

Техники тест дизайн

Sumy Educational Center
Software Quality Assurance (QA)

Вы узнаете о следующем

- Что такое тест дизайн? Место тест дизайна в тестировании ПО
- Техники тест дизайна
 - Statement coverage
 - Condition coverage
 - Decision coverage
 - Finite state machines
 - Brute force
 - Equivalence class partitioning
 - Boundary value analysis
 - Pairwise testing

Что такое тестирование (повторение)?

Суть тестирования – проверка соответствия фактического поведения ПО ожидаемому поведению.

Дефект – несоответствие фактического поведения ПО ожидаемому поведению.

Процесс исполнения тестирования:

- узнать ожидаемый результат (expected result, ER);
- получить фактический результат (actual result, AR);
- сравнить фактический результат с ожидаемым (ER passed или ER failed?).

Test Case Step = Test Case (TC) = Test = Проверка:

- сформулирован ER;
- сформулирован алгоритм получения AR;
- сравнены ER и AR.

Место test design в тестировании



Test design – этап процесса тестирования ПО, на котором проектируются и создаются тест-кейсы.

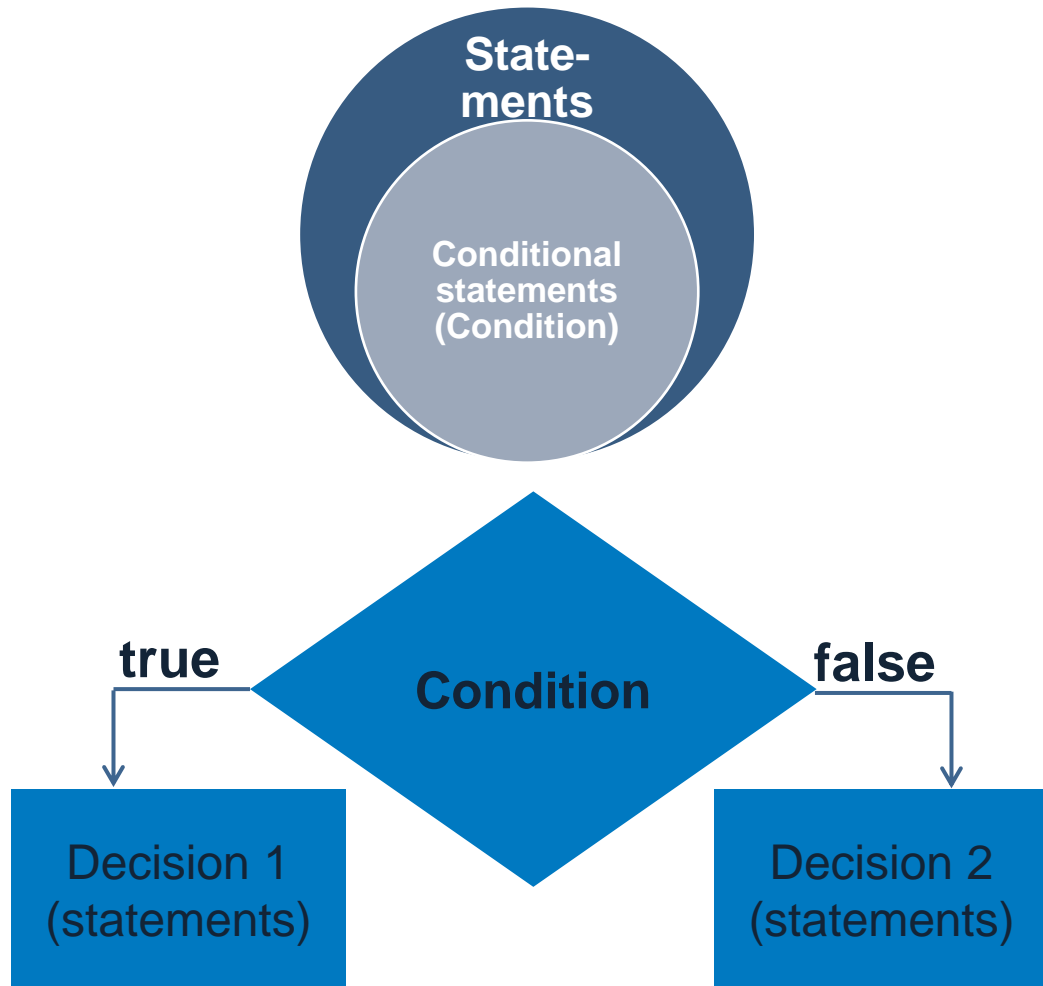


- Как получать AR, соответствующие ER?
- Как эффективно получать AR, соответствующие ER?
- Как присваивать приоритеты тест-кейсам?

1

Техники для тестирования ПО в динамике (динамические техники)

Введение терминов



Statement – любой оператор на языке исходного кода программы или любое действие программы, описанное в design specification

Conditional statement (condition) – условный оператор на языке исходного кода программы или любое условное действие программы, описанное в design specification

Пример

GUI:

Величина P =

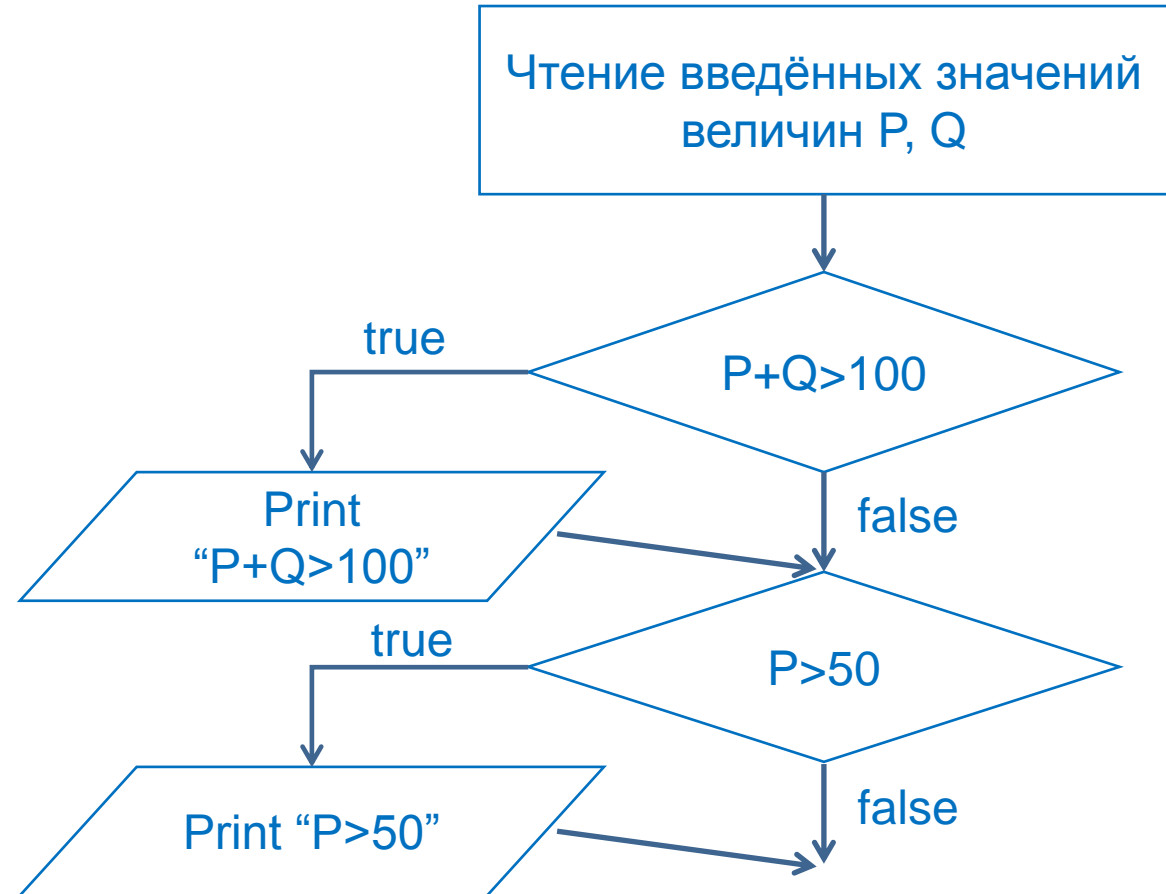
Величина Q =

Результат: ?

Source code:

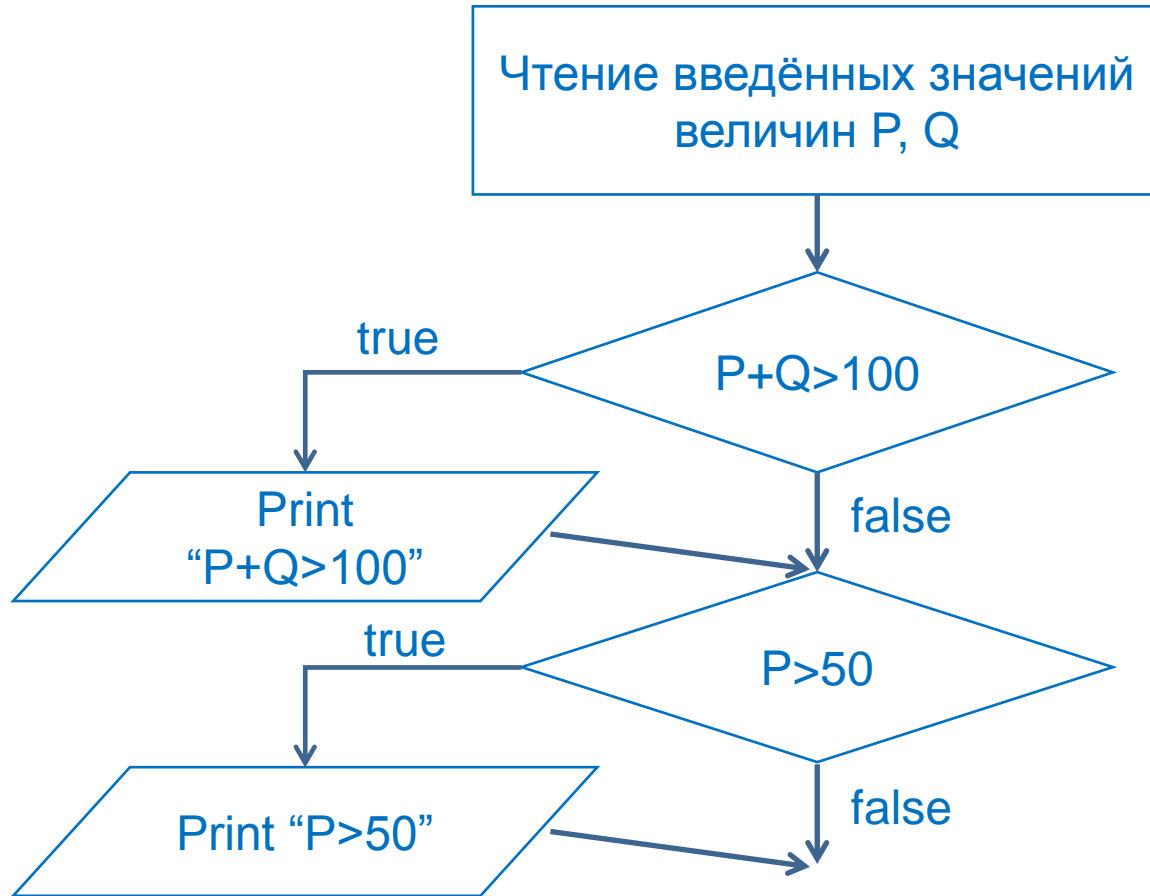
```
Read P
Read Q
If P+Q>100 Then
    Print "P+Q>100"
End If
If P>50 Then
    Print "P>50"
End If
```

Блок-схема (UML-диаграмма):



Техника statement coverage (SC)

Техника statement coverage: все statements должны быть покрыты (исполнены хотя бы один раз каждый).



TC step

Actions:

1.1 В поле «Величина P=» введите значение ...

1.2 В поле «Величина Q=» введите значения ...

1.3 Наж

ER1: Вывод в поле «Результат» значения ...

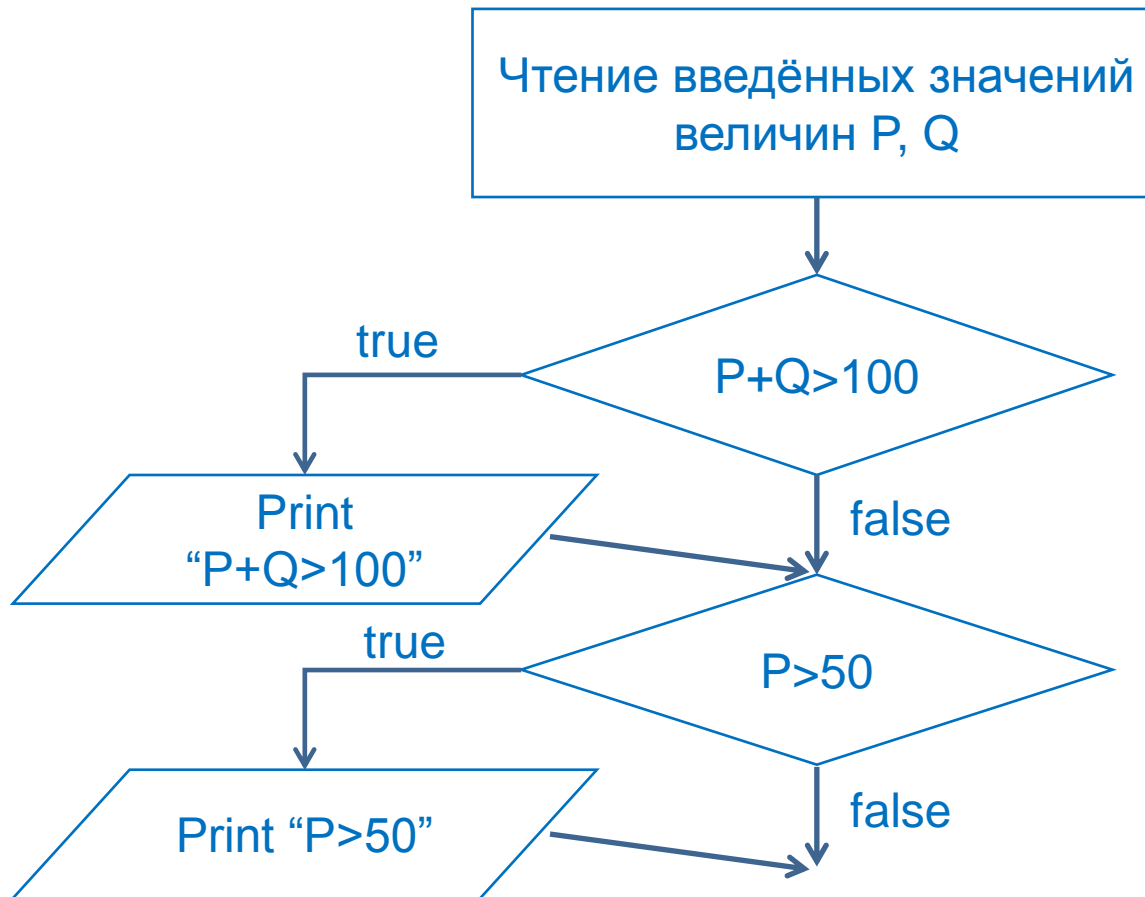
1.1. P = 60

1.2. Q = 60

ER1: {"P+Q>100", "P > 50"}

Техника condition coverage (CC)

Техника condition coverage : все conditional statements должны принять значения и true, и false хотя бы один раз каждый.



TC steps:

1.1. P = 10
1.2. Q = 100
ER1: {"P+Q>100"}

2.1. P = 60
2.2. Q = 10
ER2: {"P>50"}

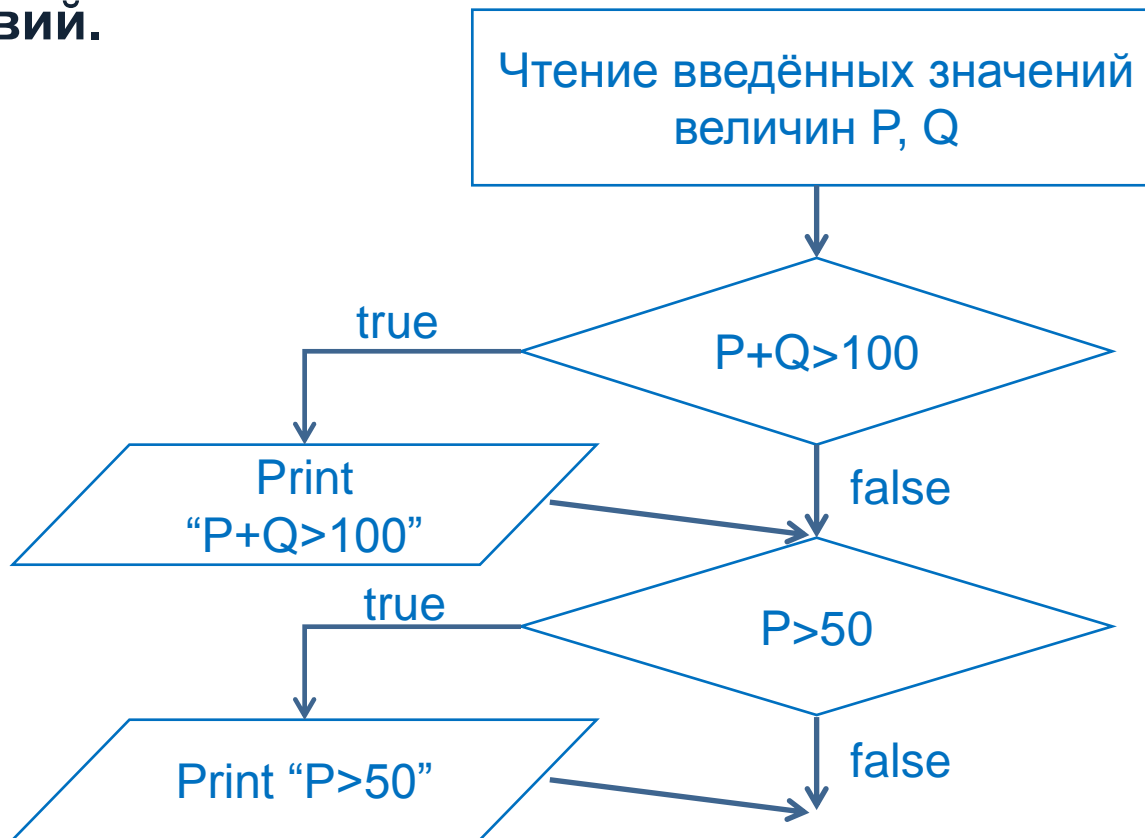
или

1.1. P = 100
1.2. Q = 10
ER1: {"P+Q>100", "P>50"}

2.1. P = 10
2.2. Q = 10
ER2: {" "}

Техника multiple condition coverage (Multiple CC)

Multiple condition coverage: все conditional statements должны принять значения и true, и false хотя бы один раз каждый и во всех возможных комбинациях друг с другом. Количество возможных комбинаций равно 2^m , где m – количество условий.



TC steps:

1.1. P = 10
1.2. Q = 100
ER1: {"P+Q>100"}

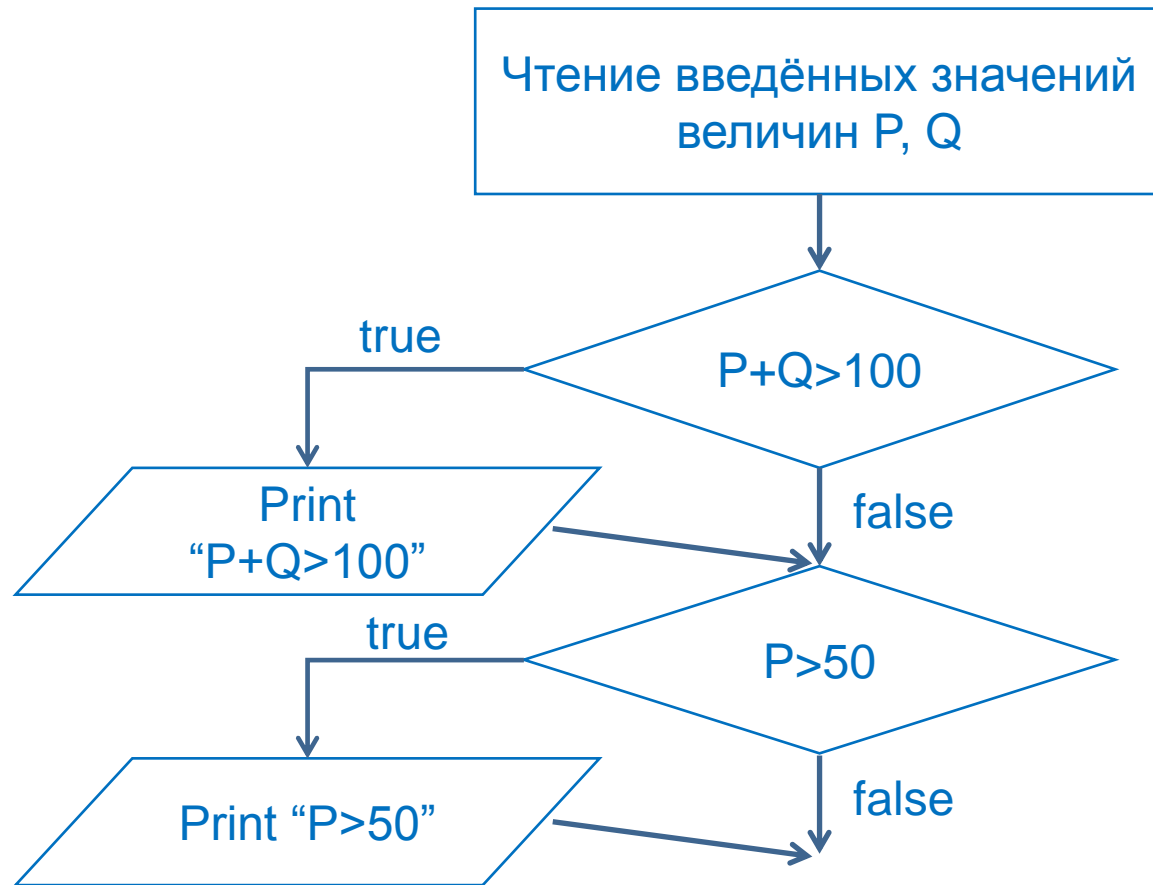
2.1. P = 60
2.2. Q = 10
ER2: {"P>50"}

3.1. P = 100
3.2. Q = 10
ER3: {"P+Q>100", "P>50"}

4.1. P = 10
4.2. Q = 10
ER4: {" "}

Техника decision coverage (DC)

Техника decision coverage : все ветви условий должны быть покрыты (исполнены хотя бы раз каждый).



TC steps:

1.1. P = 10
1.2. Q = 100
ER1: {"P+Q>100"}

2.1. P = 60
2.2. Q = 10
ER2: {"P>50"}

или

1.1. P = 100
1.2. Q = 10
ER1: {"P+Q>100", "P>50"}

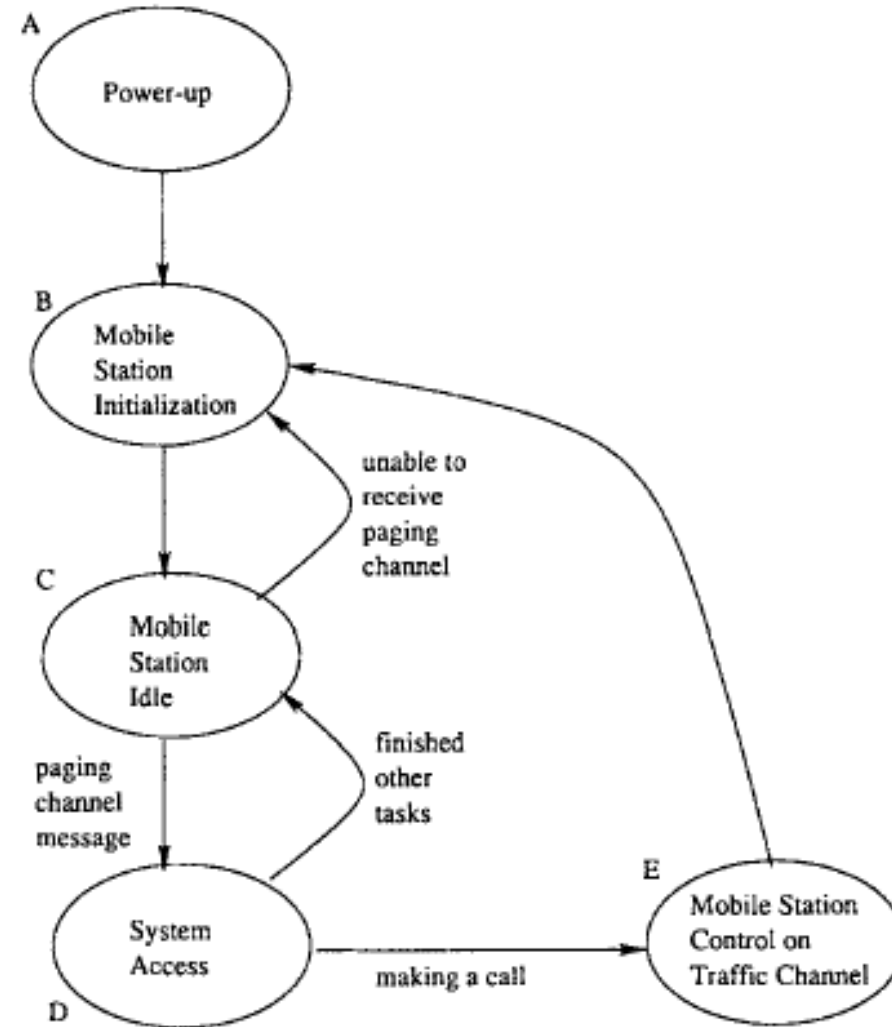
2.1. P = 10
2.2. Q = 10
ER2: {" "}

Finite state machine (пример 1)

Пример представления FSM:

- в виде диаграммы;
- в виде таблицы.

	A	B	C	D	E
A	na	-/-	na	na	na
B	na	na	-/-	na	na
C	na	NoC/-	na	msg/-	na
D	na	na	done/-	na	call/-
E	na	-/-	na	na	na



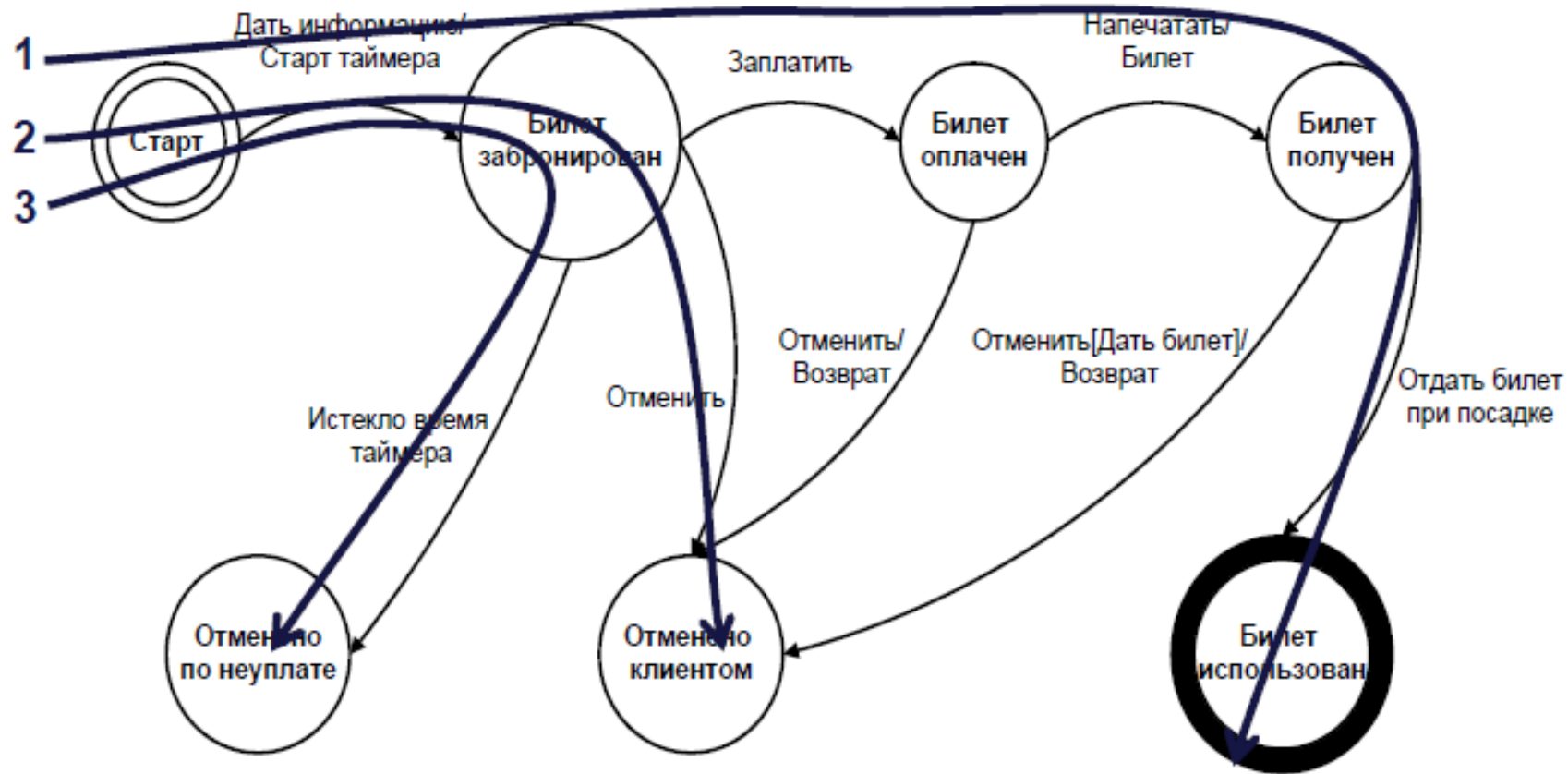
Finite state machine (пример 2)

Пример FSM



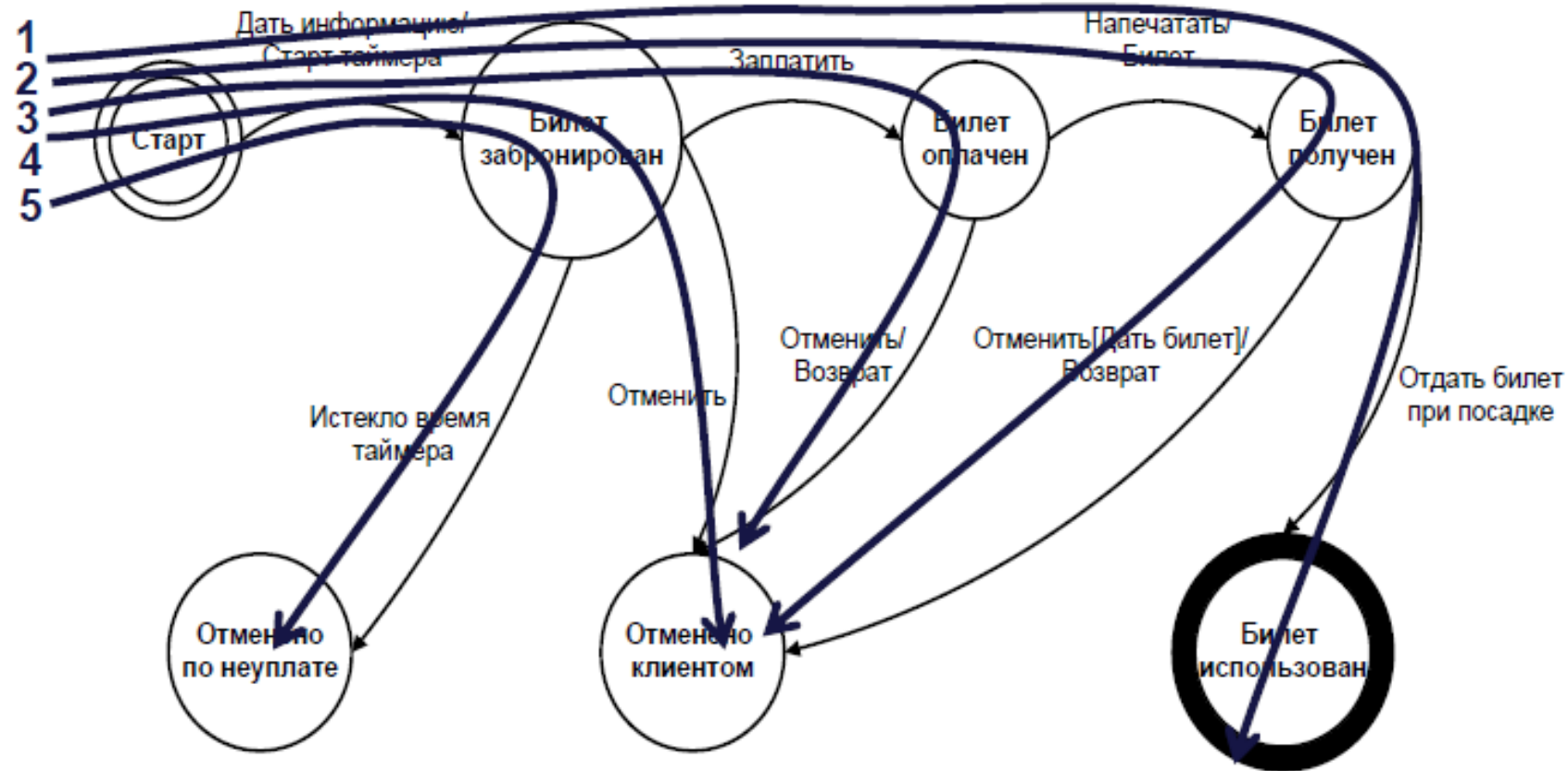
Finite state machine (пример дизайна ТС)

Пример тестовых сценариев, покрывающих все состояния FSM



Finite state machine (пример дизайна ТС)

Пример тестовых сценариев, покрывающих все переходы FSM



2

Техники для тестирования ПО в статике (статические техники)

Полный перебор (brute force)

Недостатки дизайна тест-кейсов на основе полного перебора:

- необходимость писать огромное число сценариев;
- наличие лишних сценариев;
- неэффективное использование ресурсов

Пример. Программа получает на вход три значения (a, b и c), после чего решает квадратное уравнение $ax^2 + bx + c = 0$,

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Сколько существует комбинаций входных значений такой программы?

Equivalence class partitioning (ECP)

Equivalence class (класс эквивалентности) – набор значений входных данных программы, которые приводят к «эквивалентным» AR. «Эквивалентность» AR определяется из требований и спецификации.

Equivalence class partitioning (разбиение на классы эквивалентности) – техника дизайна тестовых сценариев, основанная на разделении значений входных данных программы на классы эквивалентности и использовании в ТС по одному значению из каждого класса.

ЕСР (продолжение)

Пример. ПО вычисляет размер скидки.

Input: количество единиц товара (N), которое покупает клиент.

Output:

- скидка 7% при покупке от 100 до 999 единиц товара;
- 10% - при покупке от 1000 единиц товара включительно

Классы эквивалентности	
Валидные классы (valid)	Невалидные классы (invalid)
1) $1 \leq N < 100$ 2) $100 \leq N < 999$ 3) $1000 \leq N$	1) $N = 0$ 2) $N \leq -1$ 3) Null 4) "Строка" 5) 123.4567 6) $Max < N$

Область применения ECP

- числа (-, +, 0, 0.9999999, 0.000000006, 10^{12})
- символы (“symbol”, ”СИМВОЛ”, ”SiMBoL”, ”!@#\$%^”)
- количество (записей в БД, строк)
- типы файлов (.bmp, .jpg, .png)
- длина строки (Null, 200, $\approx\infty$)
- размер файла (Null, 4+Gb)
- объем памяти (меньше/больше требований)
- разрешение экрана (640x480, 1920x1080, 2560x1440)
- версии ОС, библиотек (Mac OS/Win/Linux, DX9/10/11)
- браузеры (Firefox, Safari, Chrome, IE6/7/8, Opera).

Boundary value analysis (BVA)

Boundary value (граничное значение) – значение на границе классов эквивалентности.

Boundary value analysis (анализ граничных значений) – техника дизайна тестовых сценариев, основанная на использовании в ТС значений из следующих областей:

- из границы между классами эквивалентности;
- из классов эквивалентности вблизи границы

Обычно используют по одному значению из каждой области.

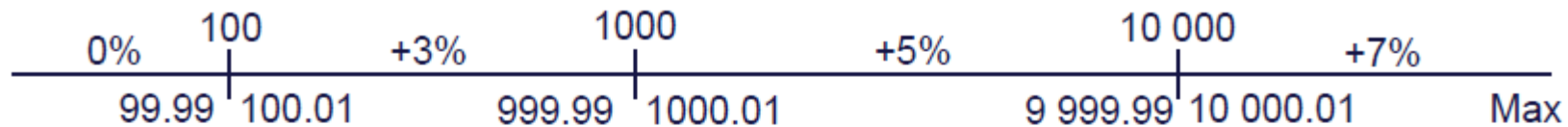
BVA (продолжение)

Пример. ПО рассчитывает бонус в зависимости от суммы на счету клиента.

Input: сумма на счету.

Output:

- 3% при сумме на счету от \$100 до \$1000 включительно;
- 5% при сумме от \$1000 до \$10000;



Границы классов эквивалентности		
	Валидные границы	Невалидные границы
0%	\$99.99, \$0	-\$0.01
+3%	\$100.00, \$100.01, \$999.99, \$1000.00	---
+5%	\$1000.01, \$9 999.99	---
+7%	\$10 000, \$10 000.01, \$Max	\$Max + \$0.01

Правила “хорошего тона” для ESR/BVA

ESR. При наличии нескольких полей (переменных) :

- **валидные классы нескольких переменных могут объединяться в один тест-кейс;**
- **невалидные классы нескольких переменных тестируются по отдельности.**

BVA. При наличии нескольких полей (переменных) :

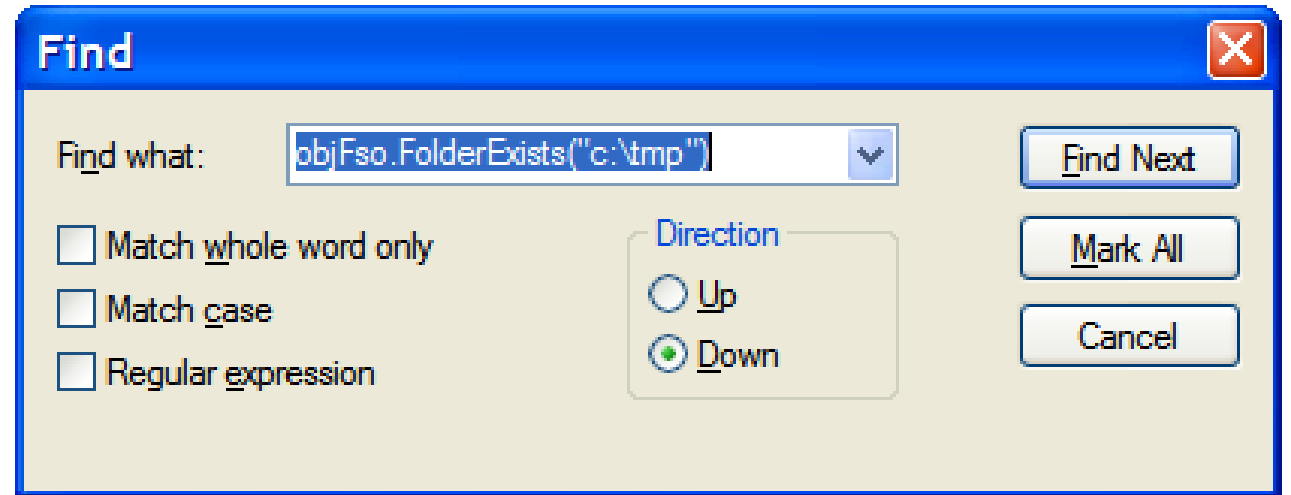
- **минимальные значения валидных границ объединяются в один тест-кейс;**
- **максимальные значения валидных границ объединяются в другой тест-кейс;**
- **при необходимости, минимальные и максимальные значения могут объединяться в один тест;**
- **невалидные границы тестируются по отдельности.**

Pairwise testing

Пример. Окно поиска в текстовом редакторе.

Входные данные:

- FW – String, например: {'lower', 'UPPER', 'MiXed'}
- MW – Boolean
- MC – Boolean
- RE – Boolean
- D – {Up, Down}



Pairwise testing

Полный перебор (количество кейсов – максимальное)

	FW	MW	MC	RE	D
1	L	Y	Y	Y	Up
2	U	Y	Y	Y	Up
3	M	Y	Y	Y	Up
4	L	Y	Y	Y	Up
5	L	N	Y	Y	Up
...
47	M	N	N	N	Up
48	M	N	N	N	D

Pairwise testing (метод всех пар) – техника дизайна тестовых сценариев, заключающаяся в составлении всех возможных комбинаций значений входных данных для каждой пары данных.

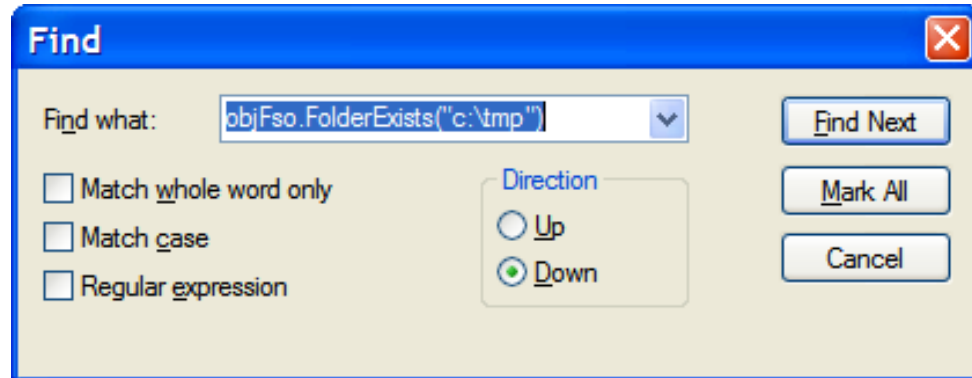
Всё по одному разу (например, для smoke testing)

	FW	MW	MC	RE	D
1	L	Y	N	Y	Up
2	U	N	Y	N	D
3	M	Y	Y	N	Up

Pairwise testing

Все пары (например, для выделения тестов первого приоритета)

	FW	MW	MC	RE	D
1	L	Y	N	Y	Up
2	L	N	Y	N	D
3	U	Y	Y	N	Up
4	U	N	N	Y	D
5	M	N	N	N	Up
6	M	Y	Y	Y	D



Q&A

Thank You

