

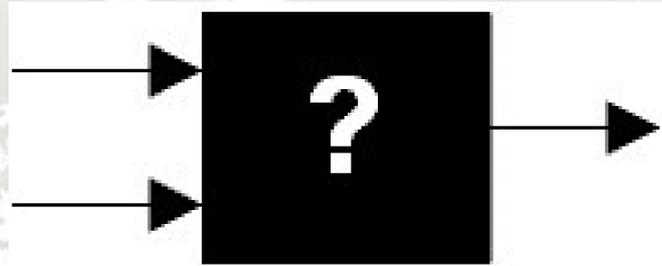


# **Testing strategies**

# **Стратегии тестирования**



# Specification-based or Black-box Techniques



## ***Метод чёрного ящика***

***(blackbox / closedbox / specification-based testing)*** — у тестировщика либо нет доступа к внутренней структуре и коду приложения, либо недостаточно знаний для их понимания, либо он сознательно не обращается к ним в процессе тестирования.

Тестировщик концентрируется на том, что делает программное обеспечение, а не как он это делает.



# Structure-based or White-box Techniques

**Метод белого ящика (*whitebox* | *openbox* | *clearbox* | *glassbox testing*)** — у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного. Некоторые авторы склонны жёстко связывать этот метод со статическим тестированием, но ничто не мешает тестировщику запустить код на выполнение и при этом периодически обращаться к самому коду.





# Grey-box testing

**Метод серого ящика (*graybox testing*)** — комбинация методов белого ящика и чёрного ящика, состоящая в том, что к части кода и архитектуры у тестировщика доступ есть, а к части — нет. Обычно говорят о методах белого или чёрного ящика в применении к тем или иным частям приложения, при этом понимая, что «приложение целиком» тестируется по методу серого ящика.



# White-box

## Преимущества

- Показывает скрытые проблемы и упрощает их диагностику.
- Допускает достаточно простую автоматизацию тест-кейсов и их выполнение на самых ранних стадиях развития проекта.
- Обладает развитой системой метрик, сбор и анализ которых легко автоматизируется.
- Стимулирует разработчиков к написанию качественного кода.
- Многие техники этого метода являются проверенными, хорошо себя зарекомендовавшими решениями, базирующимися на строгом техническом подходе.

## Недостатки

- Не может выполняться тестировщиками, не обладающими достаточными знаниями в области программирования.
- Тестирование сфокусировано на реализованной функциональности, что повышает вероятность пропуска нереализованных требований.
- Поведение приложения исследуется в отрыве от реальной среды выполнения и не учитывает её влияние.
- Поведение приложения исследуется в отрыве от реальных пользовательских



# Black-box

## Преимущества

- Тестировщик не обязан обладать (глубокими) знаниями в области программирования.
- Поведение приложения исследуется в контексте реальной среды выполнения и учитывает её влияние.
- Поведение приложения исследуется в контексте реальных пользовательских сценариев.
- Тест-кейсы можно создавать уже на стадии появления стабильных требований.
- Процесс создания тест-кейсов позволяет выявить дефекты в требованиях.
- Допускает создание тест-кейсов, которые можно многократно использовать на разных проектах.

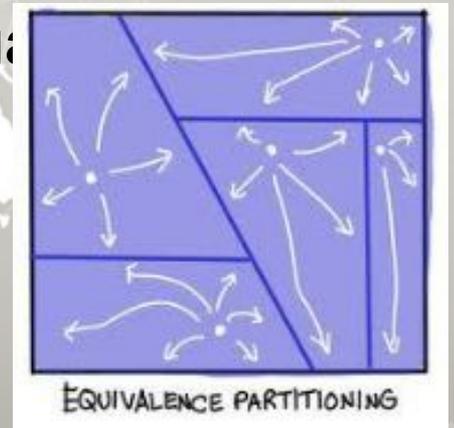
## Недостатки

- Возможно повторение части тест-кейсов, уже выполненных разработчиками.
- Высока вероятность того, что часть возможных вариантов поведения приложения останется протестированной.
- Для разработки высокоэффективных тест-кейсов необходима качественная документация.
- Диагностика обнаруженных дефектов более сложна.
- В связи с широким выбором техник и подходов затрудняется планирование и оценка трудозатрат.
- В случае автоматизации могут потребоваться сложные дорогостоящие инструментальные средства.



# Equivalence Partitioning

**Эквивалентное разбиение (*equivalence partitioning*)** - техника тестирования, направленная на сокращение количества разрабатываемых и выполняемых тест-кейсов при сохранении достаточного тестового покрытия. Основывается на предположении, что входы и выходы компонента могут быть разделены на классы, которые, в соответствии со спецификацией компонента, будут обрабатываться компонентом аналогично. Таким образом, результат тестирования одного значения из класса эквивалентности покрывает весь класс.





# Equivalence Partitioning

In a system designed to work out the tax to be paid:

An employee has £4000 of salary tax free.

The next £1500 is taxed at 10%.

The next £28000 after that is taxed at 22%.

Any further amount is taxed at 40%.

*To the nearest whole pound, which of these groups of numbers fall into three **DIFFERENT** equivalence classes?*

A £28000; £28001; £32001.

B £4000; £5000; £5500.

C £4000; £4200; £5600.

D £32001; £34000; £36500.

$X \leq £4000 \rightarrow 0\%$

$£4000 < X \leq £5500 \rightarrow 10\%$

$£5500 < X \leq £33500 \rightarrow 22\%$

$X > £33500 \rightarrow 40\%$



# Equivalence Partitioning

*Given the following specification, which of the following values for age are in the **SAME** equivalence partition?*

If you are less than 18, you are too young to be insured.

Between 18 and 30 inclusive, you will receive a 20% discount.

Anyone over 30 is not eligible for a discount.

**A 29, 30, 31**

**B 17, 29, 31**

**C 18, 29, 30**

**D 17, 18, 19.**

$X < 18 \rightarrow 0\%$

$18 \leq X \leq 30 \rightarrow 20\%$

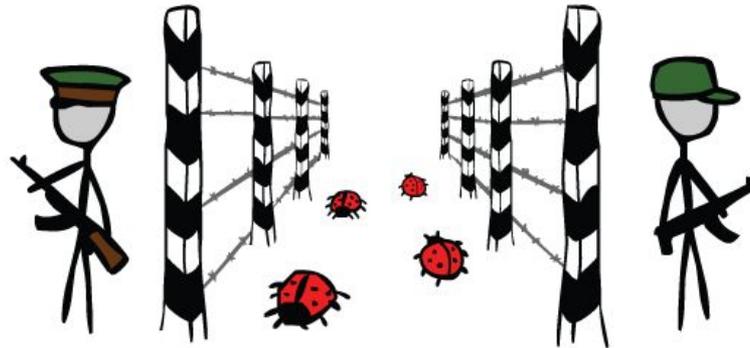
$X > 30 \rightarrow 0\%$



# Boundary Value Analysis

Тестирование на основе **граничных условий** (*boundary value analysis*) – это техника проверки ошибок на границах классов эквивалентности. Если техника анализа классов эквивалентности ориентирована на тестовое покрытие, то эта техника основана на рисках. Эта техника начинается с идеи о том, что программа может сломаться в области граничных значений.

Баги водятся на границах





# Boundary Value Analysis

In a system designed to work out the tax to be paid:

An employee has £4000 of salary tax free.

The next £1500 is taxed at 10%.

The next £28000 after that is taxed at 22%.

Any further amount is taxed at 40%.

*To the nearest whole pound, which of these is a valid Boundary Value Analysis test case?*

A £32001

B £28000

C £1500

D £33501.

$X \leq £4000 \rightarrow 0\%$

$£4000 < X \leq £5500 \rightarrow 10\%$

$£5500 < X \leq £33500 \rightarrow 22\%$

$X > £33500 \rightarrow 40\%$



# Decision Table Testing

R00101 For females less than 65 years of age, the premium is \$500

R00102 For males less than 25 years of age, the premium is \$3000

R00103 For males between 25 and 64 yrs of age, the premium is \$1000

R00104 For anyone 65 years of age or more, the premium is \$1500

**Table 1 – Causes and Effects**

Causes (input conditions)	Effects (output conditions)
C1. Sex is Male	E1. Premium is \$1000
C2. Sex is Female	E2. Premium is \$3000
C3. Age is <25	E3. Premium is \$1500
C4. Age is >=25 and < 65	E4. Premium is \$500
C5. Age is >= 65	



# Decision Table Testing

Test Case	1	2	3	4	5	6
<b><u>Causes:</u></b>						
C1 (male)	T	T	T	F	F	F
C2 (female)	F	F	F	T	T	T
C3 (<25)	T	F	F	F	T	F
C4 (>=25 and < 65)	F	T	F	F	F	T
C5 (>= 65)	F	F	T	T	F	F
<b><u>Effects:</u></b>						
E1 (Premium is \$1000)	F	T	F	F	F	F
E2 (Premium is \$3000)	T	F	F	F	F	F
E3 (Premium is \$1500)	F	F	T	T	F	F
E4 (Premium is \$500)	F	F	F	F	T	T

Test Case #	Inputs (Causes)		Expected Outputs (Effects)
	Sex	Age	Premium
1	Male	<25	\$3000
2	Male	>=25 & <65	\$1000
3	Male	>=65	\$1500
4	Female	>=65	\$1500
5	Female	<25	\$500
6	Female	>=25 & <65	\$500



# Decision Table Example 1

	Rule 1	Rule 2	Rule 3	Rule 4
<b><u>Conditions</u></b>				
UK resident?	F	TRUE	TRUE	TRUE
Age between 18-55	-	F	TRUE	TRUE
Smoker?	-	-	F	TRUE
<b><u>Actions</u></b>				
Insure client?	F	F	TRUE	TRUE
Offer 10% discount?	F	F	TRUE	F

What is the expected result for each of the following test cases?

***A.TC1: Fred is a 32 year old smoker resident in London***

***B.TC3: Jean-Michel is a 65 year non-smoker resident in Paris***



# Decision Table Example 2

	Rule 1	Rule 2	Rule 3	Rule 4
<b><u>Conditions</u></b>				
Frequent Flyer Member	Yes	Yes	No	No
Class	Business	Economy	Business	Economy
<b><u>Actions</u></b>				
Offer upgrade to First	Yes	No	No	No
Offer upgrade to Business	N/A	Yes	N/A	No

What is the expected result for each of the following test cases?

***A. Frequent flyer member, travelling in Business class***

***B. Non-member, travelling in Economy class***



# State Transition Testing

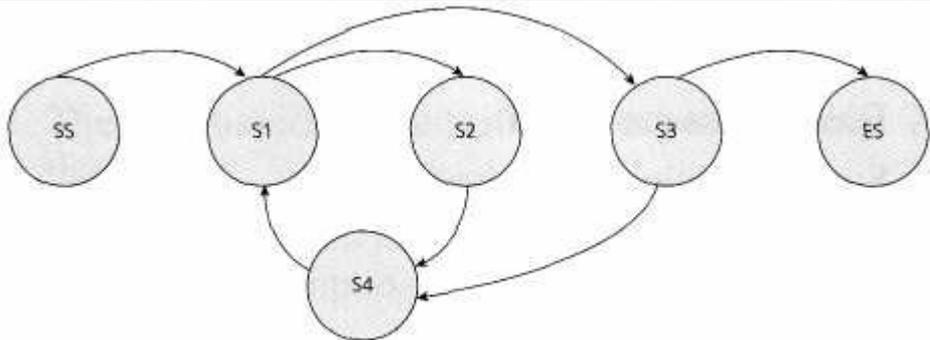


FIGURE 4.6 State diagram

	S1	S2	S3	S4	S5	ES
SS	+					
S1		+	+			
S2				+		
S3				+		+
S4	+					
S5						

Which test case is the minimum series of valid transitions to cover every state?

a. SS-S1-S2-S4-S1-S3-ES

b. SS-S1-S2-S3-S4-ES

c. SS-S1-S2-S4-S1-S3-S4-S1-S3-ES

d. SS-S1-S4-S2-S1-S3-ES

Which of the following represents an INVALID state transition?

A S3 from State S2.

B S4 from State S3.

B S2 from State S1.

D ES from State S3.



# Statement Testing and Coverage

In general a **statement** should be an atomic action, that is a statement should be executed completely or not at all. For instance:

**IF  $a$  THEN  $b$  ENDIF**

is considered as more than one statement because  $b$  may or may not be executed depending upon the condition  $a$ .

Consider, the following C code:

```
a;  
if (b) {  
    c;  
}  
d;
```

Any test case with  $b$  TRUE will achieve full statement coverage. Note that full statement coverage can be achieved without exercising with  $b$  FALSE.

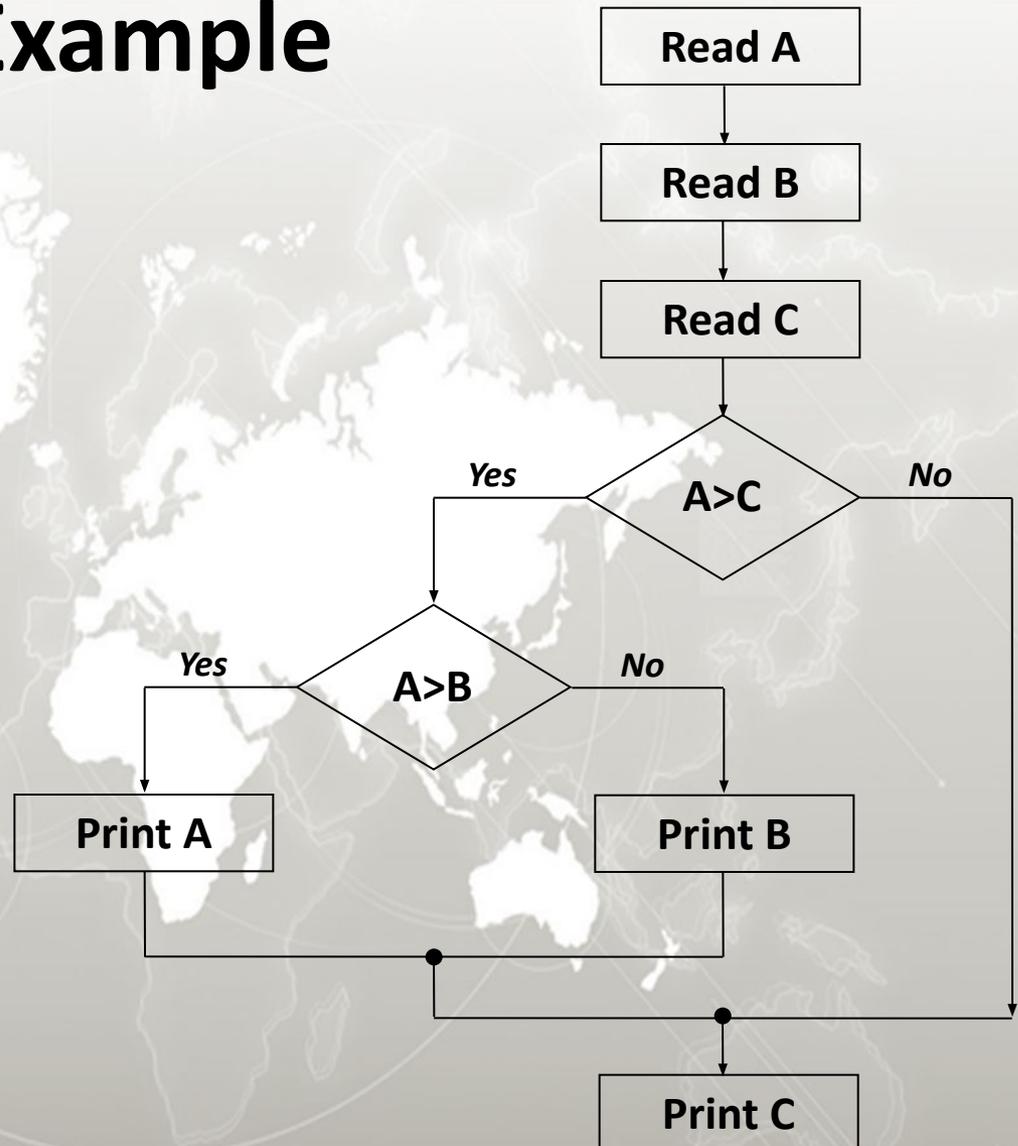


# Statement Coverage Example

```
READ A
READ B
READ C
IF A>C THEN
  IF A>B THEN
    PRINT "A "
  ELSE
    PRINT "B"
  ENDIF
ENDIF
PRINT "C"
```

TC 1:  $A > C, A > B$

TC2:  $A > C, A < B$





# Decision Testing and Coverage

**Decision coverage:** The percentage of decision outcomes that have been exercised by a test suite.

Consider, the following C code:

```
a;  
if (b) {  
    c;  
}  
d;
```

Full decision coverage should include b FALSE.



# Decision Coverage Example

```
READ A
READ B
READ C
IF A>C THEN
  IF A>B THEN
    PRINT "A "
  ELSE
    PRINT "B"
  ENDIF
ENDIF
PRINT "C"
```

TC 1:  $A > C, A > B$

TC 2:  $A > C, A < B$

TC 3:  $A < C$

