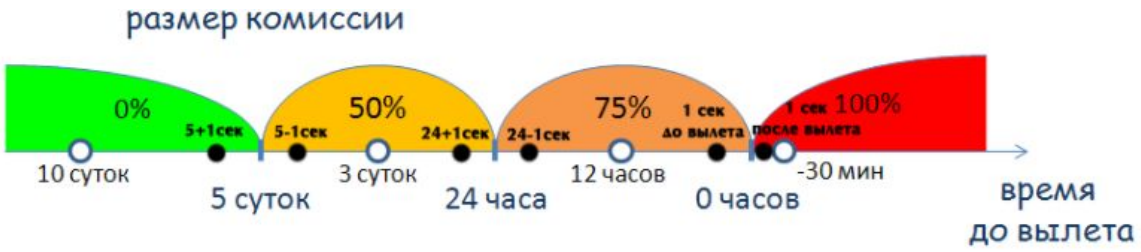
Эквивалентное разбиение + Анализ граничных значений

Если суммировать тесты из примера, необходимые для проверки классов эквивалентности и граничных значений, получим 4 + 9 =13 тестов.

1. за 5 суток + 1 секунду до вылета
2. ровно за 5 суток до вылета
3. за 5 суток – 1 секунду до вылета

4. за 24 часа + 1 секунду до вылета
5. ровно за 24 часа до вылета
6. за 24 часа - 1 секунду до вылета
7. за 1 секунду до вылета
8. во время вылета
9. спустя 1 секунду
   после вылета

1. за 10 суток до вылета
2. за 3 суток до вылета
3. за 12 часов до вылета
4. через 30 мин после вылета



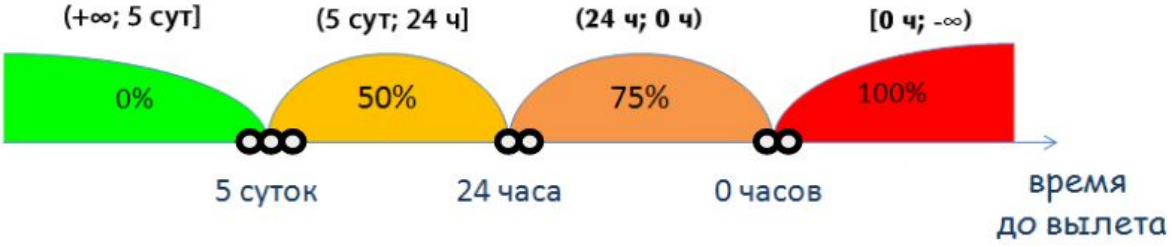
Использование классов эквивалентности и граничных условий вместе:

- ❖ Техника анализа классов эквивалентности говорит о том, что мы должны выбрать минимум по одному значению из каждого класса.
- ❖ Так как граница обычно относится к какому-то классу, то можно использовать ее как представителя этого класса.
- ❖ Тогда мы сэкономим определенное количество тестов.

Сокращать тесты нужно на свой страх и риск. Если считаете, что необходимо проверить обычные (не граничные) тесты – сделайте это. Лучше перестраховаться, чем пропустить ошибки.

Если следовать этой рекомендации, то в нашем случае останется 7 тестов.

1. за 5 суток - 1 секунду до вылета
2. ровно за 5 суток до вылета
3. за 5 суток + 1 секунду до вылета
4. ровно за 24 часа до вылета
5. за 24 часа - 1 секунду до вылета
6. во время вылета
7. спустя 1 секунду после вылета



Пример 2

Покупатель, если приобретает хотя бы 1 акционный товар, получает скидку 2% на весь чек, с 2х до 5 товаров - скидка 5% на весь чек, более 5 акционных товаров - скидка 10% на весь чек. Не приобрёл акционных товаров - нет скидки, 0%.

Граничные значения: 1, 2, 5.

Если переписать формулировку примера по классам:
(- ∞; 1)
Покупка до одного товара, для этого примера это только 0; граница 1 не входит в этот класс, поэтому закрывающая скобка - круглая ‘)‘
1 класс эквивалентности с 0% скидкой

[1; 2)
1 товар - только 1, граница 1 входит в этот класс - квадратная открывающая скобка ‘[’, граница 2 не входит в этот класс - круглая закрывающая скобка ‘)’
2 класс эквивалентности с 2% скидкой

[2; 5]
С 2х до 5ти товаров - это 2,3,4,5; границы 2 и 5 входят в этот класс - скобки квадратные ‘[’ и ‘]’

3 класс эквивалентности с 5% скидкой

(5; +∞)
Более 5 товаров - 6,7,...; граница 5 не входит в этот класс - круглая открывающая скобка ‘(‘
4 класс эквивалентности с 10% скидкой

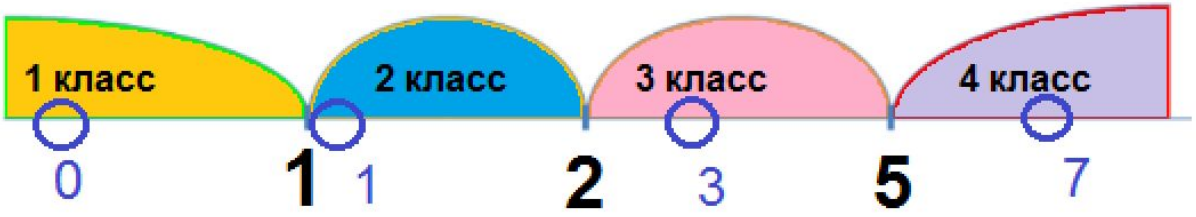
1. Применяем алгоритм использования техники классов эквивалентности:

1.1 Определить классы эквивалентности.

Определили 4 класса:

1: (- ∞; 1)
2: [1; 2)
3: [2; 5]
4: (5; +∞)

1. 2 Выбрать одного представителя от каждого класса. На этом шаге из каждого эквивалентного набора тестов мы выбираем один тест.



1: 0 товаров
2: 1 товар

3: 3 товара
4: 7 товаров

1. 3 Выполнить тесты. На этом шаге мы выполняем тесты от каждого класса эквивалентности.

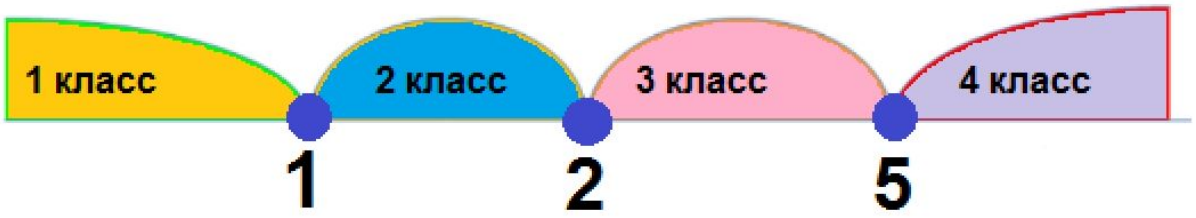
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек

По технике классов эквивалентности получаем всего 4 теста.

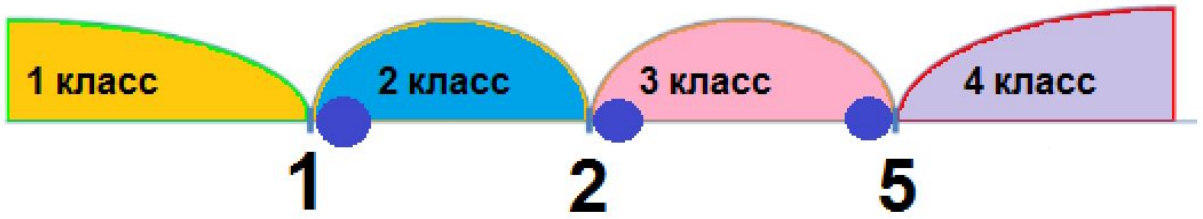
2. Применяем алгоритм использования техники граничных значений:

2.1 Во-первых, нужно выделить классы эквивалентности.
Определили 4 класса:
1: (- ∞; 1)
2: [1; 2)
3: [2; 5]
4: (5; +∞)

2.2 Далее нужно определить граничные значения этих классов.
Граничные значения классов:
1,2,5



2.3 Нам нужно понять, к какому классу будет относиться каждая граница.
1 - ко второму классу
2 - к 3му классу
5 - к 3му классу



2.4 Для каждой границы нам нужно провести тесты по проверке значения до границы, на границе, и сразу после границы.

Для границы 1: тесты с значениями 0,**1**,2

Для границы 2: тесты с значениями 1,**2**,3

Для границы 5: тесты с значениями 4,**5**,6

Количество тестов для проверки граничных значений будет равно количеству границ, умноженному на 3. Рекомендуется проверять значения вплотную к границе.

По технике тестирования граничных значений получаем всего 9 тестов.

Для границы 1:
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 2 акционных товара - скидка 5% на весь чек

Для границы 2:
4 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек

5 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
6 тест: Покупаем 3 акционных товара - скидка 5% на весь чек

Для границы 3:
7 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
8 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
9 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

В **данном конкретном примере** повторяющиеся тестовые случаи для граничных значений мы можем сократить, так как проверки полностью идентичны.

Было 9:
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
4 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
5 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
6 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
7 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
8 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
9 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

Стало 7:
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
6 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
7 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
8 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
9 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

В **данном конкретном случае** мы не потеряли ни одной проверки, просто убрали дубли.

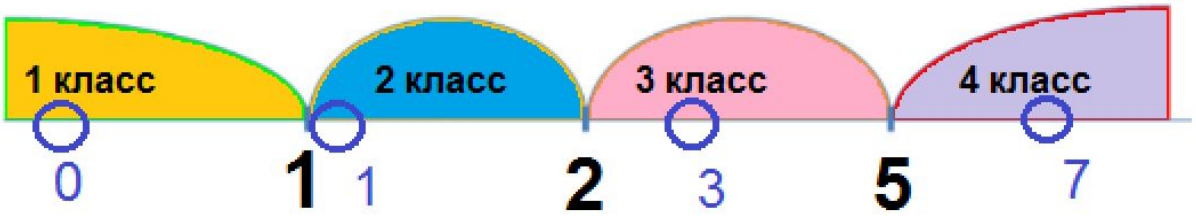
Перенумерация:
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 2 акционных товара - скидка 5% на весь чек

4 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
5 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
6 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
7 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

3. Использование классов эквивалентности и граничных условий вместе:

3.1 Техника анализа классов эквивалентности говорит о том, что мы должны выбрать минимум по одному значению из каждого класса.

В пункте 1.2 мы уже выбрали
Из 1-го класса берём представителя 0
Из 2-го класса берём представителя 1
Из 3-го класса берём представителя 3
Из 4-го класса берём представителя 7



Если суммировать тесты из примера, необходимые для проверки классов эквивалентности и граничных значений, получим 4 + 9 =13 тестов.
4, по классам эквивалентности:
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек

9, по граничным значениям:
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
4 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек

5 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
6 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
7 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
8 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
9 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

3.2 Так как граница обычно относится к какому-то классу, то можно использовать ее как представителя этого класса.

В пункте 2.3 определили, что границы
1 - относится ко второму классу
2 - относится к 3му классу
5 - относится к 3му классу

То есть мы можем взять границу 1 как представителя второго класса,
Границу 2 и границу 5 как представителей третьего класса.

Получается, что:
из 1-го класса берём представителя 0
из 2-го класса берём представителя 1
из 3-го класса берём представителя 2 или 5
Из 4-го класса берём представителя 7

Границу 2 мы берём как представителя третьего класса.

Было, по классам эквивалентности:
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем **3** акционных товара - скидка 5% на весь чек
4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек

Вместо представителя 3 берём 2 - оба значения относятся к третьему классу.

Стало, по классам эквивалентности:
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем **2** акционных товара - скидка 5% на весь чек
4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек

Выписываем все получившиеся тесты:

4, по классам эквивалентности:

1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек

9, по граничным значениям:

5 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
6 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
7 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
8 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
9 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
10 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
11 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
12 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
13 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

3.3. Убираем тесты по классам эквивалентности, идентичные тестам по граничным значениям

1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
3 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек
5 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
6 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
7 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
8 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
9 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
10 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
11 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
12 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
13 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

Стало:

4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек
5 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
6 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
7 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
8 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
9 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
10 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
11 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
12 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
13 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

В данном конкретном примере повторяющиеся тестовые случаи для граничных значений мы можем сократить, так как проверки полностью идентичны.

4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек
5 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
6 тест: Покупаем 1 акционный товар - скидка 2% на весь чек
7 тест: Покупаем 2 акционных товара - скидка 5% на весь чек
8 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
9 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
10 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
11 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
12 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
13 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

Стало:

4 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек
5 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
8 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
9 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
10 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
11 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
12 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
13 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

Итого у нас при комбинации техник классов эквивалентности и граничных значений 13 тестов (4+9) сократилось до 8:

1 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек
2 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%

3 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
4 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
5 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
6 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
7 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
8 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

В условиях нехватки времени и ограниченности ресурсов можно продолжить сокращать количество тестов:

Например 2, 3, 4, 5 товаров относятся к третьему классу эквивалентности, но 2 и 5 являются границами, их лучше оставить, но можно сократить на 3 и 4.
Например 6,7 товаров относятся к четвертому классу эквивалентности, но 6 является околограничным значением, его лучше оставить, но можно сократить на 7.

1 тест: Покупаем 7 акционных товаров - скидка 10% на весь чек
2 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
3 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
4 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
5 тест: Покупаем 3 акционных товара - скидка 5% на весь чек
6 тест: Покупаем 4 акционных товаров - скидка 5% на весь чек
7 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
8 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

Итого, минимальный набор тестов для приведённого примера будет выглядеть так:
2 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
3 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
4 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
7 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
8 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

Этим минимальным набором для описанного примера мы проверяем все классы эквивалентности (скидки в 0%, 2%, 5% и 10%) и проверяем каждую границу (1,2,5 товаров):
1 тест: Покупаем 0 акционных товаров - нет скидки на чек, 0%
2 тест: Покупаем 1 акционных товаров - скидка 2% на весь чек
3 тест: Покупаем 2 акционных товар - скидка 5% на весь чек
4 тест: Покупаем 5 акционных товар - скидка 5% на весь чек
5 тест: Покупаем 6 акционных товаров - скидка 10% на весь чек

НО! Сокращать тесты нужно на свой страх и риск. Если считаете, что необходимо проверить обычные (не граничные) или околограничные тесты – сделайте это. Лучше перестраховаться, чем пропустить ошибки.

5.3 Таблицы принятия решений (Decision table)

Таблица принятия решений/ Таблица альтернатив – разработка тестов методом черного ящика, описывает логику приложения основываясь на сущностях (свойствах/условиях) состояния системы.

❖ Условия/сущности (conditions) – это разные свойства системы, они представляют в таблице входные данные, которые можно ввести в систему.

❖ Действия (actions) – это действия которые могут произойти с указанной комбинацией сущностей, в зависимости от комбинации всех входных данных сущностей, действия принимают нужные значения.

❖ Каждое правило определяет уникальный набор входных данных всех свойств, которые приводят к исполнению конкретных действий .

		Правила (rules)							
Условия	Значения	1	2	3	4	5	6	7	8
Условие1	T,F	T	T	T	T	F	F	F	F
Условие2	T,F	T	T	F	F	T	T	F	F
Условие3	T,F	T	F	T	F	T	F	T	F
Действия									
Действие1		X	X			X			
Действие2		X					X		
Действие3			X			X	X		
Действие4				X	X			X	X

Таблицы решений могут использоваться для записи сложных бизнес-правил, которые должна реализовывать система.

❖ Таблица решений содержит условия, обычно комбинации значений «истина» и «ложь» для всех входных условий, и результирующие действия для каждой комбинации условий.

❖ Основана на анализе технического задания.

❖ Применяется на любых уровнях тестирования.

❖ Техника эффективна для определения комбинаций, которые в других случаях могут быть не замечены.

❖ Каждое условие предполагает ответ Да/Нет.

❖ Каждый вариант содержит различную комбинацию ответов, приводящую к разным действиям.

❖ Отдельный тест-кейс необходим для каждого столбика таблицы

Шаги построения таблицы:

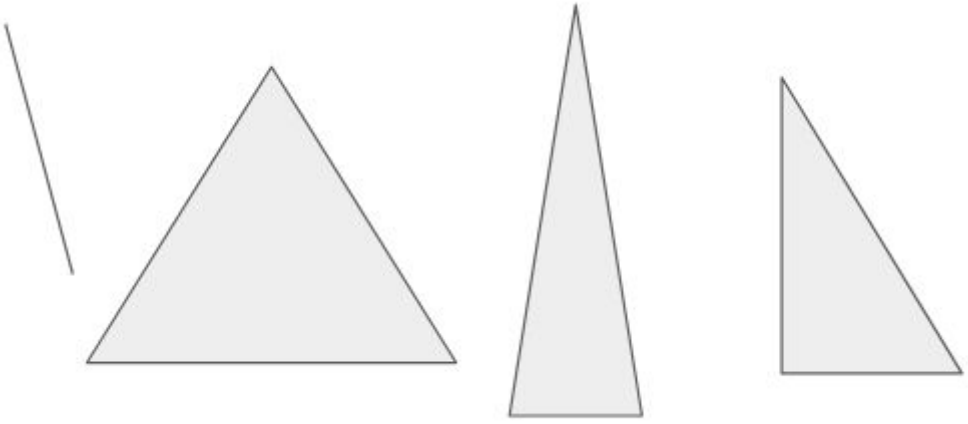
1. Определить/записать все условия
2. Подсчитать количество комбинаций условий:
   N = n1*n2*n3...n*m
3. Заполнить комбинации
4. Записать действия
5. Убрать лишние комбинации

(если таковые имеются)

Пример1:

Ввод: a,b,c

Вывод: определить тип треугольника (не треугольник, равносторонний, равнобедренный, разносторонний)



1. Определить/записать все условия

Условия:

Условие 1: a,b,c образуют треугольник? {Y,N}

Условие 2: a=b? {Y,N}

Условие 3: a=c? {Y,N}

Условие 4: b=c? {Y,N}

2. Подсчитать количество комбинаций условий

Все условия ({Y,N}): 4 условия

Кол-во комбинаций = У1 * У2 * У3 * У4

$N = 2^4 = 16$

3. Заполнить комбинации

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C1: Треугольник?	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
C2: a=b?	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
C3: a=c?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
C4: b=c?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N

4. Записать действия

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C1: Треугольник?	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
C2: a=b?	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
C3: a=c?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
C4: b=c?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N

Не треугольник																
Равносторонний																
Равнобедренный																
Разносторонний																
Не возможно																

5. Убрать лишние комбинации

	1	2	3	4	5	6	7	8	9
C1: Треугольник?	Y								N
C2: a=b?	Y				N				-
C3: a=c?	Y		N		Y		N		-
C4: b=c?	Y	N	Y	N	Y	N	Y	N	-
A1: Не треугольник									X
A2: Равносторонний	X								
A3: Равнобедренный				X		X	X		
A4: Разносторонний								X	
A5: Невозможно		X	X		X				

По данной методике сократили изначальное количество тестов 16 до 9.

Пример2:

1. Определить/записать все условия
2. Условия:

Горит красный? да/нет

Горит жёлтый? да/нет
Горит зелёный? да/нет

3. Подсчитать количество комбинаций условий

N = 2*2*2 = 8

4. Заполнить комбинации

Условия	Значения	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Горит красный?	Y, N	Y	Y	Y	Y	N	N	N	N
Горит желтый?	Y, N	Y	Y	N	N	Y	Y	N	N
Горит зеленый?	Y, N	Y	N	Y	N	Y	N	Y	N

5. Записать действия

Условия	Значения	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Горит красный?	Y, N	Y	Y	Y	Y	N	N	N	N
Горит желтый?	Y, N	Y	Y	N	N	Y	Y	N	N
Горит зеленый?	Y, N	Y	N	Y	N	Y	N	Y	N
Действия									
Ехать								X	
Стоять			X		X				
Готовиться			X						
Спец.действие		X		X		X	X		X

Желтый может мигать?
А красный? Зеленый?
Специальное действие - что это?
Все ли светофоры одинаковые?

Review

Условия	Значения	1	2	3	4	5	6	7	8	9	10	11	12
Горит красный?	Y, N	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N
Горит желтый?	Y, N, Blinking	Y	Y	N	N	B	B	Y	Y	N	N	B	B
Горит зеленый?	Y, N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Действия													
Ехать										X			
Стоять			X		X								
Готовиться			X										
Спец.действие		X		X		X	X	X	X		X	X	
Не регулируется													X

Количество комбинаций условий: красный (Y,N), жёлтый (Y,N,B), зелёный (Y,N), $N = 2*3*2 = 12$

6. Убрать лишние комбинации

Сворачивание таблицы принятия решений

Когда значение одного или нескольких конкретных условий не могут повлиять на действия для двух или более комбинаций условий, можно свернуть таблицу альтернатив.

1. Найти “одинаковые” и/или “невозможные” правила.
2. Убедиться, что они приводят к **одинаковым действиям**.
3. Поставить вместо условия “-” и свернуть лишние комбинации.

Прочерк обычно означает: не важно, не имеет значения или этого не может произойти при заданных условиях.

Сворачивание таблицы

Условия	Значения	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Горит красный?	Y, N	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N
Горит желтый?	Y, N, Blinking	Y	Y	N	N	B	B	Y	Y	N	N	B	B
Горит зеленый?	Y, N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Действия													
Ехать										X			
Стоять			X		X								
Готовиться			X										
Спец.действие		X		X		X	X	X	X		X	X	
Не регулируется													X

Сворачивание таблицы

Условия	Значения	**1**	**2**	**3**	**4**	**5**	**7**	**8**	**9**	**10**	**11**	**12**
Горит красный?	Y, N	Y	Y	Y	Y	Y	N	N	N	N	N	N
Горит желтый?	Y, N, Blinking	Y	Y	N	N	B	Y	Y	N	N	B	B
Горит зеленый?	Y, N	Y	N	Y	N	-	Y	N	Y	N	Y	N
Действия												
Ехать									X			
Стоять			X		X							
Готовиться			X									
Спец.действие		X		X		X	X	X		X	X	
Не регулируется												X

Аналогично можно “свернуть” столбцы 1 и 3, 7 и 8. С помощью этой техники, определили все тесты, которые можем выполнить и сократили количество тестов с 12 до 9.

Подсказки при составлении таблицы

1. Наиболее доминантные условия ставить сверху
2. Родственные комбинации располагать рядом
3. Условия с большим кол-вом значений снизу

Условия	Значения
Условие1 - родственное	Y, N
Условие2 - родственное	Y, N
Условие3	Y, N
Условие4 - мульти	A1, A2, A3, A4

Верхнее условие изменяется медленнее всего. Половина колонок Истина, половина - Ложь.
Второе сверху условие изменяется более быстро, но медленнее чем все остальные условия. Правило заполнения - четверть Истина, четверть Ложь, четверть Истина, четверть Ложь.
Нижнее условие изменяется быстрее всех - чередование Истина, Ложь, Истина, Ложь, и т.д.
Такой шаблон заполнения позволяет гарантировать, что ничего не будет пропущено.

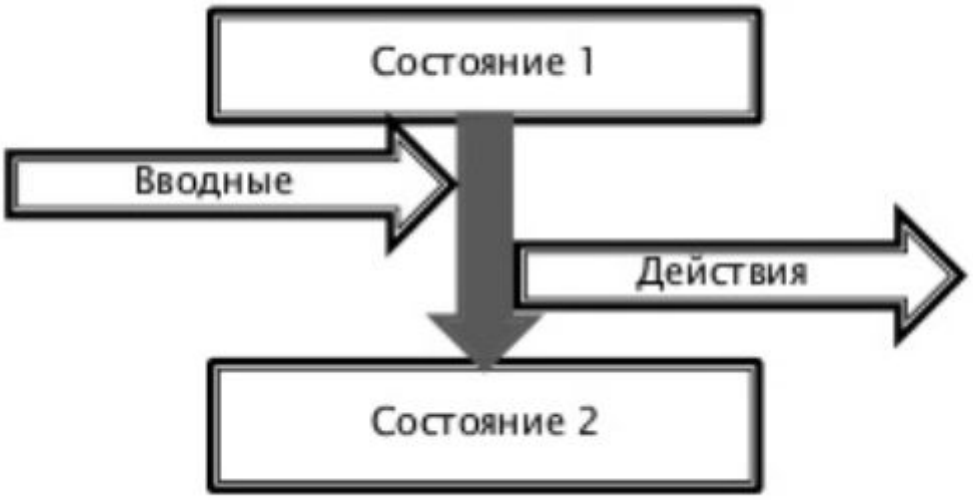
У1	y	y	y	y	n	n	n	n
У2	y	y	n	n	y	y	n	n
...								
УN	y	n	y	n	y	n	y	n

5.4 Тестирование переходов состояний (State transition testing)

Система может “реагировать” по-разному в зависимости от:

❖ Действий, выполненных пользователем;

❖ Состояния, в котором находилась система перед выполнением действия (или череды состояний)



Тестирование переходов состояний – это разработка тестов методом черного ящика, при котором сценарии тестирования строятся на основе выполнения корректных и некорректных переходов состояний. Диаграммы/таблицы определяют все события, которые возникают во время работы приложения, и как приложение реагирует на эти события может подсказать возможные некорректные переходы.

Система анализируется с точки зрения:

- ❖ возможности определения состояний;
- ❖ переходов между состояниями;

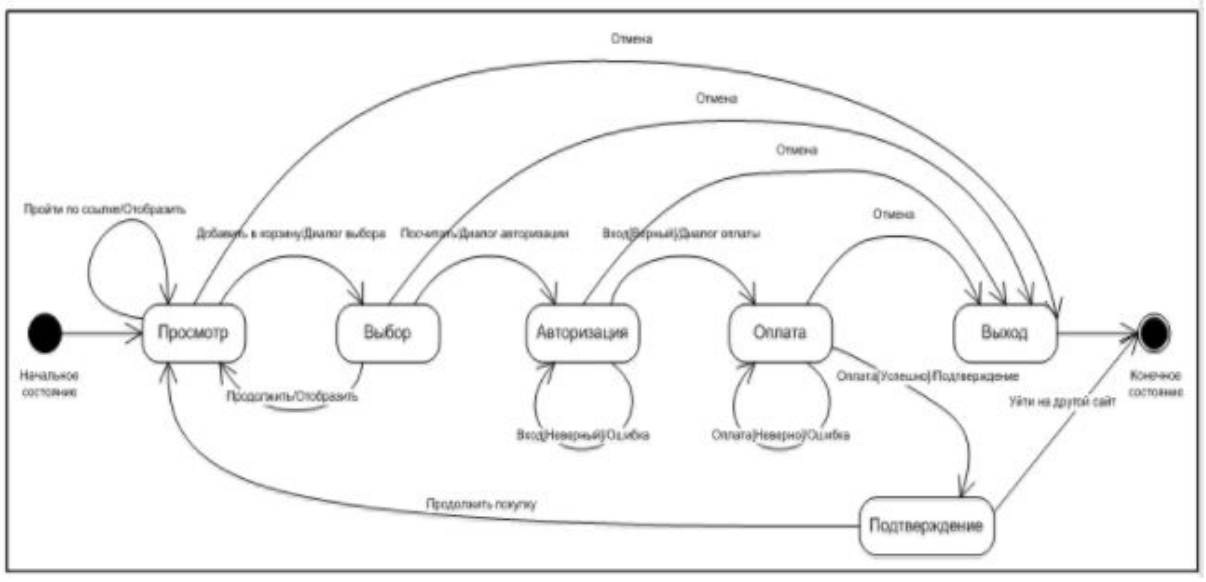
- ❖ условий;
- ❖ действий (результатов).

Таблица переходов состояний:

- ❖ Удобна для тестирования очередности действий;
- ❖ Эффективна для тестирования некорректных данных/действий;
- ❖ Может использоваться для тестирования диалоговых окон.

Два вида визуального представления этой техники:

- ❖ State Transition Diagrams (диаграммы)



- ❖ State Transition Tables (таблицы)

Состояния		События/Условия		Текущее состояние	Событие/Условие	Действие	Новое состояние
				Просмотр	Пройти по ссылке	Отобразить	Просмотр
		Пройти по ссылке		Просмотр	Добавить в корзину	Диалог выбора	Выбор
		Добавить в корзину		**Просмотр**	**Продолжить**	**Не определено**	**Не определено**
		Продолжить		**Просмотр**	**Выписать**	**Не определено**	**Не определено**
Просмотр		Выписать		**Просмотр**	**Вход [неверный]**	**Не определено**	**Не определено**
Выбор		Вход [неверный]		**Просмотр**	**Вход [верный]**	**Не определено**	**Не определено**
Авторизация	X	Вход [верный]	=	**Просмотр**	**Оплата [неверно]**	**Не определено**	**Не определено**
Оплата		Оплата [неверно]		**Просмотр**	**Оплата [успешно]**	**Не определено**	**Не определено**
Подтверждение		Оплата [успешно]		**Просмотр**	**Отмена**	**Нет действия**	**Выйти**
Выйти		Отмена		**Просмотр**	**Продолжить покупку**	**Не определено**	**Не определено**
		Продолжить покупку		**Просмотр**	**Уйти на другой сайт**	**Не определено**	**Не определено**
		Уйти на другой сайт		**Выбор**	**Пройти по ссылке**	**Не определено**	**Не определено**
				{53 строки, сгенерированных по шаблону, не показаны}			
				Выйти	**Уйти на другой сайт**	**Не определено**	**Не определено**

State Transition Tables имеет не такой наглядный вид как State Transition Diagrams , зато более полный и систематизированный.

State transition tables состоит из 4 столбцов:

- ❖ Текущее состояние (Current State),
- ❖ Событие (Event),
- ❖ Действие (Action)
- ❖ Следующее состояние (Next state).

Преимущество State Transition Tables в том, что они определяют все возможные State Transition варианты, а не только валидные.

Поэтому State Transition Tables часто приводят к нахождению неопределенных, недокументированных State Transition комбинаций, которые лучше находить перед написанием кода.

Состояние (state) - это состояние приложения, в котором оно ожидает 1 или более событий.

Состояние помнит входные данные полученные до этого и показывает как приложение будет реагировать на полученные события.

События могут вызывать смену состояния и/или инициировать действия.

Состояния должны быть:

- обособленными;
- определяемыми;
- конечными по количеству.

Система анализируется с точки зрения:

- возможности определения состояний;
- переходов между состояниями;
- условий;
- действий (результатов).

Переход (transition) (представленное в виде стрелки на диаграмме) - представляет переход одного состояния в другое, происходящий по событию.

Событие (event) (представленное ярлыком над/под стрелкой) - это что-то, что заставляет приложение поменять свое состояние. События могут поступать извне приложения, поступать через интерфейс приложения. Так же само приложение может генерировать события. Когда происходит событие приложение может поменять состояние или остаться в том же состоянии и/или выполнить действие. События могут иметь параметры, например событие "payMoney" может иметь параметры "Cash", "Check", "Debit Card", или "Credit Card".

Действие (action) (представлено после "/" в ярлыке над переходом) - это действие инициированное сменой состояния. Это может быть "напечатать билет", "показать на экране" и др. Обычно действия создают что-то, что является выходными/возвращаемыми данными системы. Действия возникают при переходах, сами по себе состояния пассивны.

Таблица переходов состояний:

- Удобна для тестирования очередности действий;
- Эффективна для тестирования некорректных данных/действий;
- Может использоваться для тестирования диалоговых окон.

Преимущества использования:

- ❖ Лучше проанализировать тестируемый продукт
- ❖ Систематизировать все знания по нему
- ❖ Получить готовые тест-кейсы
- ❖ Предугадывать ошибки

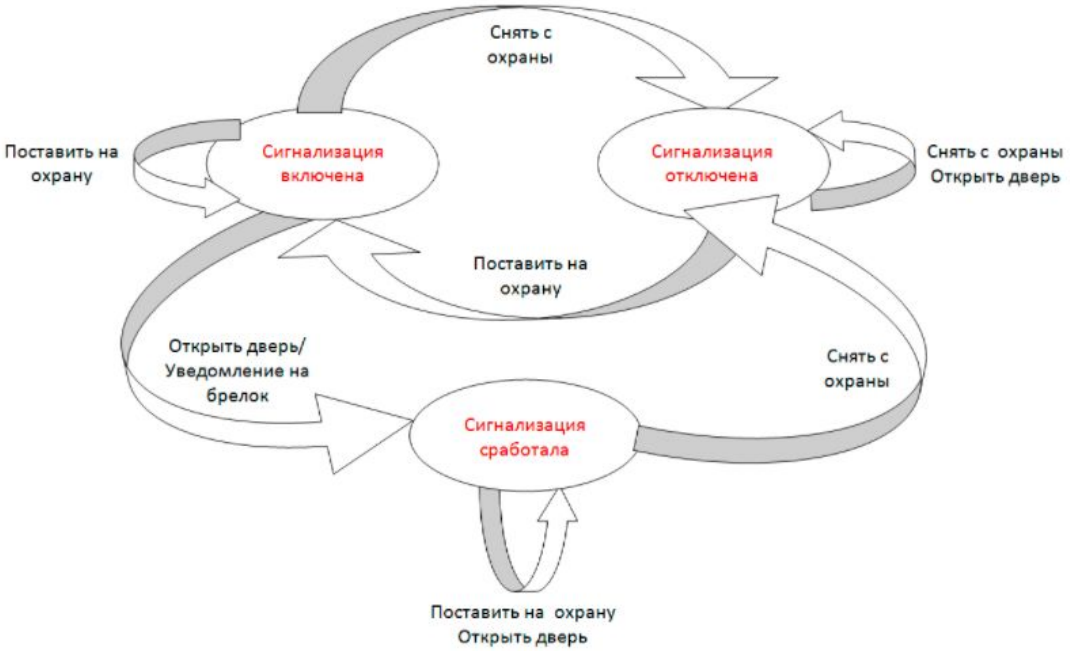
Таблицы позволяют:

- ❖ Лучше проанализировать тестируемый продукт
- ❖ Систематизировать все знания по нему
- ❖ Получить готовые тест-кейсы
- ❖ Предугадывать ошибки

Такие таблицы/диаграммы будут понятны всей проектной команде.

Пример1:

Сигнализация автомобиля


Снять с
охраны
Поставить на
охрану
Сигнализация
включена
Сигнализация
отключена
Снять с охраны
Открыть дверь
Поставить на
охрану
Открыть дверь/
Уведомление на
брелок
Снять с
охраны
Сигнализация
сработала
Поставить на охрану
Открыть дверь

Текущее состояние	Событие	Действие	Следующее состояние
Сигнализация отключена	Нажать на кнопку снять с охраны		Сигнализация отключена
Сигнализация отключена	Нажать на кнопку поставить на охрану охрану		Сигнализация включена
Сигнализация отключена	Открыть дверь		Сигнализация отключена
Сигнализация включена	Нажать на кнопку снять с охраны		Сигнализация отключена
Сигнализация включена	Нажать на кнопку поставить на охрану охрану		Сигнализация включена
Сигнализация включена	Открыть дверь	Уведомление на брелок	Сигнализация сработала
Сигнализация сработала	Нажать на кнопку снять с охраны		Сигнализация отключена
Сигнализация сработала	Нажать на кнопку поставить на охрану охрану		Сигнализация сработала
Сигнализация сработала	Открыть дверь		Сигнализация сработала

Тестирование из одного состояние в другое:

- Сигнализация отключена => сигнализация включена
- Сигнализация включена => сигнализация сработала

Тестирование переходов парами (транзитом):

- Сигнализация отключена => сигнализация включена => сигнализация сработала
- Сигнализация включена => сигнализация сработала => сигнализация отключена

Пример2:

Бронирование авиабилетов

- Мы как клиенты, предоставляем авиакомпании информацию для бронирования - место отправления, место прибытия, дату и время отправления.
- Служащий авиакомпании служит интерфейсом между нами и системой бронирования авиабилетов, и использует предоставленную нами информацию для создания брони.
- После этого наша бронь находится в состоянии "Made" (сделана).
- После создания брони, система бронирования, запускает таймер.
- Если таймер не истекает и мы оплатили зарезервированный билет, то система приобретает состояние "Paid" (оплаченный). Это показано стрелкой "payMoney" (заплатить деньги) и переходом из состояния "Made" в состояние "Paid".
- Из состояния "Paid" должен быть переход в состояние "Ticketed" (обилечен), когда билет будет напечатан и передан нам в руки. Обратите внимание, что при переходе в состояние "Ticketed", авиабилет (Ticket) является входными данными состояния.
- Из состояния "Ticketed" мы переходим в состояние "Used" (использованный), когда отдаем свой билет персоналу аэропорта, при посадке в самолет.
- После какого-то действия (сесть в самолет, например) путь диаграммы заканчивается символом мишени.
- Если бронь не оплачена по истечению таймера, то она

отменяется как не оплаченная.

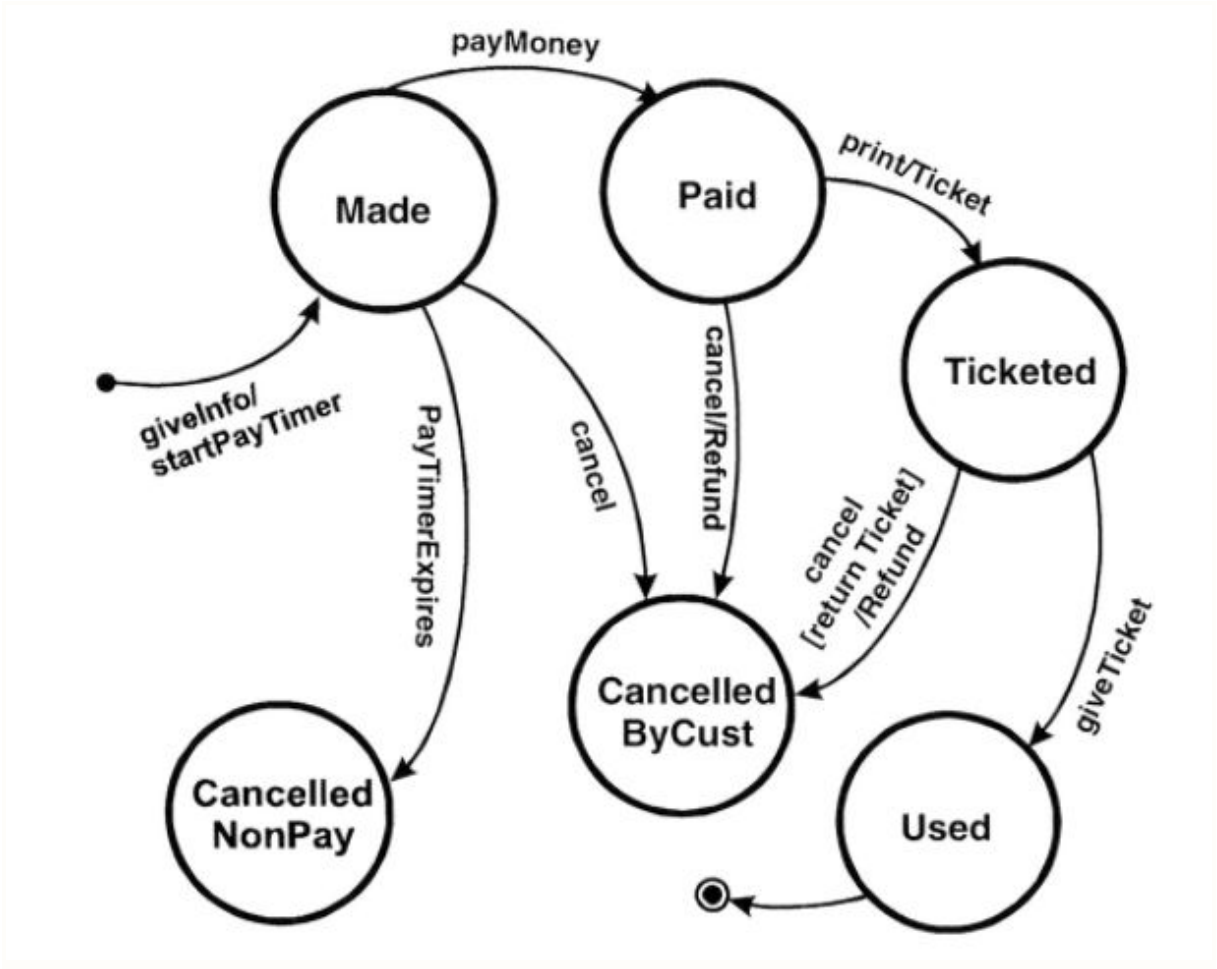
- ❖ Иногда клиенты отменяют заказы из состояния "Made". Для этого нам нужно еще 1 состояние "Cancelled By Customer".
- ❖ Так же, клиент может отменить бронь в состоянии "Paid". В этом случае состояние тоже становиться "Cancelled By Customer", и стоимость билета клиенту возмещают.
- ❖ Отменить бронь можно и из состояния "Ticketed".

В этом случае состояние опять становиться "Cancelled By Customer" и авиакомпания возмещает стоимость билета клиенту.

Но! Компания возместит стоимость билета только тогда, когда клиент вернет билет. Этот случай представляет еще один не описанный элемент диаграмм - квадратные скобки " [ ] ", которые представляют условие к переходу.

Переход в состояние в этом случае случится только если условие (то что в " [ ] ") = true.

Диаграмма


payMoney
Made
Paid
print/Ticket
Ticketed
giveInfo/
startPayTimer
PayTimerExpires
cancel
cancel/Refund
cancel
[return Ticket]
/Refund
giveTicket
Cancelled
ByCust
Cancelled
NonPay
Used

Таблица

Текущее состояние	Событие	Действие	Следующее состояние
null	***giveInfo***	***startPayTimer***	***Made***
null	payMoney	--	null
null	print	--	null
null	giveTicket	--	null
null	cancel	--	null
null	PayTimerExpires	--	null
Made	giveInfo	--	Made
Made	***payMoney***	**--**	***Paid***
Made	print	--	Made
Made	giveTicket	--	Made
Made	***cancel***	**--**	***Can-Cust***
Made	***PayTimerExpires***	**--**	***Can-NonPay***
Paid	giveInfo	--	Paid
Paid	payMoney	--	Paid
Paid	***print***	***Ticket***	***Ticketed***
Paid	giveTicket	--	Paid
Paid	***cancel***	***Refund***	***Can-Cust***
Paid	PayTimerExpires	--	Paid
Ticketed	giveInfo	--	Ticketed
Ticketed	payMoney	--	Ticketed
Ticketed	print	--	Ticketed
Ticketed	***giveTicket***	**--**	***Used***
Ticketed	***cancel***	***Refund***	***Can-Cust***
Ticketed	PayTimerExpires	--	Ticketed

Used	giveInfo	--	Used
Used	payMoney	--	Used
Used	print	--	Used
Used	giveTicket	--	Used
Used	cancel	--	Used
Used	PayTimerExpires	--	Used
Can-NonPay	giveInfo	--	Can-NonPay
Can-NonPay	payMoney	--	Can-NonPay
Can-NonPay	print	--	Can-NonPay
Can-NonPay	giveTicket	--	Can-NonPay
Can-NonPay	cancel	--	Can-NonPay
Can-NonPay	PayTimerExpires	--	Can-NonPay

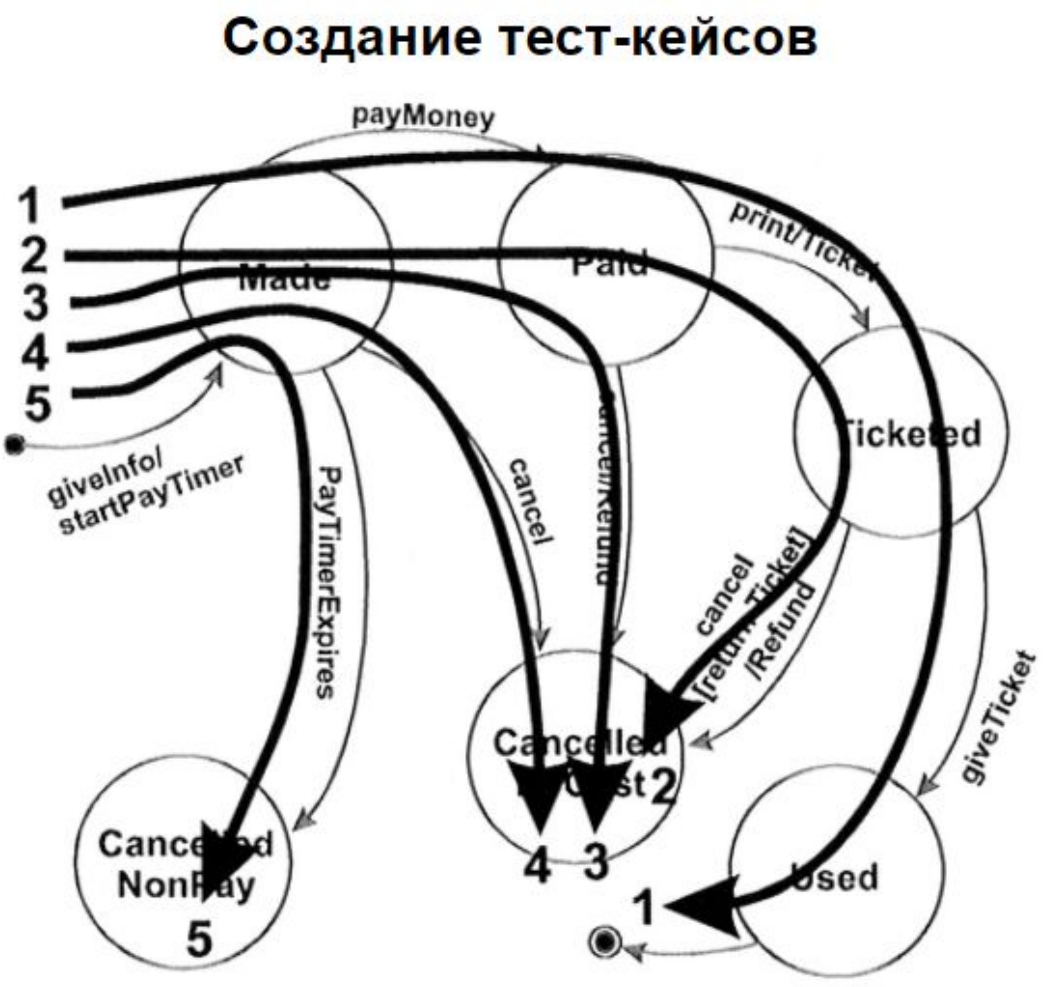
Can-Cust	giveInfo	--	Can-Cust
Can-Cust	payMoney	--	Can-Cust
Can-Cust	print	--	Can-Cust
Can-Cust	giveTicket	--	Can-Cust
Can-Cust	cancel	--	Can-Cust
Can-Cust	PayTimerExpires	--	Can-Cust

Создание тест-кейсов

- State Transition Diagrams могут быть легко использованы для создания тест кейсов. Необходимо создать набор тест-кейсов, который должен пройти по всем переходам хотя бы раз.
- Из State Transition Tables тоже достаточно легко делать тест кейсы. Стоит пройтись по всем валидным комбинациям (если есть время или не позволяют риски можно пройтись и по всем

невалидным комбинациям).

В таблице все валидные комбинации выделены жирным.



5.5 Тестирование по сценариям использования

Пользовательские сценарии

- ❖ Описывают поведение системы с точки зрения пользователя;
- ❖ Действия тестировщика имеют реальное значение для пользователя
- ❖ Могут иметь основные и “запасные” сценарии;

❖ Имеют предварительные условия и конечные результаты.

Пользовательские сценарии - сформулированы на ежедневном языке пользователя и содержат небольшие детали, оставаясь открытыми для интерпретации.

Тестирование по сценариям использования

❖ Тест-кейсы создаются на основе реальных сценариев использования системы;

❖ Найденные ошибки имеют большое значение для качества и огромны шансы быть обнаруженными пользователями;

❖ Чаще всего используется при приемочном тестировании.

Сценарии использования - описывают процесс и его шаги подробно, и могут быть сформулированы с точки зрения формальной модели, обеспечивает всю необходимую информацию и детали для понимания.

Пример:

Пользовательский сценарий:

Пользователь ищет необходимый товар, помещает его в корзину. Приступает к оформлению: указывает адрес доставки. Переходит к оплате товара.

Сценарий использования:

Пользователь в строку поиска вводит наименование товара, нажимает кнопку “Найти”. Помещает товар в корзину по иконке + , расположенной рядом с найденным товаром. Переход в корзину, кликнув по изображению или надписи “Корзина” в верхнем правом углу. В корзине пользователь переходит к оформлению товара, в поле “Адрес” указывает адрес доставки, заполняет контактные данные в соответствующих полях. Переходит к оплате товара, по кнопке “Оплатить”. Переход на страницу оплаты, пользователь вводит данные.

5.6 Предугадываение ошибок (Error Guessing - EG)

Предугадывание ошибки - используем знания для того, чтобы "предугадать" при каких входных условиях система может выдать ошибку.

Использование знаний для того, чтобы предугадать при каких входных условиях система может выдать ошибку.

Пример:

Например, спецификация говорит:

"пользователь должен ввести код 4 цифры"

Вы должны думать:

Что, если я не введу код?

Что, если я введу спецсимволы?

Введу не цифры?

Не 4 цифры, а 1 или 8?

…

Преимущества

- Эффективен в качестве дополнения к более формальным техникам
- Выявление тестовых случаев, которые “никогда не должны случиться”
-

Недостатки

- Данный метод в значительной степени основан на интуиции
- Необходим опыт в тестировании схожих систем
- Прямая зависимость от опыта тестировщика
- Малое покрытие тестами

Тест-дизайн:

- Тестировщик с большим опытом может выискивать ошибки без сценариев.
- Источники: собственные знания и опыт, знания и опыт других людей, спецификации.
- Необходимо составить список, который перечисляет возможные ошибки и ситуации, в которых эти ошибки могли/могут проявиться. Потом на основе списка составляются тесты.

Пример:

Строка с вводом данных на форме.

Проверки, для выявления багов:

- Ввод пустой строки (если поле обязательное)
- Пробел
- Кавычки одинарные
- Ограничение на количество символов - вставка “Война и мир” в строку
- И тд

5.7 Техника причина/следствие Cause/Effect - CE

Причина/Следствие - это Ввод комбинаций условий (причина: входное условие или класс эквивалентности), для получения ответа от системы (следствие: выходное условие или преобразование системы).

Ввод комбинаций условий
(причина - входное условие или класс эквивалентности), для получения ответа от системы (следствие - выходное условие или преобразование системы).

Анализ причинно-следственных связей позволяет:

- системно выбирать высокорезультативные тесты (минимум тестов находят максимум багов)
- соотнести все следствия ко всем причинам (нет пропущенных вариантов поведения)
- позволяет обнаруживать неполноту и неоднозначность исходных спецификаций (улучшение тестовой документации)

Недостатки:

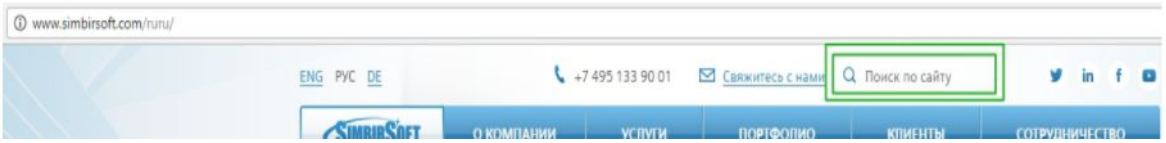
- Нужно время на анализ всех причинно-следственных связей, в том числе “невозможных”
- Возможно слишком большое количество комбинаций причин и следствий

Тест-дизайн:

1. Выделяем причины и следствия в спецификациях.
2. Связываем причины и следствия.
3. Учитываем “невозможные” сочетания причин и следствий.
4. Фиксируем «таблицу решений», где в каждой строке конкретная комбинация входов и выходов, каждая строка - готовый тестовый сценарий.
5. Расставляем приоритеты.

Пример:

Строка поиска.



1. Выделяем причины и следствия в спецификациях.
2. Связываем причины и следствия.

Набор причин:	**Набор следствий:**
Поиск символьной строки	Результаты поиска отсутствуют
Поиск символьной строки	Несколько результатов поиска
Поиск символьной строки	Один результат поиска
Поиск символьной строки	Сообщение что введённые символы некорректны
Поиск пустой строки	Сообщение, что строка поиска пуста
...	...

3. Учитываем “невозможные” сочетания причин и следствий.

4. Фиксируем «таблицу решений», где в каждой строке конкретная комбинация входов и выходов, каждая строка - готовый тестовый сценарий.

5. Расставляем приоритеты.

Набор причин:	**Набор следствий:**
1 Поиск символьной строки	Результаты поиска отсутствуют
1 Поиск символьной строки	Несколько результатов поиска
2 Поиск символьной строки	Один результат поиска
2 Поиск символьной строки	Сообщение что введённые символы некорректны
2 Поиск пустой строки	Сообщение, что строка поиска пуста
2 Поиск символьной строки	Сообщение, что строка поиска пуста
Поиск пустой строки	Результаты поиска отсутствуют
Поиск пустой строки	Несколько результатов поиска
Поиск пустой строки	Один результат поиска
Поиск пустой строки	Сообщение что введённые символы некорректны

Получили 6 тест-кейсов.

5.8 Исследовательское тестирование

Исследовательское тестирование - это тестирование без предварительно подготовленных сценариев.
(EXPLORATORY TESTING - ET)

Тестирование без предварительно подготовленных

сценариев, оно же Тестирование Свободным Поиском.

Одновременно происходит

Разработка тестов, выполнение тестов и обучение.

Не путать с Ad Hoc и Monkey тестированием.

Ad hoc - интуитивное тестирование в полном Хаосе, когда нет спецификации, нет плана, нет процесса. Выполняется без подготовки, без определения ожидаемых результатов, без проектирования тестовых сценариев. Тестировщик просто идет и проверяет, что ему хочется, нет определенной цели, структуры тестов, системы. Выполняется в условиях нехватки времени.

Ad hoc = Exploratory - мозги и мастерство

Monkey testing — произвольное и беспорядочное тестирование программы с целью ее сломать, используется как часть стандартных средств стресс-тестирования.

Ad Hoc тестирование VS Исследовательское

Преимущества:

- Эффективно работает в условиях неопределенности.
- Исследовательское тестирование используется в тех случаях, когда выполнение следующего теста неочевидно, или когда вы хотите выйти за рамки очевидного.
- Исследовательское тестирование ограничивается только широтой и глубиной фантазии тестеровщика, а также пониманием природы тестируемого приложения.
- **Находит ошибки, так как проверки по тестовым сценариям могут лишь подтвердить их отсутствие.**

Недостатки:

1. При исследовательском тестировании нет возможности предотвращать появление дефектов, из-за того, что проектирование формализованных тест кейсов не начинается на стадии сбора требований.
2. Исследовательские тесты, в отличие от сценарных тестов, не определены заранее и не выполняются в точном соответствии с планом.
3. Тяжесть планирования: тест-кейсы не легко поделить между различными тестировщиками или командами.
4. Важные кейсы могут остаться не пройденными.
5. Сложно оценить процент покрытия проекта тестированием и понять, какая часть уже протестирована.

Тест-дизайн:

1. Изучаем продукт, разрабатываем и выполняем тесты одновременно.
2. Записываем идеи тестов и используем их в дальнейшем.
3. Если времени на прохождение тестов нет, нужно выбирать наиболее критичные области приложения, которые реально протестировать за имеющееся время.
4. Записываем выполненные тест-кейсы для того, чтобы иметь возможность воспроизводить найденные

ошибки и определять тестовое покрытие приложения. **Нет времени на кейсы - пишем чек-лист**.

5. Сценарное и исследовательское тестирование являются полностью совместимыми и компенсируют недостатки друг друга. Особенно, если хочется убедиться в том, что ничего не было упущено.

Примеры:

- когда требований нет или они часто меняются,
- когда отдельные части программы уже работают, но некоторые еще не реализованы,
- когда тестировщик осваивает новую предметную область или новый вид тестирования,
- когда все остальные подходы уже исчерпали себя, а пользователи почему-то все равно не считают продукт идеальным,
- нужно обеспечить быструю обратную связь о новом продукте или фиче,
- нужно быстро изучить продукт,
- проверка работы другого тестировщика.

5.9 Исчерпывающее тестирование

Исчерпывающее тестирование - проверка всех возможных комбинаций входных значений.

(Exhaustive Testing - ET)

Проверка всех возможных комбинаций входных значений и предусловий.

Считается недостижимым.

Преимущества и недостатки:

- В пределах этой техники нужно проверить все возможные комбинации входных значений, и это должно найти все проблемы.
- Обеспечивает обнаружение всех возможных ошибок, но практически нереализуемо даже для очень небольших программ.

Примеры:

Доступно проверить все возможные варианты

- Radiobutton
- CheckBox
- Выпадающий список
- Одна / две / три кнопки в интерфейсе
- Строго определённое тестовое окружение

Тест-дизайн:

- Выписываем все варианты
- Выписываем все условия
- Проверяем все комбинации вариантов и условий

5.10 Парное

PAIRWISE - методика тестирования, основанная на ПРЕДПОЛОЖЕНИИ, что большинство дефектов возникает при взаимодействии не более чем ДВУХ факторов.

Представьте себе, что вам нужно протестировать систему с большим числом параметров, влияющих на её работу. Ярким примером такого рода может быть конфигурационное тестирование: например проверка работы системы под различными операционными системами или работа сайта в различных браузерах. Кто знает, какое сочетание параметров приведет к сбою? Каждый тестировщик знает, что все комбинации не проверить. К примеру, для проверки всех сочетаний 10 параметров с 10 значениями каждый, потребуется 10,000,000,000 тестов, в то время как метод перебора пар позволяет реализовать сравнимое по качеству тестирование (учитывая количество и

критичность найденных в результате багов) используя всего 177 тестов.

D. Richard Kuhn, Senior Member, IEEE, Dolores R. Wallace, Member, IEEE Computer Society опубликовали исследование, проводимое IEEE для ПО отозванных в течении 15 лет мед. устройств.
Вывод: 98% дефектов возникают при конфликте ПАР входных данных или ОДНОГО входного параметра (что PAIRWISE TESTING также покрывает).

Метод парного тестирования основан на довольно простой, но от того не менее эффективной идее, что подавляющее большинство багов выявляется тестом, проверяющим один параметр, либо сочетание двух. Ошибки, причиной которых явились комбинации трех и более параметров как правило значительно менее критичны, чем пары параметров и тем более одного, не говоря уже о том что никто не мешает дополнить свое тестовое покрытие кейсами на желаемые комбинации параметров.

Перебрать все пары немудрено, трудность в том, чтобы обеспечить при этом минимум тестов, комбинируя проверки нескольких пар в одном тесте.

Важно: Перед применением этого метода обязательно нужно применить классы эквивалентности и работать с ними.

Преимущества и недостатки:

- Хорошо масштабируется при увеличении количества параметров
- Уменьшает количество тест-кейсов
- Требуется намного меньше сил в сравнении с другими тактиками выполнения функционального тестирования
- Количество багов, обнаруженных с помощью попарного тестирования будет больше, чем при проверке всех значений для каждого параметра ввода в отдельности.
- Плохо масштабируется при увеличении количества значений для параметра

КОГДА ПРИМЕНЯТЬ?
1. Если есть параметры, зависимые между собой
2. Если очень большое количество параметров с малым количеством значений.

Составление нужных комбинаций данных - задачка часто не самая простая, но, к счастью, для её решения существует много инструментов, разного уровня качества и (бес)платности: **http://www.pairwise.org/tools.asp**

КАК ВСЕ ИСПОРТИТЬ?
•Определили неверные входные значения
•Что-то не учли
•Пропустили самые популярные комбинации
•Упустили взаимодействие переменных

ЧТО НУЖНО?

•Предварительно оптимизировать входные данные
•Исключить негативные кейсы
•Использовать техники тест-дизайна

Тест-дизайн:

1. Сбор входных данных (переменные и их параметры)
2. Построение таблицы всех комбинаций
3. Оптимизация данных
4. Прохождение теста

Техника формирования набора тестовых данных:
3 параметра с 3 значениями для каждого параметра
3*3*3 = 27 тест-кейсов
5 параметров с 3 значениями для каждого параметра
3*3*3*3*3= 243 тест-кейсов
7 параметров с 7 значениями для каждого параметра
7*7*7*7*7*7*7= 823 543 тест-кейсов

ГДЕ ИСПОЛЬЗУЕМ?
•Кросс-браузерное, кросс-платформенное тестирование
•Входные данные для автоматизированного тестирования

КОГДА ИСПОЛЬЗУЕМ?
•Когда тестируемый функционал стабилен
•Когда по отдельности это все работает

Пример:

Требуется тест приложения на ОС Win и Mac OS, браузеры FF и Chrom последних версий. Языки приложения en, ru, fr.

Количество комбинаций будет равно: 2*2*3 = 12
Составлям таблицу, со всеми возможными комбинациями:

Windows	Firefox	ru
Windows	Firefox	fr
Windows	Firefox	en
Windows	Chrome	ru
Windows	Chrome	fr
Windows	Chrome	en
Mac OS	Firefox	ru
Mac OS	Firefox	fr
Mac OS	Firefox	en
Mac OS	Chrome	ru
Mac OS	Chrome	fr
Mac OS	Chrome	en

Сокращаем количество проверяемых комбинаций, таким образом, чтобы в тест попали уникальные пары значений параметров.

Сначала таблицу разворачиваем в пары и отбираем комбинации, сверяя с таблицей пар:

Windows	Firefox
Windows	Chrome
Mac OS	Firefox
Mac OS	Chrome
Firefox	ru
Firefox	fr
Firefox	en
Chrome	ru
Chrome	fr
Chrome	en
Windows	ru
Windows	fr
Windows	en
Mac OS	ru
Mac OS	fr
Mac OS	en

Выбираем такие комбинации ОС+Браузер+язык, покрывающие все пары:

1

пары					
Windows	Firefox		Window	Firefox	ru
Windows	Chrome				
Mac OS	Firefox				
Mac OS	Chrome				
Firefox	ru				
Firefox	fr				
Firefox	en				
Chrome	ru				
Chrome	fr				
Chrome	en				
Windows	ru				
Windows	fr				
Windows	en				
Mac OS	ru				
Mac OS	fr				
Mac OS	en				

2

пары					
Window	Firefox		Windows	Firefox	ru
Window	Chrome		Windows	Chrome	fr
Mac OS	Firefox				
Mac OS	Chrome				
Firefox	ru				
Firefox	fr				
Firefox	en				
Chrome	ru				
Chrome	fr				
Chrome	en				
Window	ru				
Window	fr				
Window	en				
Mac OS	ru				
Mac OS	fr				
Mac OS	en				

3

пары				
Window	Firefox	Windows	Firefox	ru
Window	Chrome	Windows	Chrome	fr
Mac OS	Firefox	Mac OS	Firefox	en
Mac OS	Chrome			
Firefox	ru			
Firefox	fr			
Firefox	en			
Chrome	ru			
Chrome	fr			
Chrome	en			
Window	ru			
Window	fr			
Window	en			
Mac OS	ru			
Mac OS	fr			
Mac OS	en			

4

пары				
Window	Firefox	Windows	Firefox	ru
Window	Chrome	Windows	Chrome	fr
Mac OS	Firefox	Mac OS	Firefox	en
Mac OS	Chrome	Mac OS	Chrome	ru
Firefox	ru			
Firefox	fr			
Firefox	en			
Chrome	ru			
Chrome	fr			
Chrome	en			
Window	ru			
Window	fr			
Window	en			
Mac OS	ru			
Mac OS	fr			
Mac OS	en			

5

пары				
Window	Firefox		Windows	Firefox ru
Window	Chrome		Windows	Chrome fr
Mac OS	Firefox		Mac OS	Firefox en
Mac OS	Chrome		Mac OS	Chrome ru
Firefox	ru		Windows	Chrome en
Firefox	fr			
Firefox	en			
Chrome	ru			
Chrome	fr			
Chrome	en			
Window	ru			
Window	fr			
Window	en			
Mac OS	ru			
Mac OS	fr			
Mac OS	en			

6

пары				
Window	Firefox		Windows	Firefox ru
Window	Chrome		Windows	Chrome fr
Mac OS	Firefox		Mac OS	Firefox en
Mac OS	Chrome		Mac OS	Chrome ru
Firefox	ru		Windows	Chrome en
Firefox	fr		Mac OS	Firefox fr
Firefox	en			
Chrome	ru			
Chrome	fr			
Chrome	en			
Window	ru			
Window	fr			
Window	en			
Mac OS	ru			
Mac OS	fr			
Mac OS	en			

6 комбинациями покрыли все возможные пары, итоговые комбинации для теста:

Windows	Firefox	ru
Windows	Chrome	fr
Mac OS	Firefox	en
Mac OS	Chrome	ru
Windows	Chrome	en
Mac OS	Firefox	fr

И таким образом, мы можем получить гораздо меньше наборов значений (в них

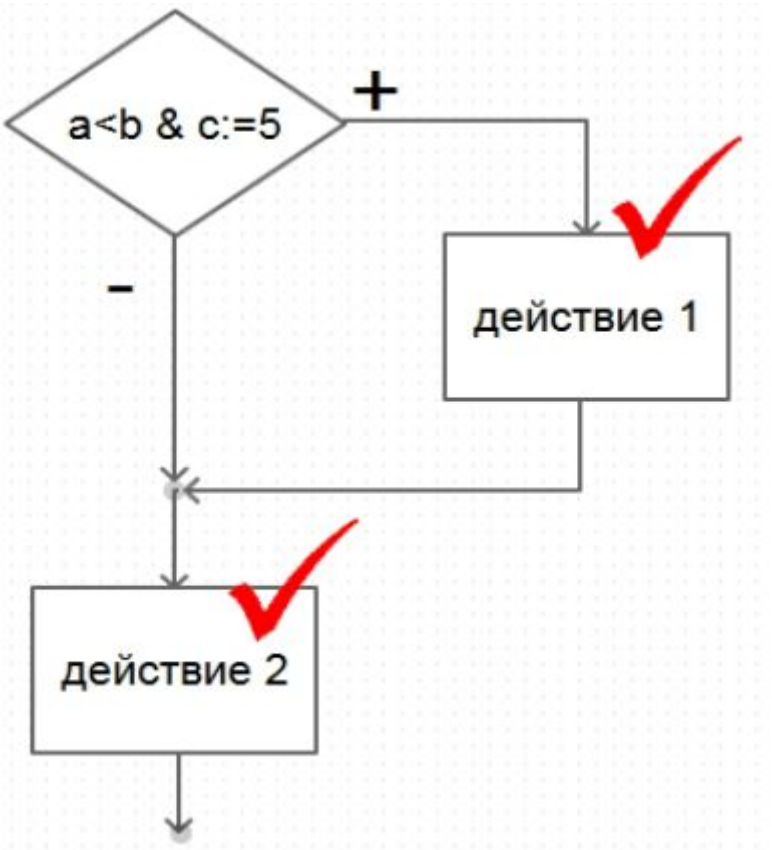
есть все пары значений, правда некоторые дважды)
Такой подход примерно и составляет суть техники pairwise testing - мы не проверяем все сочетания всех значений, но проверяем все пары значений.

6. Методики белого ящика

6.1 Покрытие операторов (statement coverage)

Покрытие операторов (statement coverage) – процентное отношение операторов, исполняемых набором тестов, к их общему количеству. В тесте должен быть задействован каждый оператор кода хотя бы раз.

Пример:
Строки
if (a<b & c=5) then Действие 1
Действие 2



Варианты:

- a<b; c=5 -> Действие 1 и Действие 2
- a>=b; c=5 -> Действие 2
- a<b; c<>5 -> Действие 2
- a>=b; c<>5 -> Действие 2

6.2 Покрытие решений/альтернатив (decision coverage)

Покрытие решений/альтернатив (decision coverage) - процент результатов альтернативы, который был проверен набором тестов. Стопроцентное покрытие решений подразумевает стопроцентное покрытие операторов. Если условие имеет два варианта (ИСТИНА или ЛОЖЬ)

решения, оно должно быть выполнено по разу для каждого случая.
Покрытие ветвей (branch coverage) - процент ветвей, которые были выполнены набором тестов. 100% покрытие ветвей подразумевает 100% покрытие альтернатив и 100% покрытие операторов. Проверка для каждого условия и ответвления.

Покрытие решений/альтернатив включает в себя Покрытие ветвей, а Покрытие ветвей включает в себя Покрытие решений/альтернатив. Таким образом Покрытие решений/альтернатив = Покрытие ветвей.

6.3 Покрытие ветвей (branch coverage)

Покрытие ветвей (branch coverage) – процент ветвей, которые были выполнены набором тестов. 100% покрытие ветвей подразумевает 100% покрытие альтернатив и 100% покрытие операторов.

Пример:
Строки
if (a<b & c=5) then Действие 1
Действие 2

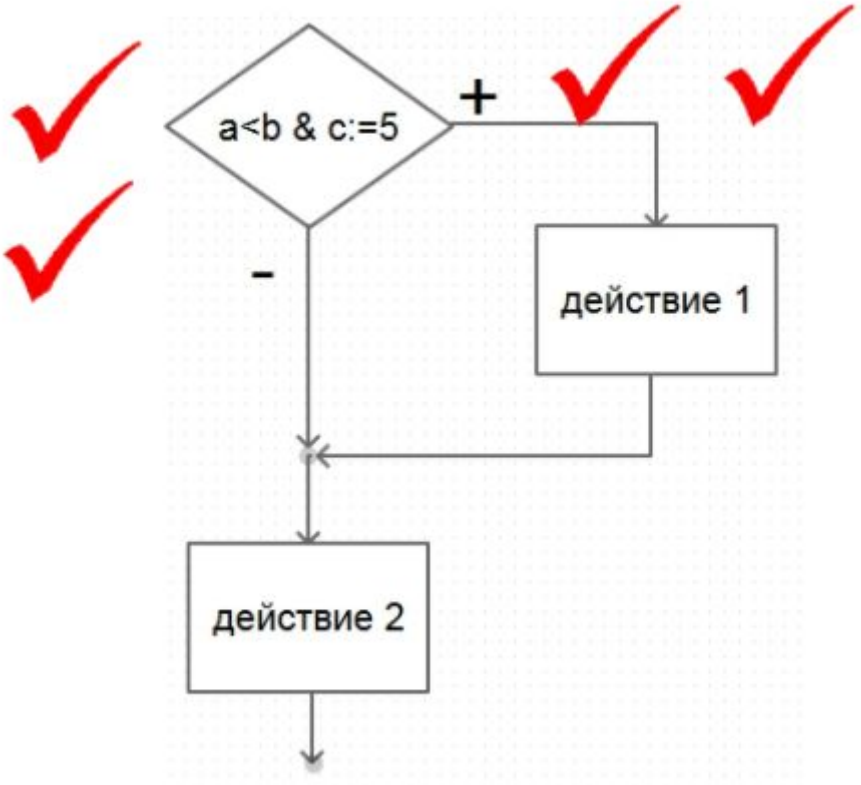
Варианты:

- a<b; c=5 -> Действие 1 и Действие 2
- a>=b; c=5 -> Действие 2
- a<b; c<>5 -> Действие 2
- a>=b; c<>5 -> Действие 2

6.4 Покрытие условий (condition coverage)

Покрытие условий (condition coverage) – процент исходов условий, которые были проверены набором тестов. 100% покрытие условий требует, чтобы каждое отдельное условие в каждом выражении решения было проверено как “Истина” и “Ложь”.

Пример1:
Строки
if (a<b & c=5) then Действие 1
Действие 2



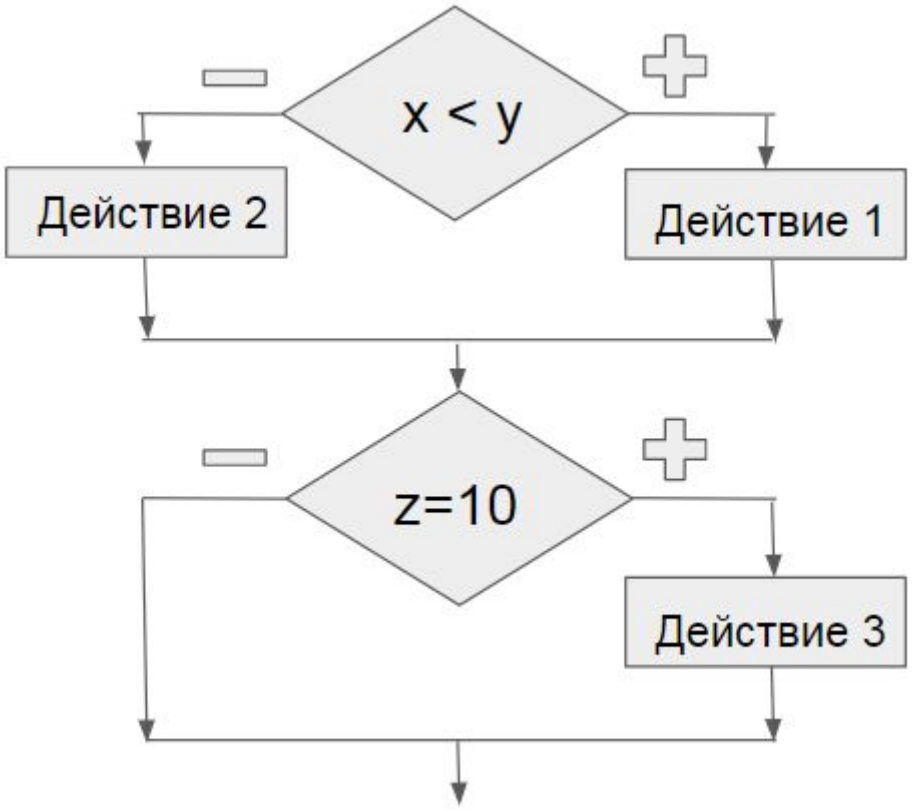
Варианты:

- a<b; c=5 -> Действие 1 и Действие 2
- a>=b; c=5 -> Действие 2
- a<b; c<>5 -> Действие 2
- a>=b; c<>5 -> Действие 2

Пример2:

Строки

if (x<y) then Действие 1 else Действие 2
if (z=10) then Действие 3



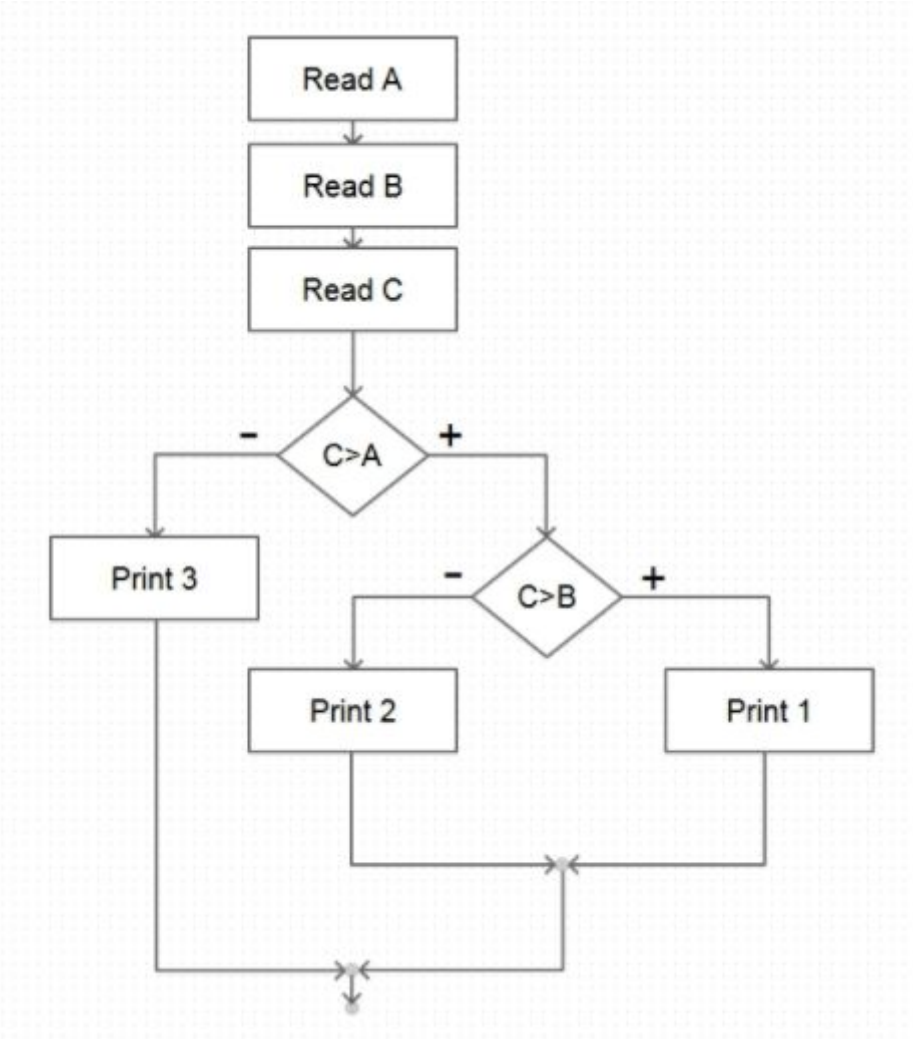
Варианты:

- x<y; z=10 -> Действие 1 и Действие 3
- x>=y;z=10 -> Действие 2 и Действие 3
- x<y; z<>10 -> Действие 1
- x>=y;z<>10 -> Действие 2

Пример для операторов, ветвей, условий:

Какое минимальное количество тест-кейсов необходимо выполнить для 100% покрытия операторов, решений, условий?

```
READ A
READ B
READ C
IF C>A THEN
IF C>B THEN
PRINT "C must be smaller than at least one number"
ELSE
PRINT "Proceed to next stage"
ENDIF
ELSE
PRINT "B can be smaller than C"
ENDIF
```



Минимальное кол-во тест-кейсов для:
100% покрытия операторов - 3
100% покрытие решений - 3
100% покрытие условий - 4
Для покрытия условий:
C > A, C > B
C > A, C ≤ B
C ≤ A, C > B
C ≤ A, C ≤ B