Focused forward
Netcracker

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

© Netcracker 2016

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.

© Netcracker 2016

Место test design в тестировании

Test design → Test execution → Defect reporting

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?**

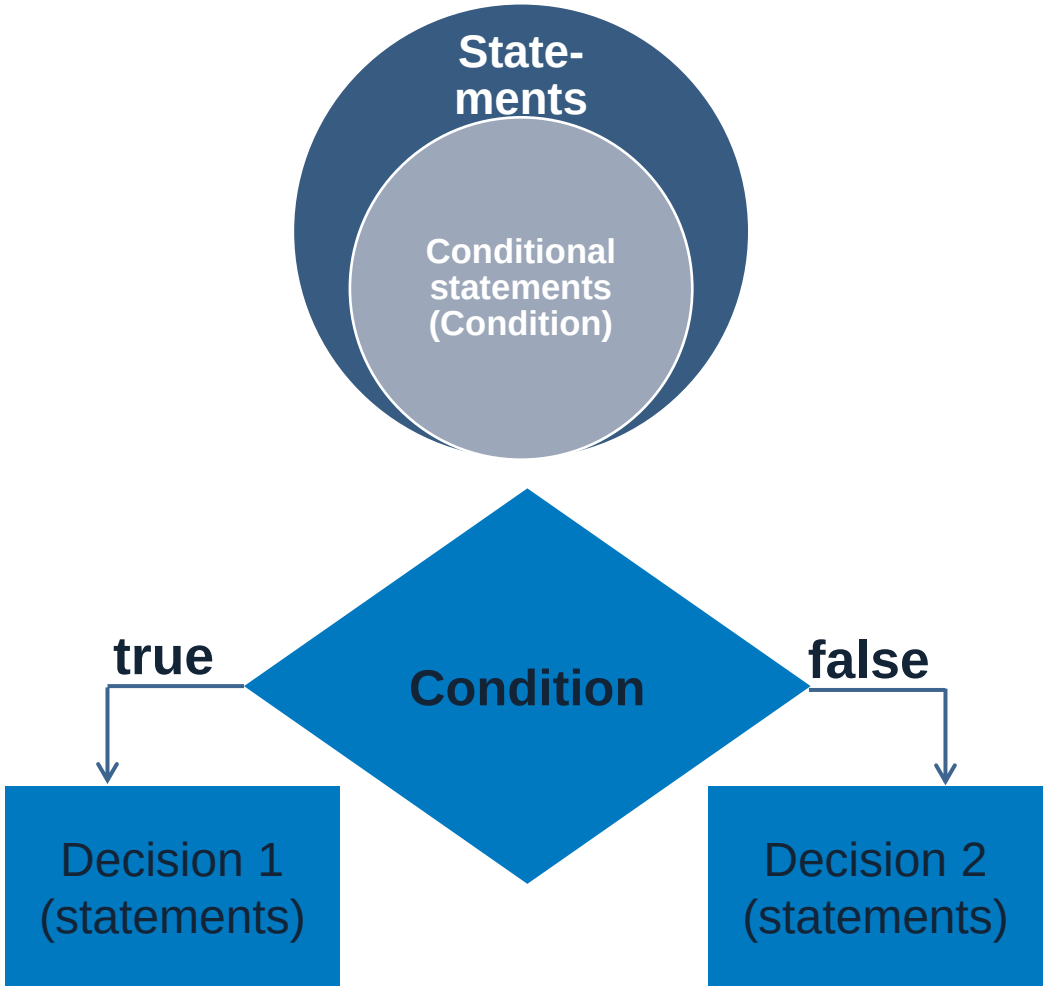
© Netcracker 2016

1

Техники для тестирования ПО в динамике (динамические техники)

© Netcracker 2016

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

© Netcracker 2016

Пример

GUI:

Величина P =

Величина Q =

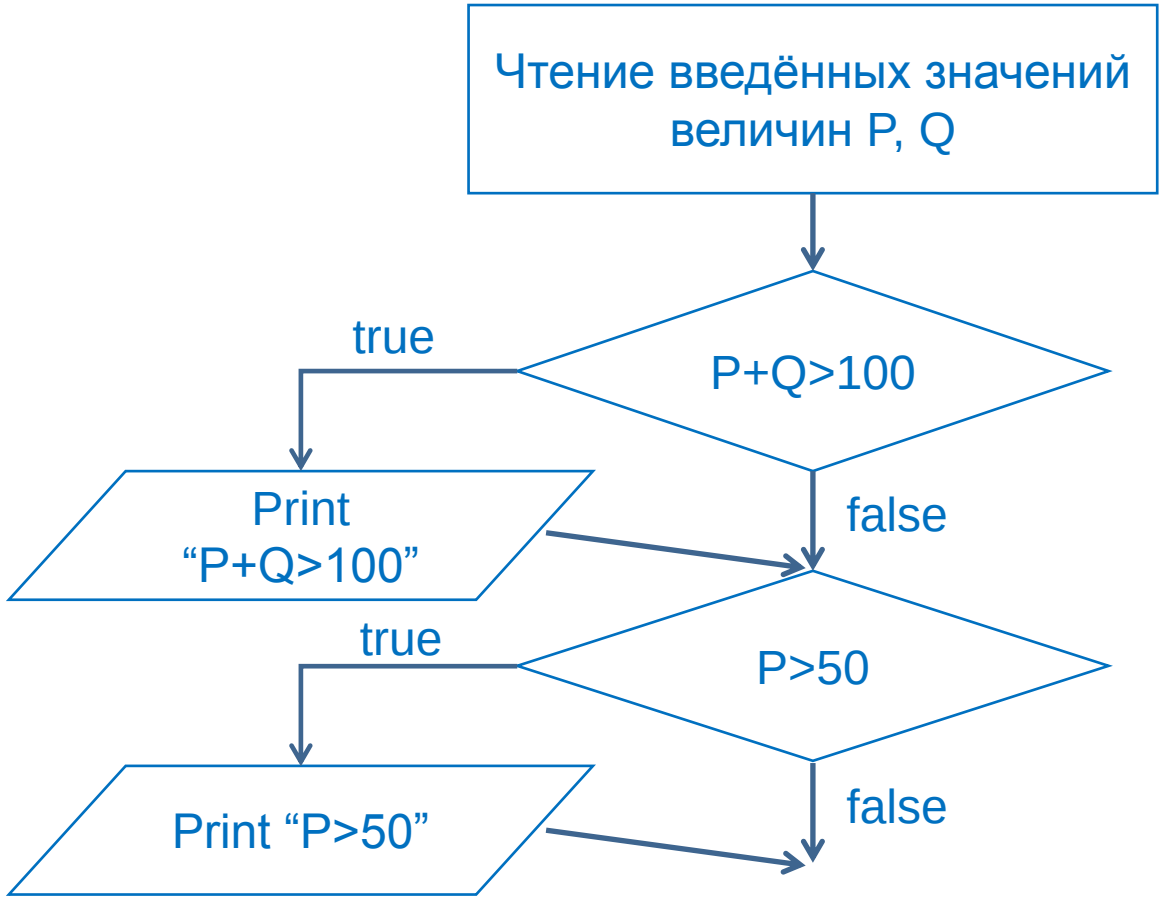
Результат: ?

Узнать 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
```

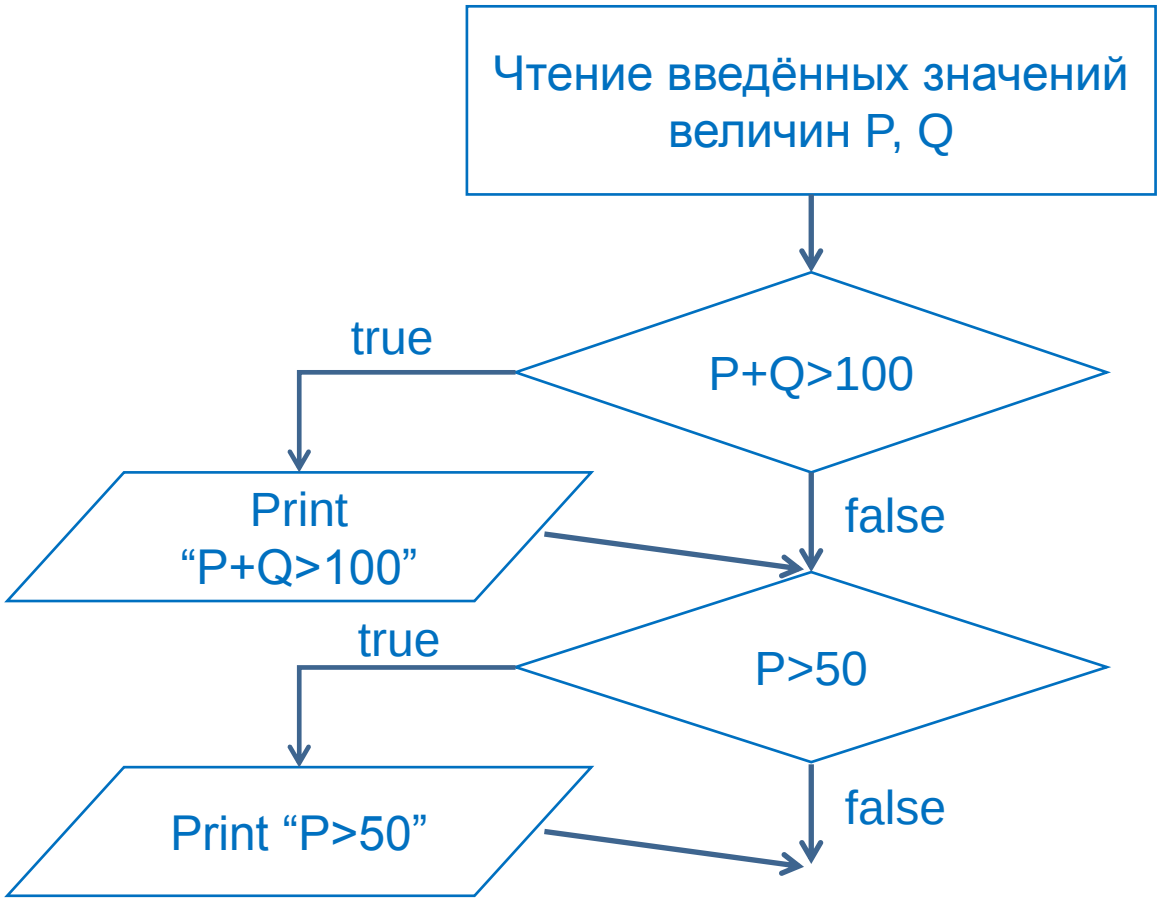
Блок-схема (UML-диаграмма):



© Netcracker 2016

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

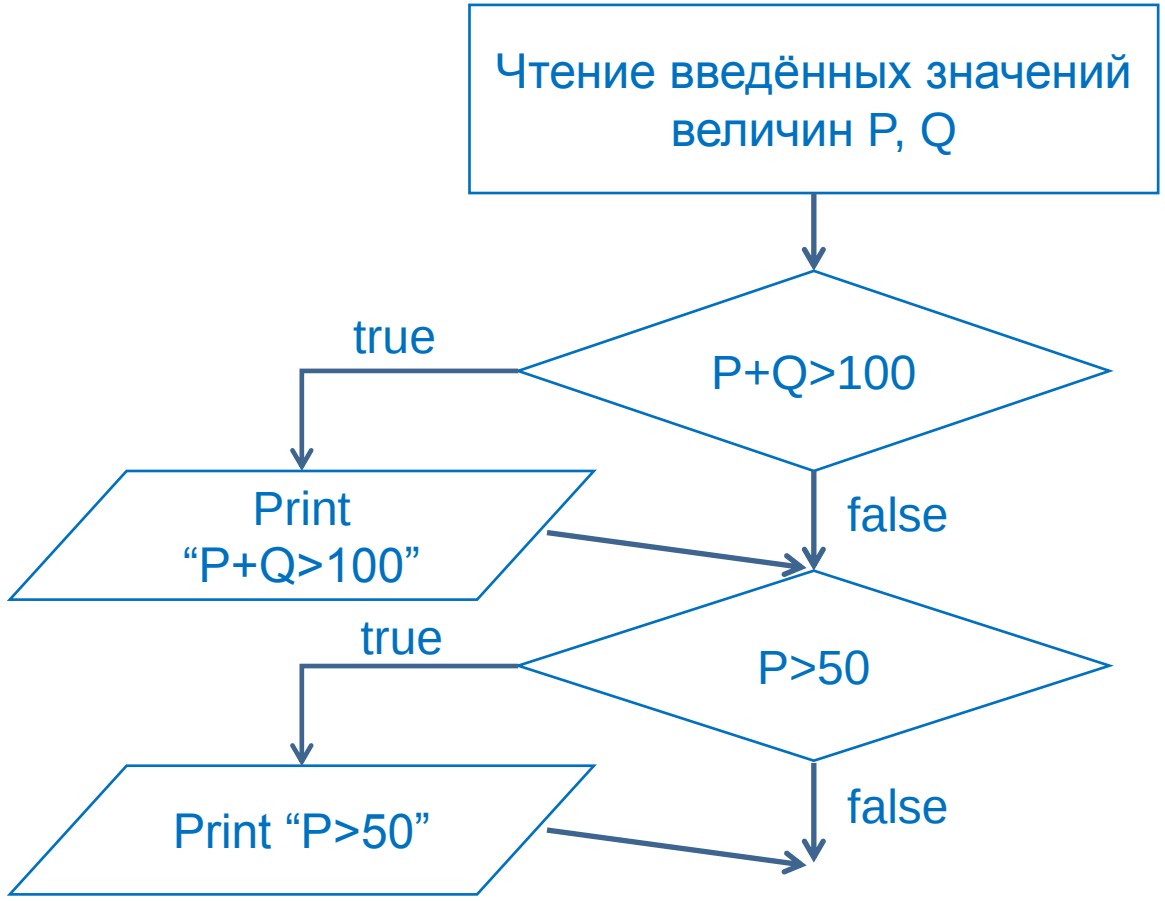
ER1: Вывод в поле «Результат» значения …

1.1. P = 60
1.2. Q = 60
ER1: {“P+Q>100”, “P > 50”}

© Netcracker 2016

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

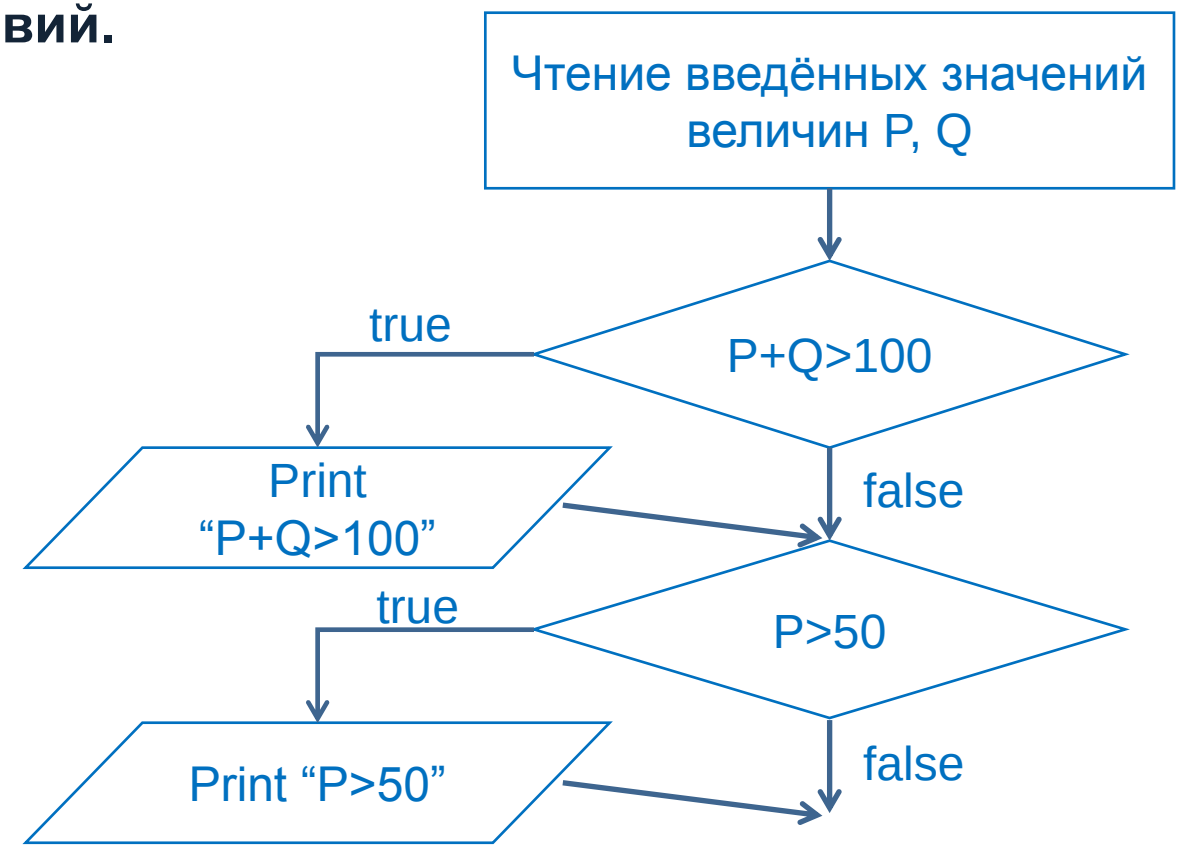
1.1. P = 100
1.2. Q = 10
ER1: {“P+Q>100”, “P>50”}

2.1. P = 10
2.2. Q = 10
ER2: {“ ”}

© Netcracker 2016

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"}

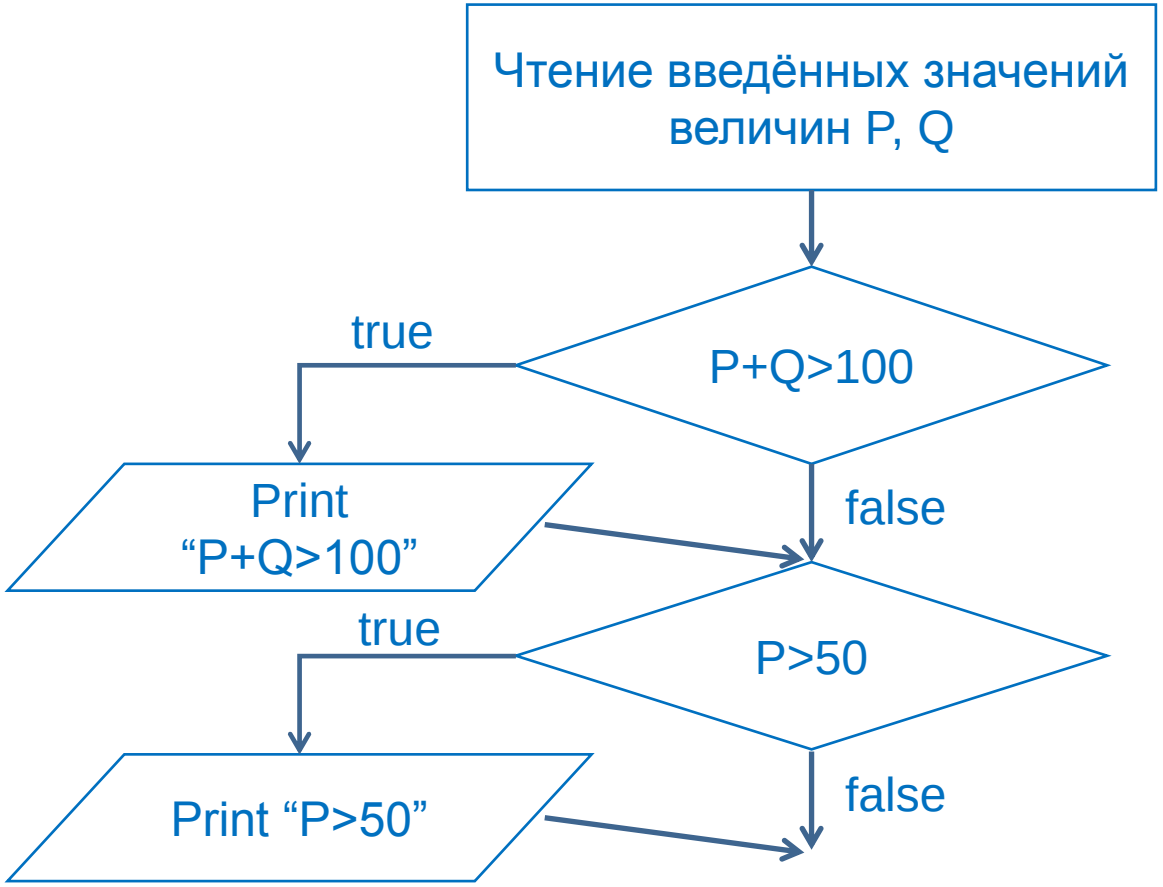
3.1. P = 100
3.2. Q = 10
ER3: {"P+Q>100", "P>50"}

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

© Netcracker 2016

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: {“ ”}

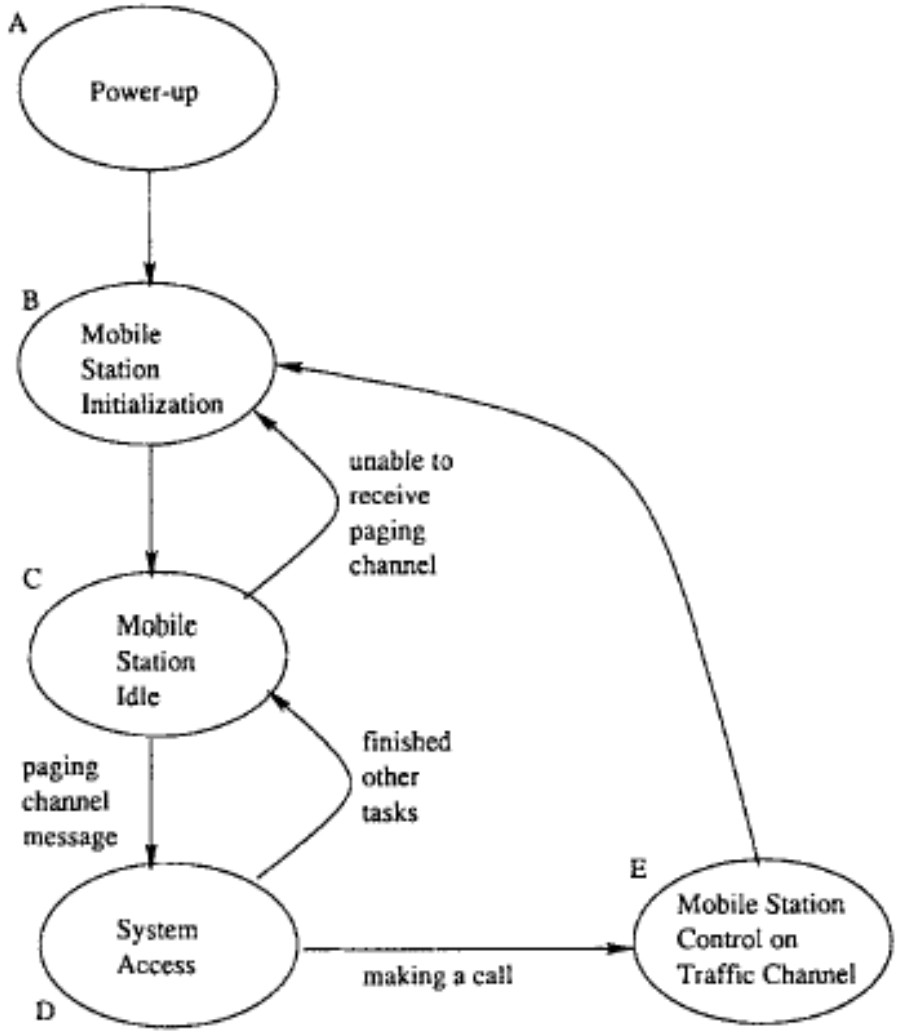
© Netcracker 2016

Finite state machine (пример 1)

Пример представления FSM:

- **в виде диаграммы;**
- **в виде таблицы.**

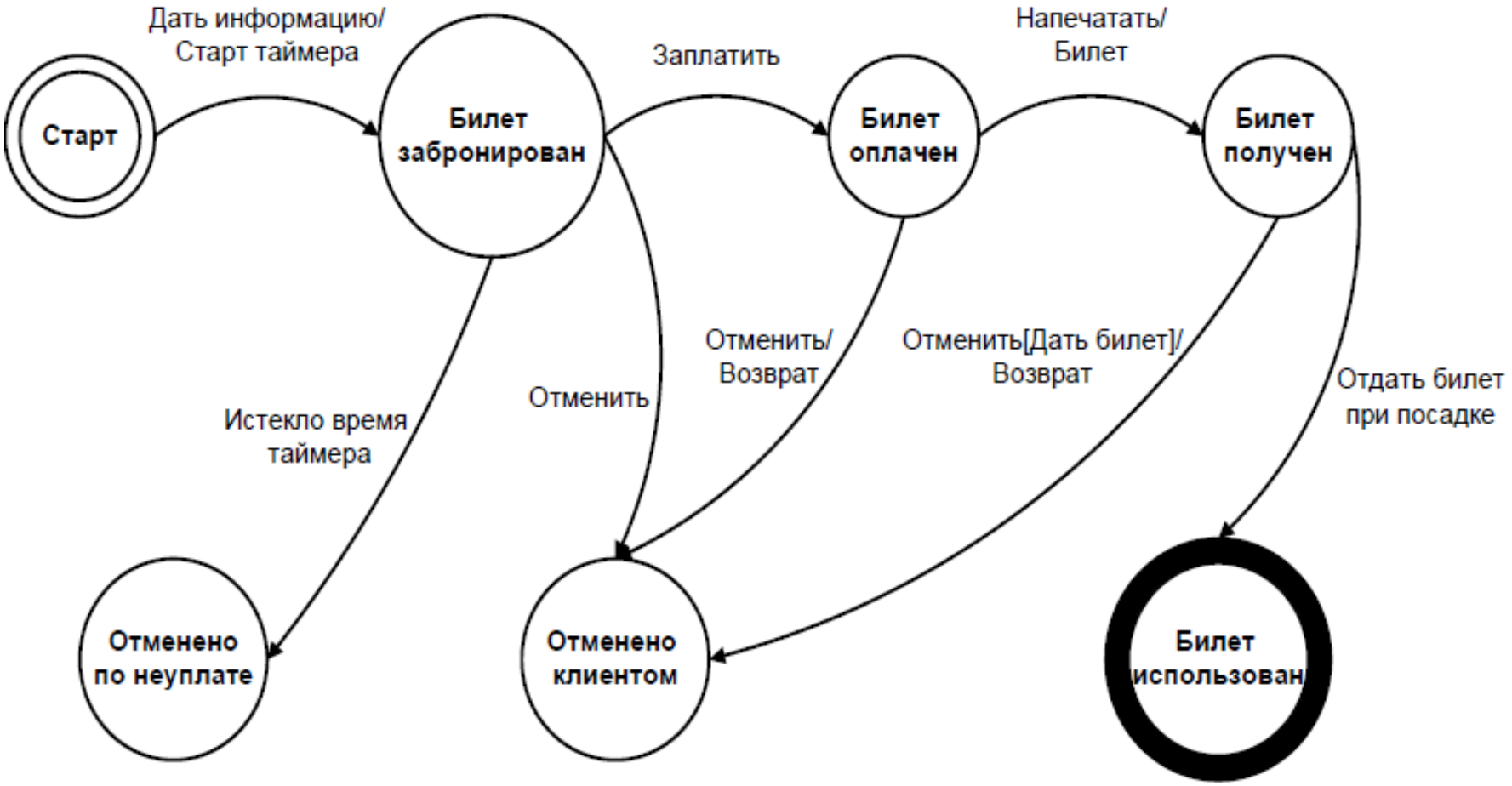
	A	B	C	D	E
A	na	-/-	na	na	na
B	na	na	-/-	na	na
C	na	NoC/-	na	msg/-	na
D	na	na	done/-	na	call/-
E	na	-/-	na	na	na



© Netcracker 2016

Finite state machine (пример 2)

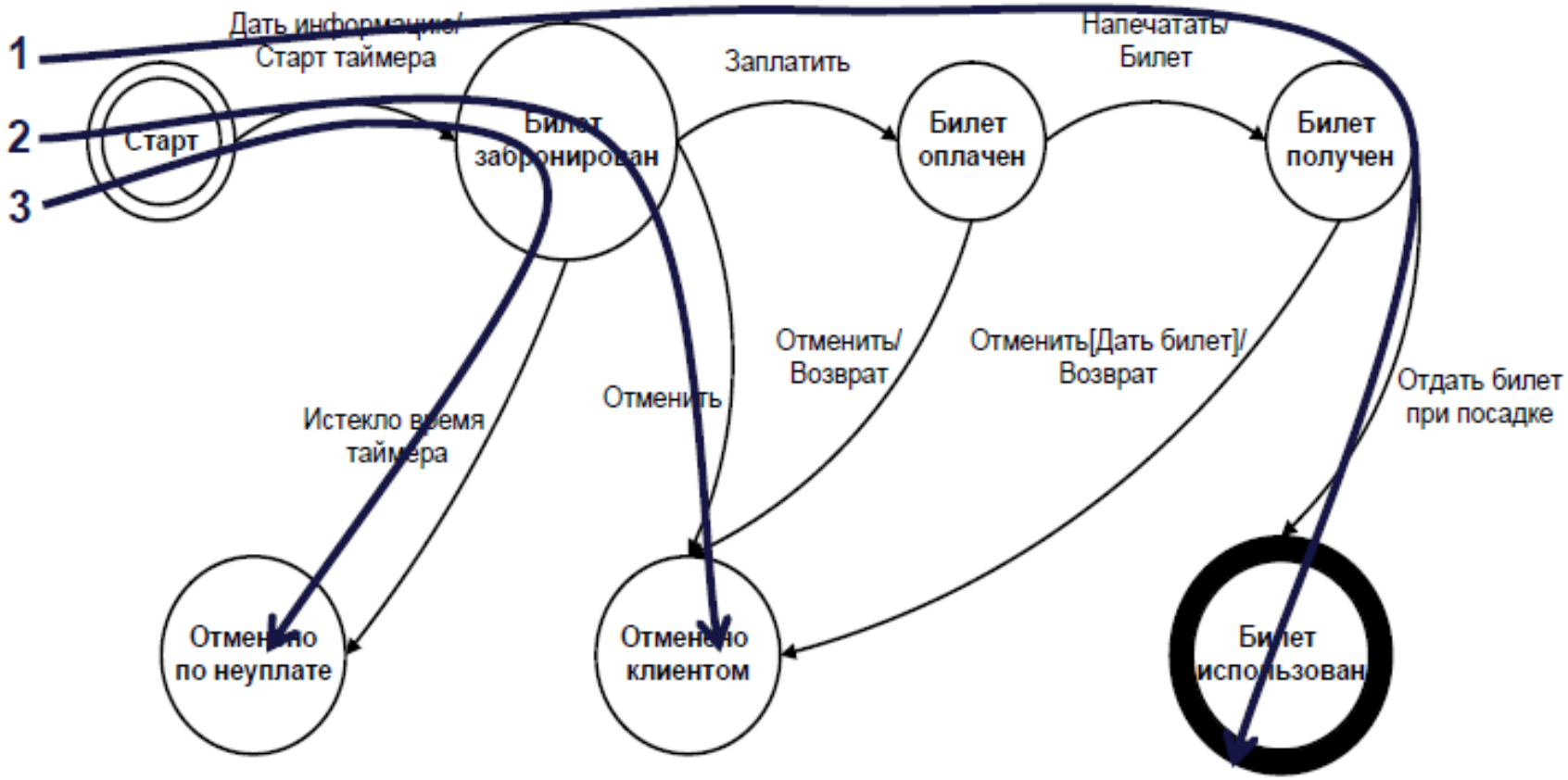
Пример FSM



© Netcracker 2016

Finite state machine (пример дизайна ТС)

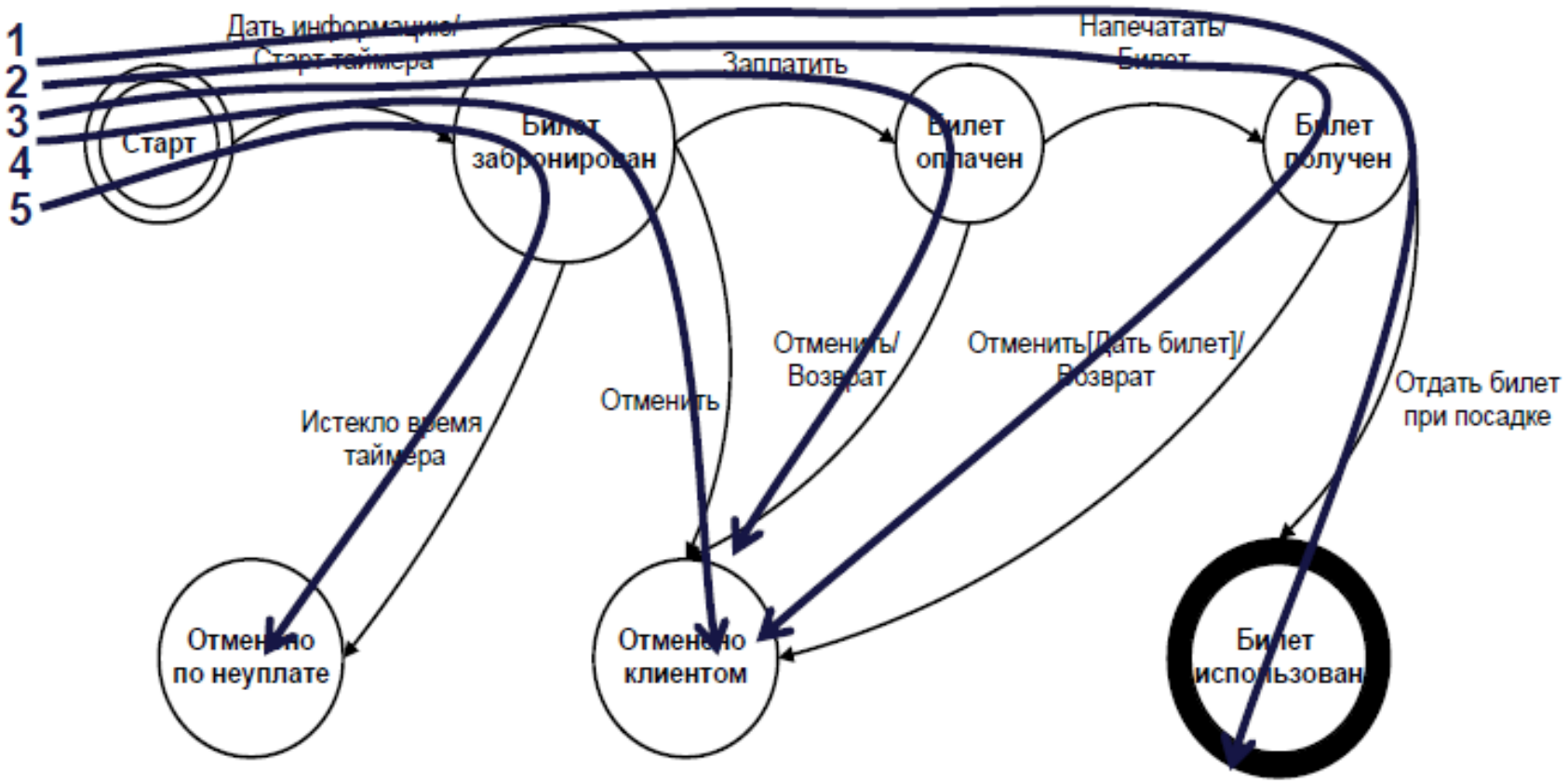
Пример тестовых сценариев, покрывающих все состояния FSM



© Netcracker 2016

Finite state machine (пример дизайна ТС)

Пример тестовых сценариев, покрывающих все переходы FSM



© Netcracker 2016

2

Техники для тестирования ПО в статике (статические техники)

© Netcracker 2016

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?

© Netcracker 2016

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 разделение значений входных данных программы на классы эквивалентности и использовании в ТС по одному значению из каждого класса.

© Netcracker 2016

ECP (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 ≤ N < 100 2) 100 ≤ N < 999 3) 1000 ≤ N	1) N = 0 2) N ≤ -1 3) Null 4) “Строка” 5) 123.4567 6) Max < N

© Netcracker 2016

Область применения ECP

- **числа (-, +, 0, 0.9999999, 0.00000006, 10^{12})**
- **символы (“symbol”, ”СИМВОЛ”, ”SиMвоL”, ”!@#$%^”)**
- **количество (записей в БД, строк)**
- **типы файлов (.bmp, .jpg, .png)**
- **длина строки (Null, 200, ≈∞)**
- **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).**

© Netcracker 2016

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.

© Netcracker 2016

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

© Netcracker 2016