

502045

Software Engineering

Chapter 08

Lesson 12: Unit Testing

- Unit Test - What and Who?
- Unit Test - Why?
- Unit Test - How? (method, technique)
- Unit Test - Practice (3-5 labs) + Answer

- The course helps attendees understand:
 - Unit Test Fundamentals (basic): Answer the question of what, why, when and how to do unit test
 - Unit Test Technique: Introduce approaches and techniques to do unit test

Unit Test – What and Who ?

- Unit Testing Actions:

- Validate that individual units of software program are working properly
- A unit is the smallest testable part of an
- Units are distinguished from modules in that modules are typically made up of units

★ Unit Testing Deliverables:

- Tested software units
- Related documents (Unit Test case, Unit Test Report)

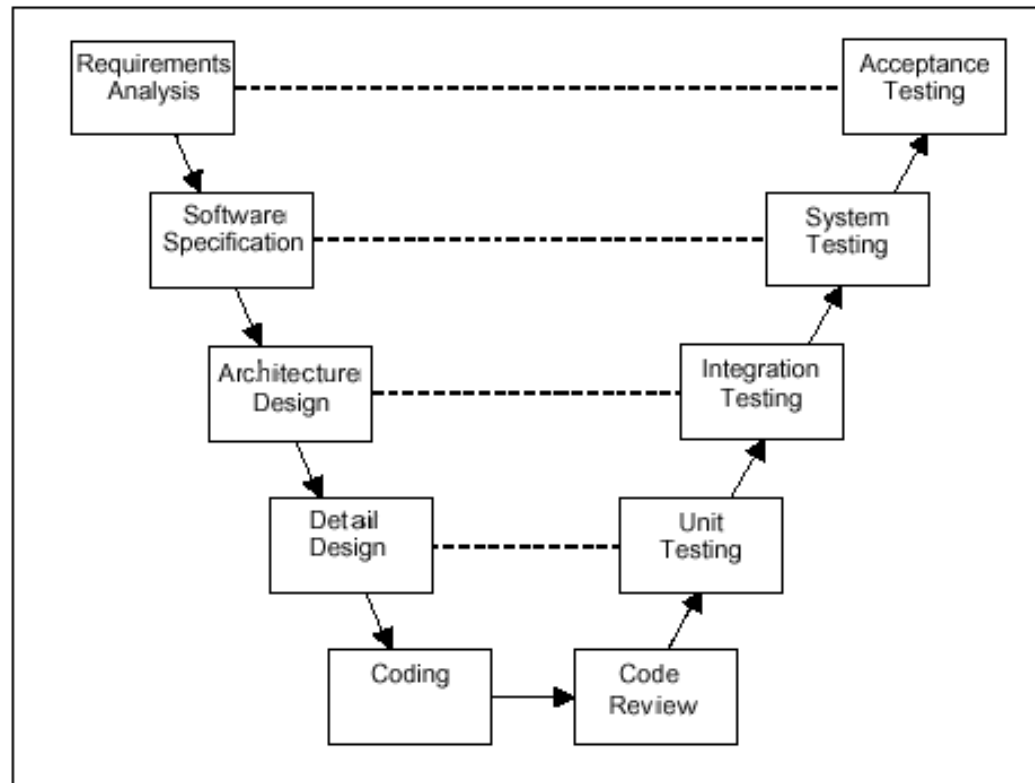
★ Unit Testing Conductor: Development team

Unit Test – Why ?

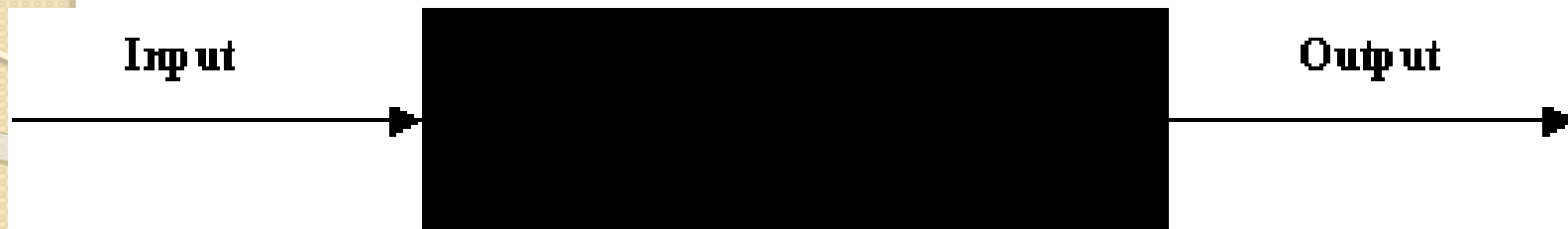
- Ensure quality of software unit
 - ★ Detect defects and issues early
 - ★ Reduce the Quality Effort & Correction Cost

Unit Test – When ?

- After Coding
- Before Integration Test



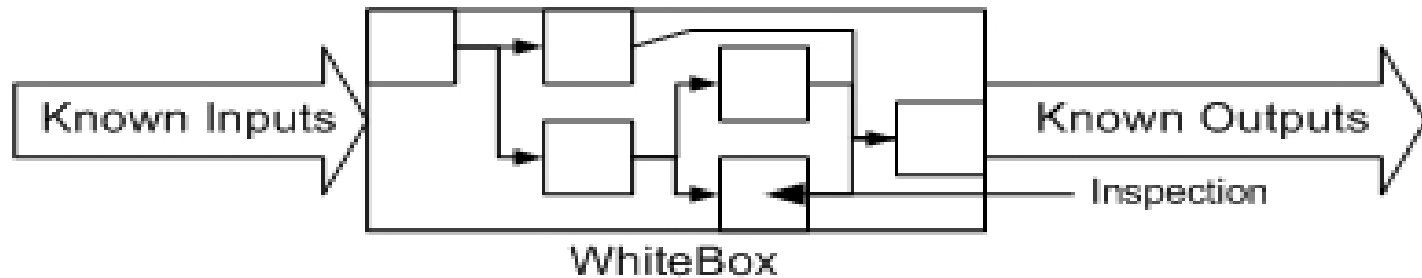
Unit Test – How ? Methodologies



A Simple Black box Specification

- ❑ Black-box testing
 - ★ Functional testing: ensure each unit acts right as its design
 - ★ Business testing: ensure the software program acts right as user requirement

Unit Test – How ? Methodologies



- White-box testing
 - Developer does himself
 - Check syntax of code by compiler to avoid syntax errors
 - Run code in debug mode
 - Examine local data structure
 - Check boundary conditions
 - Review all error handling paths
 - Apply WB Test techniques
 - See next slides

Unit Test – How ? Techniques

- Black box test (Functional)
 - ★ **Specification** derived tests
 - ★ Equivalence partitioning
 - ★ Boundary value analysis
- White box (Structural)
 - ★ Statement coverage
 - ★ Decision (branch) coverage
 - ★ Path coverage

Black box test – Specification derived test

- You can choose all or some statements in the specification of software
- Create test cases for each statements of specification
- Execute test cases to check test result will output as the specification

Example Specification

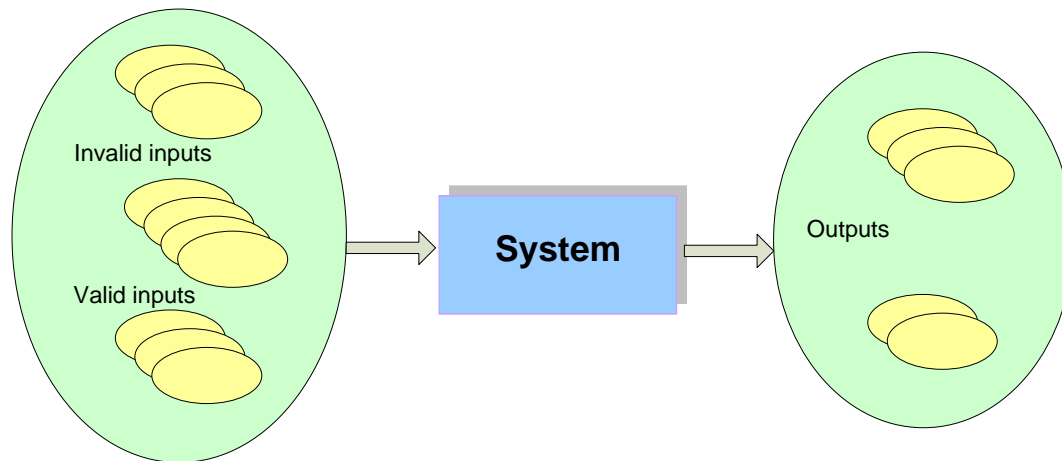
- Input - real number
- Output - real number
- When given an input of 0 or greater, the positive square root of the input shall be returned.
- When given an input of less than 0, the error message "Square root error - illegal negative input" shall be displayed and a value of 0 returned.

Example Test Cases

- **Test Case 1:** Input 4, Return 2
 - Use the first statement in the specification
 - ("When given an input of 0 or greater, the positive square root of the input shall be returned.").
- **Test Case 2:** Input -10, Return 0, Output "Square root error - illegal negative input"
 - Use the second and third statements in the specification
 - ("When given an input of less than 0, the error message "Square root error - illegal negative input" shall be displayed and a value of 0 returned.").

Black box test: Equivalence partitioning

- Divide the input of a program into **classes of data** from which test cases can be derived. This might help you to **reduce number** of test cases that must be developed.
- Behavior of software is equivalent for any value within particular partition
- A limited number of representative test cases should be chosen from each partition



Example Test Cases

Input partitions		Output partitions	
1	≥ 0	a	≥ 0
2	< 0	b	Error

- **Test Case 1:** Input 4, Return 2
 - Use the ≥ 0 input partition (1)
 - Use the ≥ 0 output partition (a)
- **Test Case 2:** Input -10, Return 0, Output "Square root error - illegal negative input"
 - Use the < 0 input partition (2)
 - Use the "Error" output partition (b)

Example Test Cases

Tuổi ứng viên	Kết quả
0-16	Không thuê
16-18	Thuê dạng bán thời gian
18-55	Thuê toàn thời gian
55-99	Không thuê

1. Testcase 1 : {Input : 2 tuổi, Output : không thuê}
2. Testcase 2 : {Input : 17 tuổi, Output : thuê bán thời gian}
3. Testcase 3 : {Input : 35 tuổi, Output : thuê toàn thời gian}
4. Testcase 4 : {Input : 90 tuổi, Output : không thuê}

Black box test: Boundary value analysis

- It is similar to equivalence partitioning, is a selection of test cases, test input that check bounding values
- Errors are most likely to exist at the boundaries between partitions
- Test the software at either side of boundary values

Example Test Cases

Input partitions		Output partitions	
1	≥ 0	a	≥ 0
2	< 0	b	Error

Example Test Cases

Tuổi ứng viên	Kết quả
0-15	Không thuê
16-17	Thuê dạng bán thời gian
18-54	Thuê toàn thời gian
55-99	Không thuê

Ta sẽ định nghĩa các testcase tương ứng với các tuổi sau : {-1,0,1}, {14,15,16}, {15,16,17}, {16,17,18}, {17,18,19}, {53,54,55}, {54,55,56}, {98,99,100}.

Có nhiều testcase trùng nhau, nếu loại bỏ các testcase trùng nhau, ta còn : -1, 0, 1, 14, 15, 16, 17, 18, 19, 53, 54, 55, 56, 98, 99, 100 (16 testcase so với hàng trăm testcase nếu vét cạn).

Example Test Cases

Sử dụng EP và BVA (10 phút)

- Bài 1: Một ngân hàng trả lãi cho khách hàng dựa vào số tiền còn lại trong tài khoản. Nếu số tiền từ 0 đến 100\$ thì trả 3% lãi, từ lớn hơn 100 \$ đến nhỏ hơn 1000\$ trả 5% lãi, từ 1000\$ trở lên trả 7% lãi.
- Bài 2: Tìm ngày kế tiếp với các ràng buộc dưới đây
 - $1 \leq \text{day} \leq 31$
 - $1 \leq \text{Month} \leq 12$
 - $1812 \leq \text{Year} \leq 2016$

PRACTICE

1. Xác định phân vùng tương đương và test case thích hợp theo yêu cầu dưới đây:
 - a. Zip Code - 5 chữ số
 - b. Last Name: từ 1 đến 15 ký tự (bao gồm chữ cái, ký tự, gạch ngang, dấu nháy, khoảng trắng và số)

Example Test Cases

- (1) Phân vùng tương đương ZIP Code
 - Nhập ký tự số
 - Nhập ký tự chữ
 - Nhập ký tự đặc biệt
 - Không nhập ký tự nào (Nếu có yêu cầu not Null thì báo lỗi ngược lại thì không làm gì)
 - Nhập < 5 ký tự
 - Nhập 5 ký tự
 - Nhập > 5 ký tự

Example Test Cases

- (2) Phân vùng tương đương Last name
 - Nhập ký tự số
 - Nhập ký tự chữ
 - Dấu chấm
 - Dấu phẩy
 - Khoảng trắng
 - Dấu nháy đơn
 - Nhập ký tự đặc biệt: Khác các ký tự dấu chấm, phẩy, khoảng trắng, dấu nháy đơn
 - Không nhập ký tự nào
 - Nhập 1 đến 15 ký tự
 - Nhập >15 ký tự

White box test:

- Statement coverage
- Branch coverage
- Path coverage

White box test: Statement coverage (phủ cấp 1)

- **Mỗi lệnh chạy ít nhất 1 lần**

```
1  float foo(int a, int b, int c, int d) {  
2      float e;  
3      if (a==0)  
4          return 0;  
5      int x = 0;  
6      if ((a==b) || ((c==d) && bug(a)))  
7          x = 1;  
8      e = 1/x;  
9      return e;  
10 }
```

1. foo(0,0,0,0), trả về 0

2. foo(1,1,1,1), trả về 1

nhưng không phát hiện lỗi chia 0
ở hàng lệnh 8

White box test: Statement coverage (phủ cấp 1)

- **Mỗi lệnh chạy ít nhất 1 lần**

```
1  float foo(int a, int b, int c, int d) {  
2      float e;  
3      if (a==0)  
4          return 0;  
5      int x = 0;  
6      if ((a==b) || ((c==d) && bug(a)))  
7          x = 1;  
8      e = 1/x;  
9      return e;  
10 }
```

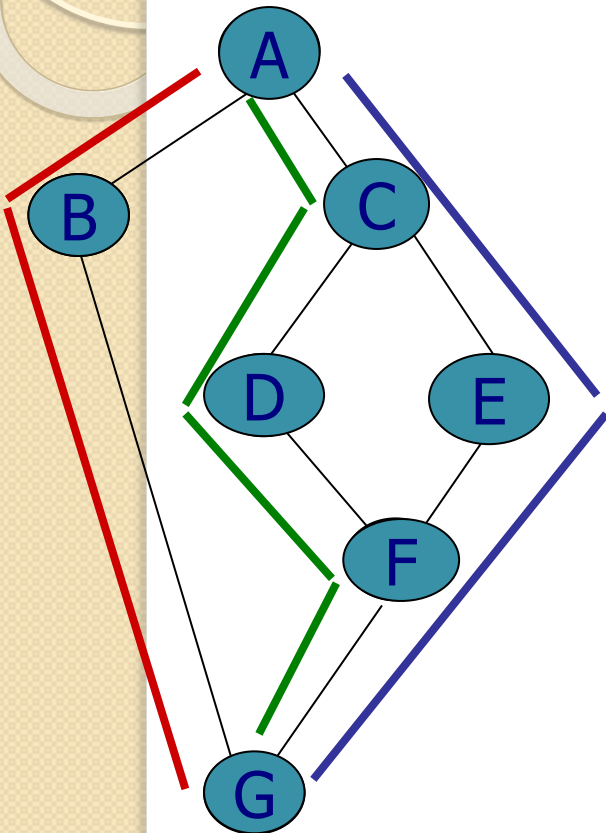
White box test: Decision (branch) coverage (phủ cấp 2)

```

1  float foo(int a, int b, int c, int d) {
2      float e;
3      if (a==0)
4          return 0;
5      int x = 0;
6      if ((a==b) || ((c==d) && bug(a)))
7          x = 1;
8      e = 1/x;
9      return e;
10 }
    
```

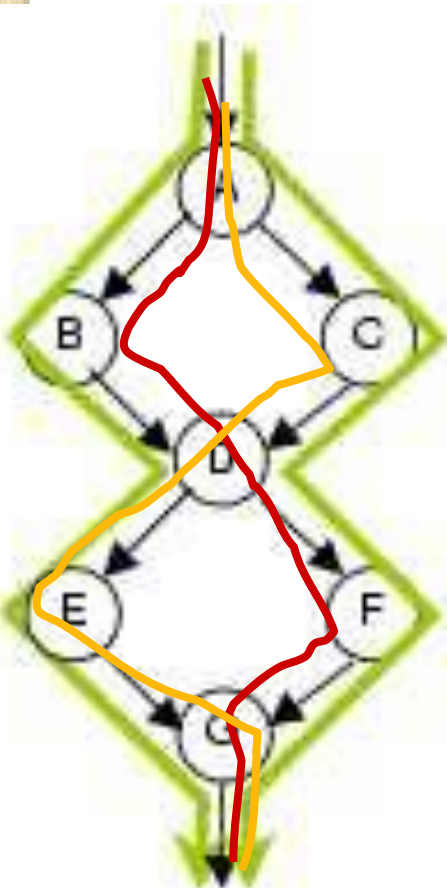
Line	Predicate	True	False
3	(a == 0)	Test Case 1 foo(0, 0, 0, 0) return 0	Test Case 2 foo(1, 1, 1, 1) return 1
6	((a==b) OR ((c == d) AND bug(a)))	Test Case 2 foo(1, 1, 1, 1) return 1	Test Case 3 foo(1, 2, 1, 2) return 1

White box test: Path coverage



- Mỗi path (đường chạy) qua chương trình đều được đi qua ít nhất một lần bởi test case nào đó

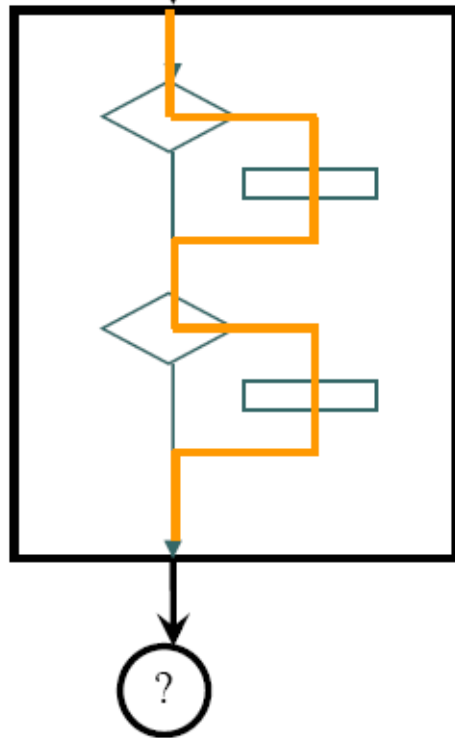
Example: White box test case



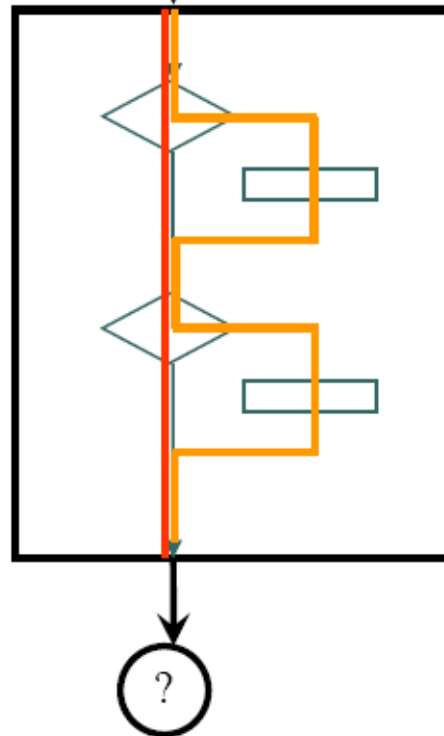
- Test cases covering ABDEG and ACDFG cover 4/4 branches (100%) and 7/7 statements (100%)
- They, however, only cover 2/4 paths (50%).
- 2 more tests are required to achieve 100% path coverage
 - ★ ABDFG
 - ★ ACDEG

White box test: Comparison

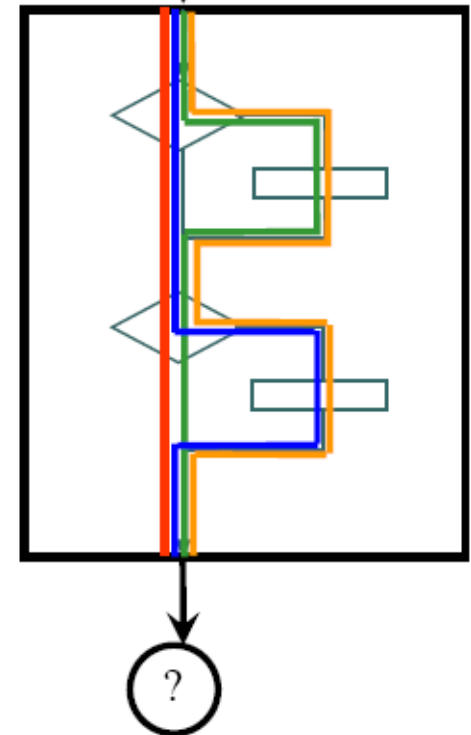
Statement



Decision



Path coverage



Practice 1

```
public bool ValidPassword(string password)
{
    bool validPassword = false;
    // Check valid password length
    if(IsValidLength(password, minLength, maxLength))
    {
        validPassword = true;
        // Check valid password mix between lowercase and upcase
        if(!IsMixedCase(password))
            return false;
        // Check valid password mix between alpha & numeric
        if(!IsAlphaNumeric(password))
            return false;
    }
    return validPassword;
}
```

How many unit test cases you must do to check this function?

Result

6: There are 4 cases:

1. Test (ValidPasswordLength)
2. Test (ValidPasswordAlphaNumeric)
3. Test (ValidPasswordMixedCase)
4. Test với điều kiện đúng của 3 trường hợp trên (ValidPasswordLengthAlphaNumericMixedCase)

Practice 1

```
char Grade(int score){  
    int res;  
    if(score < 0 || score > 10)  
        return 'I';  
    if(score>=9)  
        res = 'A';  
    else  
        if(score >=8)  
            res ='B';  
        else  
            if(score >=6.5)  
                res = 'C';  
            else  
                if(score >=5)  
                    res = 'D';  
                else  
                    res = 'F';  
    return res;  
}
```

* Statement,
Branch, Path

Practice 1

```
int LaSoNguyenTo(int n){  
    int i=2;  
    do{  
        if((n % i) == 0)  
            return 0;  
        i++;  
    } while(i <= n/2);  
    return 1;  
}
```

* Statement,
Branch, Path

Practice 1

```
int UCLN(int m, int n){  
    if (m < 0) m = -m;  
    if (n < 0) n = -n;  
    if (m == 0)    return n;  
    if (n == 0)    return m;  
    while (m != n) {  
        if(m > n)  
            m = m - n;  
        else  
            n = n - m;  
    }//end while  
    return m;  
}
```

* Statement,
Branch, Path