

Môn học  
ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

## Chương II

### TIỀN DỰ ÁN VÀ VÒNG ĐỜI DỰ ÁN (P2)

Đảm bảo chất lượng phần mềm

1

#### II.3 Giai đoạn phát triển(tiếp theo)

- ❑ Nội dung:
  1. Các hoạt động SQA
  2. Các yếu tố ảnh hưởng đến cường độ của các hoạt động SQA trong quy trình phát triển PM
  3. Rà soát phần mềm
  4. Kiểm thử phần mềm

Đảm bảo chất lượng phần mềm

3

## Software testing

### ❑ Định nghĩa:

Kiểm thử là quá trình vận hành chương trình để tìm ra lỗi trước khi chuyển giao cho người dùng cuối

(Glen Myers)

Đảm bảo chất lượng phần mềm

5

## Nội dung

- ❑ Tiền dự án
  1. Rà soát hợp đồng
  2. Kế hoạch phát triển và kế hoạch chất lượng
- ❑ Vòng đời dự án
  3. Giai đoạn phát triển(Tiếp theo)
  4. Giai đoạn bảo trì

Đảm bảo chất lượng phần mềm

2

#### II.3.4 Kiểm thử phần mềm

- A. Khái niệm cơ bản
- B. Các mức kiểm thử
- C. Kỹ thuật kiểm thử
- D. Quy trình kiểm thử
- E. Tổ chức kiểm thử

Đảm bảo chất lượng phần mềm

4

## Tại sao cần phải kiểm thử?

- ❑ Phần mềm luôn luôn có lỗi
- ❑ Thất bại phần mềm sẽ rất đắt
- ❑ Giúp kiểm tra về độ tin cậy của PM
- ❑ Đánh giá và quản lý rủi ro

Đảm bảo chất lượng phần mềm

6

## Mục tiêu kiểm thử PM

- ❑ Phát hiện ra nhiều lỗi nhất có thể
- ❑ Có một PM đạt được chất lượng ở mức có thể chấp nhận
- ❑ Thực hiện các tests cần thiết và hiệu quả trong phạm vi ngân sách và lịch biểu đã đưa ra.

Đảm bảo chất lượng phần mềm

7

## Test Case

- ❑ Gồm 3 thành phần chính:
  - Các bước thực hiện test
  - Dữ liệu test
  - Kết quả mong đợi
- ❑ Một test case tốt:
  - Hiệu quả
  - Tổng quát
  - Dễ cập nhật, sửa chữa
  - Kinh tế

Đảm bảo chất lượng phần mềm

8

## Kiểm thử triệt để (Exhaustive testing)

- ❑ Thực hiện tổ hợp tất cả đầu vào có thể có và điều kiện tiên quyết.
- ❑ Cần thực hiện một số lượng lớn các Test case → Chiếm thời gian rất lớn, không thực tế.

Đảm bảo chất lượng phần mềm

9

## Kiểm thử bao nhiêu là đủ?

- ❑ Phụ thuộc vào kết quả test
- ❑ Phụ thuộc vào các rủi ro:
  - Bỏ lỡ các faults quan trọng
  - Chi phí phát sinh không cao
  - Phá hành phần mềm chưa được test
  - Thực thi các test không hiệu quả
  - ....

Đảm bảo chất lượng phần mềm

10

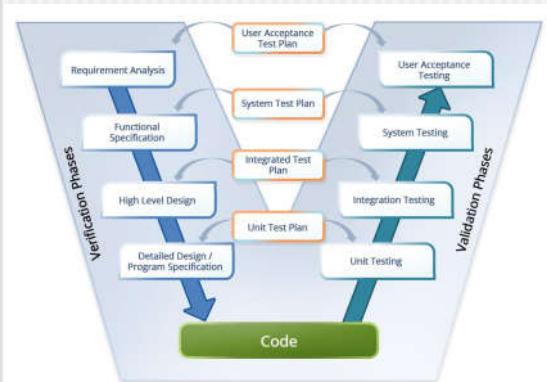
## Kiểm thử bao nhiêu là đủ?

- ❑ Sử dụng các rủi ro để quyết định
  - Cái gì cần test trước
  - Cái gì cần nhấn mạnh
  - Cái gì chưa cần test trong thời điểm hiện tại
- Thực hiện sắp xếp độ ưu tiên cho các TCs

Đảm bảo chất lượng phần mềm

11

## B.Các mức kiểm thử



Đảm bảo chất lượng phần mềm

12

## Kiểm thử đơn vị (Unit Testing)

- ❑ Mức thấp nhất
- ❑ Kiểm thử các thành phần nhỏ nhất có thể kiểm thử được như: function, class, module,...
- ❑ Các thành phần được test độc lập
- ❑ Do developer thực hiện

Đảm bảo chất lượng phần mềm

13

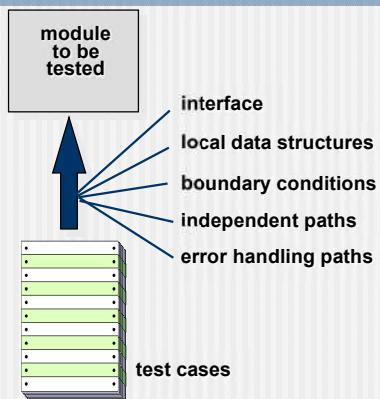
## Chiến lược kiểm thử đơn vị

- ❑ Xác định các kĩ thuật kiểm thử(white-box)
- ❑ Đưa ra các tiêu chí để hoàn thành test
- ❑ Xác định mức độ độc lập khi thiết kế test
- ❑ Lập tài liệu về qui trình test và các hoạt động test(inputs và outputs)
- ❑ Những hoạt động lặp đi lặp lại sau mỗi lần fix lỗi hoặc thay đổi

Đảm bảo chất lượng phần mềm

14

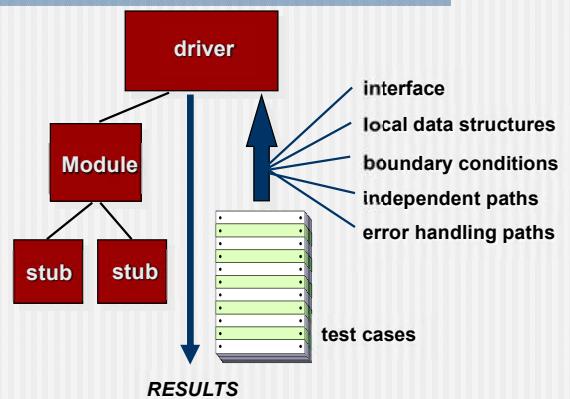
## Nội dung kiểm thử đơn vị



Đảm bảo chất lượng phần mềm

15

## Môi trường kiểm thử đơn vị



Đảm bảo chất lượng phần mềm

16

## Kiểm thử tích hợp (Integration Testing)

- ❑ Tích hợp các thành phần phụ thuộc đã được test
- ❑ Tập trung tìm các lỗi:
  - Thiết kế và xây dựng kiến trúc PM
  - Các thành phần được tích hợp ở mức sub-system
  - Giao tiếp giữa các thành phần
- ❑ Do developers/testers thực hiện

Đảm bảo chất lượng phần mềm

17

## Chiến lược kiểm thử tích hợp

- ❑ Có 2 hướng tiếp cận cơ bản:
  - Big-bang
  - Incremental (top-down, bottom-up)



Đảm bảo chất lượng phần mềm

18

## Chiến lược kiểm thử tích hợp

### ❑ Big-bang:

- Tại sao chúng ta lại kết hợp tất cả các thành phần và test cùng 1 lúc? → tiết kiệm thời gian
- Trong thực tế:
  - Mất nhiều thời gian để tìm lỗi và sửa lỗi
  - Test lại sau khi sửa lỗi sẽ phức tạp hơn nhiều

Đảm bảo chất lượng phần mềm

19

## Chiến lược kiểm thử tích hợp

### ❑ Incremental

- Mức 0: các thành phần đã được test
- Mức 1: 2 thành phần
- Mức 2: 3 thành phần,
- ....

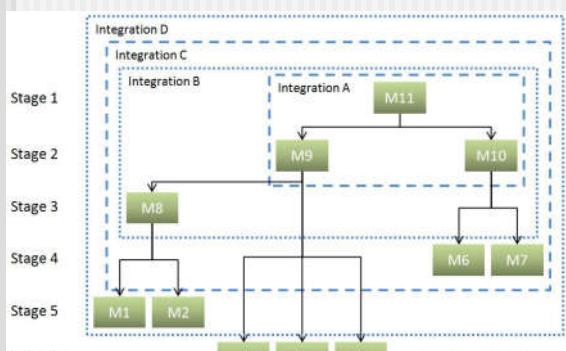
- ❑ Giúp tìm ra các khuyết điểm sớm → sửa lỗi sớm

- ❑ Dễ dàng phục hồi sau lỗi

Đảm bảo chất lượng phần mềm

20

## Tích hợp Top-down



Đảm bảo chất lượng phần mềm

21

## Tích hợp top-down

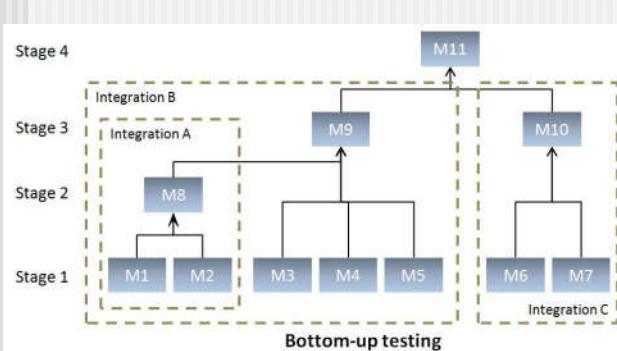
### Ưu điểm

- ❑ Cấu trúc điều khiển quan trọng được test trước
- ❑ Hàm I/O được gọi sớm → viết test dễ
- ❑ Mô phỏng chức năng chính của PM sớm → nổi bật vấn đề liên quan đến yêu cầu

### Khuyết điểm

- ❑ Phụ thuộc vào nhiều stub
- ❑ Khó khăn khi phân tích kết quả test

## Tích hợp Bottom-up



Đảm bảo chất lượng phần mềm

23

## Tích hợp Bottom-up

### Ưu điểm

- ❑ Các mức thấp nhất được test đầu tiên
- ❑ Điều kiện test được tạo dễ dàng
- ❑ Dễ quan sát các kết quả test chi tiết

### Khuyết điểm

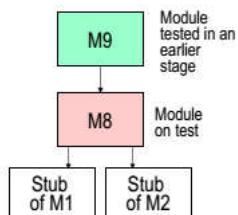
- ❑ Phải tạo các driver
- ❑ Chương trình tổng thể được quan sát trễ

Đảm bảo chất lượng phần mềm

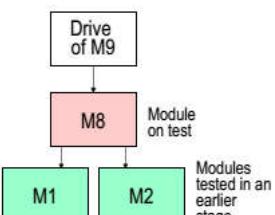
24

## Stubs and Drivers

Top-down testing of module M8



Bottom-up testing of module M8



Đảm bảo chất lượng phần mềm

25

## Re-testing and regression testing

### ❑ Regression testing:

- Chạy lại tất cả các test cases đã được thực thi trước đó
- Phiên bản mới có thể bao gồm các khuyết cũ được fix, và các khuyết mới
- Có thể sử dụng các công cụ test tự động

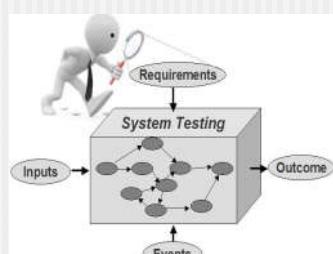
Đảm bảo chất lượng phần mềm

27

## Functional System Testing

### ❑ Tìm các lỗi:

- Input/Output
- Giao diện người dùng
- Giao tiếp của hệ thống với các phần khác
- Hành vi của hệ thống



Đảm bảo chất lượng phần mềm

29

## Re-testing and regression testing

### ❑ Re-testing:

- Chạy lại các test cases không thành công,
- Phiên bản mới chỉ bao gồm các khuyết cũ được fix
- Chạy lại chính xác test

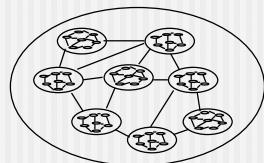
Đảm bảo chất lượng phần mềm

26

## System testing

### ❑ Là bước tích hợp cuối cùng

- ❑ Kiểm thử chức năng (Functional)
- ❑ Kiểm thử phi chức năng (Non-Functional)
- ❑ Do các tester thực hiện



Đảm bảo chất lượng phần mềm

28

## Non-functional System Testing

### ❑ Usability

- ❑ Security
- ❑ Documentation
- ❑ Storage
- ❑ Volume

### ❑ Configuration / installation

- ❑ Reliability
- ❑ Recovery
- ❑ Performance, load, stress

Đảm bảo chất lượng phần mềm

30

## Usability Testing

- ❑ Đơn giản, hiệu quả khi sử dụng
- ❑ Giao diện nhất quán và phù hợp
- ❑ Message phù hợp và có ý nghĩa cho người sử dụng
- ❑ Hỗ trợ thông tin phản hồi
- ❑ Liên kết tắt

Đảm bảo chất lượng phần mềm

31

## Security Testing

- ❑ Kiểm tra cơ chế bảo mật của hệ thống có hiệu quả không?
- ❑ Tester sẽ đóng vai trò là hacker
- ❑ Bài toán thiết kế hệ thống an ninh là:
  - Chi phí công cụ bảo vệ **nhỏ hơn** lợi ích bảo vệ khỏi đột nhập
  - Chi phí đột nhập **lớn hơn** lợi ích thu được từ đột nhập

Đảm bảo chất lượng phần mềm

32

## Security Testing

- ❑ Ví dụ:
  - Passwords, encryption
  - Mức độ truy cập thông tin, quyền
- ❑ Một số kỹ thuật test lỗ hổng của ứng dụng Web:
  - SQL Injection
  - Cross-site Scripting(XSS)
  - ...

Đảm bảo chất lượng phần mềm

33

## Documentation Testing

- ❑ Rà soát tài liệu
  - Kiểm tra định dạng, lỗi chính tả, ....
  - Độ chính xác về nội dung
- ❑ Kiểm tra tài liệu
  - Có làm việc không?
  - Tài liệu bảo trì
  - Hướng dẫn sử dụng

Đảm bảo chất lượng phần mềm

34

## Performance Testing

- ❑ Xác định tốc độ, khả năng phân tải
- ❑ Tìm điểm “thắt cổ chai” → cải tiến nâng cao khả năng hoạt động của PM
- ❑ Timing Tests
  - Thời gian phục vụ và đáp ứng
  - Thời gian phục hồi CSDL
- ❑ Capacity & Volume Tests
  - Khối lượng/kích thước dữ liệu lớn nhất khả năng xử lý được

Đảm bảo chất lượng phần mềm

35

## Stress/Load Testing

- ❑ Vận hành hệ thống khi sử dụng nguồn lực với số lượng, tần suất lớn
- ❑ **Load Testing:**
  - kiểm tra PM ở điều kiện liên tục tăng mức độ chịu tải, nhưng PM vẫn hoạt động được
- ❑ **Stress Testing:**
  - Kiểm tra PM ở trạng thái vận hành trong điều kiện bất thường

Đảm bảo chất lượng phần mềm

36

## Configuration/Installation Testing

- ❑ Configuration tests
  - Môi trường phần cứng, phần mềm khác nhau
  - Nâng cấp 1 phần của hệ thống có thể dẫn đến xung đột với phần khác
- ❑ Installation Tests
  - PM có thể cài đặt qua CD, networks,...
  - Thời gian cài đặt
  - Uninstall

Đảm bảo chất lượng phần mềm

37

## Reliability Testing

- ❑ Mean Time Between Failures (MTBF)  
để đo độ tin cậy
- ❑ Tạo ra một tập test đại diện, thu thập đủ thông tin để thống kê tỉ lệ thất bại

Đảm bảo chất lượng phần mềm

38

## Recovery Testing

- ❑ Bắt phần mềm phải thất bại để xem khả năng phục hồi của nó đến đâu
- ❑ Có 2 cách phục hồi
  - Phục hồi tự động(back-up)
  - Phục hồi dựa trên sự can thiệp của con người

Đảm bảo chất lượng phần mềm

39

## User acceptance testing

- ❑ Thẩm định xem các chức năng của PM có thỏa mãn sự mong đợi của khách hàng không?
- ❑ Do customer thực hiện

Đảm bảo chất lượng phần mềm

40

## Tại sao người dùng nên tham gia test?

- ❑ Người dùng họ biết:
  - Những tình huống nghiệp vụ xảy ra trong thực tế
  - Cách người dùng thực hiện công việc của mình khi sử dụng hệ thống
  - Trường hợp đặc biệt có thể gây ra vấn đề
- ❑ Lợi ích:
  - Họ sẽ hiểu chi tiết về hệ thống mới

Đảm bảo chất lượng phần mềm

41

## Kiểm thử Alpha và Beta

- ❑ Do người sử dụng đầu cuối thực hiện, không phải người đặt hàng
- ❑ Họ sẽ đưa ra các feedback về sản phẩm(phát hiện lỗi, đề nghị cải tiến,...)

Đảm bảo chất lượng phần mềm

42

## Kiểm thử Alpha

- ❑ Do bên phát triển PM tiến hành
- ❑ Được tiến hành trong môi trường được điều khiển
- ❑ Dữ liệu thường là mô phỏng

Đảm bảo chất lượng phần mềm

43

## Kiểm thử Beta

- ❑ Do bên khách hàng tiến hành
- ❑ Được tiến hành trong môi trường thực, không có sự kiểm soát của Dev
- ❑ Khách hàng sẽ báo cáo tất cả các vấn đề trong quá trình test một cách định kì

Đảm bảo chất lượng phần mềm

44

## C.Kĩ thuật kiểm thử

- ❑ Static Testing
- ❑ Dynamic Testing

Đảm bảo chất lượng phần mềm

45

## Kĩ thuật kiểm thử

- ❑ Là thủ tục chọn lựa hoặc thiết kế test tốt
- ❑ Dựa trên cấu trúc bên trong hoặc đặc tả của hệ thống PM
- ❑ Giúp tìm ra được nhiều lỗi
- ❑ Giúp đo nỗ lực test

Đảm bảo chất lượng phần mềm

46

## Ưu điểm của các kĩ thuật

- ❑ Xác suất tìm ra lỗi của các tester là như nhau
- ❑ Kiểm thử hiệu quả hơn: Tìm ra được nhiều lỗi hơn với chi phí ít
  - Tập trung vào các loại lỗi nhất định
  - Kiểm thử đúng hướng
  - Tránh trùng lặp

Đảm bảo chất lượng phần mềm

47

## Static Testing

- ❑ Là kĩ thuật kiểm thử mà không thực thi code
- ❑ Kiểm thử dựa trên công cụ (**static code analysis**)
  - Data Flow Analysis
  - Control Flow Analysis

Đảm bảo chất lượng phần mềm

48

## Phân tích luồng dữ liệu

- ❑ Xem xét các biến trong chương trình
  - Khai báo, định nghĩa biến
  - Sử dụng biến
  - Biến phải được khai báo trước khi sử dụng
  - Tham chiếu tới biến chưa được gán giá trị(define)
  - Biến chưa bao giờ được sử dụng
  - Phạm vi tồn tại của biến

Đảm bảo chất lượng phần mềm

49

## Phân tích luồng dữ liệu

1. n := 0
  2. read (x)
  3. n := 1
  4. while x > y do
  5. begin
  6. read (y)
  7. write( n\*y)
  8. x := x - n
  9. end
- Luồng dữ liệu bắt thường  
n : được định nghĩa lại và ko  
được sử dụng*
- Lối luồng dữ liệu  
y: được sử dụng mà chưa  
được định nghĩa(gán giá trị)*

Đảm bảo chất lượng phần mềm

50

## Phân tích luồng điều khiển

- ❑ Vòng lặp vô hạn, đệ quy vô hạn
- ❑ Mã “chết”
- ❑ Nhảy đến nhãn không xác định
- ❑ Độ phức tạp của lưu đồ
- ❑ ....

Đảm bảo chất lượng phần mềm

51

## Unreachable code

```
int f(int x, int y)    double x = sqrt(2);  
{                      if (x > 5)  
    return x+y;        {  
    int z=x*y;          printf("%d",x);  
}
```

Đảm bảo chất lượng phần mềm

52

## Ưu điểm và nhược điểm

- ❑ **Ưu điểm:**
  - Tìm ra khuyết khuyết khó thấy
  - Đưa ra những đánh giá về chất lượng code
- ❑ **Khuyết điểm:**
  - Không phân biệt được “fail-safe” code với lỗi thực khi lập trình → nhiều fail message
  - Không thực thi code

Đảm bảo chất lượng phần mềm

53

## Công cụ

- ❑ **Multi-language**
  - [Moose](#) : C/C++, Java, Smalltalk, .NET,...
  - [Copy/Paste Detector \(CPD\)](#): [Java](#), [JSP](#), [C](#), [C++](#) and [PHP](#) code.
  - [CheckKing](#): Java, JSP, Javascript, HTML, XML, .NET, PL/SQL, embedded SQL, C/C++, Cobol,...

Đảm bảo chất lượng phần mềm

54

## Công cụ

- ❑ C/C++
  - [Cppcheck](#), [Frama-C](#), ...
- ❑ Java
  - [Checkstyle](#), [FindBugs](#), [Jtest](#), ...

Đảm bảo chất lượng phần mềm

55

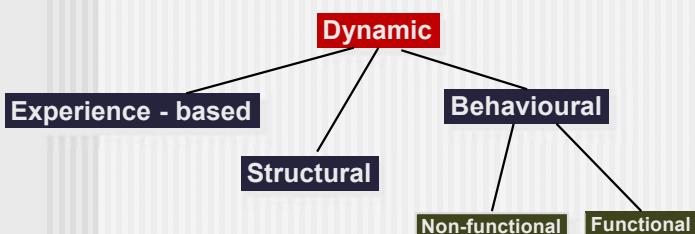
## Dynamic Testing

- ❑ Là kĩ thuật kiểm thử dựa trên thực thi code
- ❑ Tại sao phải sử dụng các kĩ thuật kiểm thử động?
  - Kiểm thử triệt để(Exhaustive testing ) không thực tế
    - Chỉ sử dụng tập nhỏ nhất các test cases
    - Xác suất tìm ra lỗi lớn nhất
  - Cần chọn lựa các tests sử dụng các kĩ thuật thiết kế test cases

Đảm bảo chất lượng phần mềm

56

## Các kĩ thuật kiểm thử động

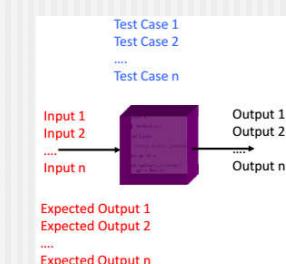


Đảm bảo chất lượng phần mềm

57

## Black-box testing

- ❑ Là phương pháp kiểm thử dựa trên hành vi hoặc chức năng của PM



Đảm bảo chất lượng phần mềm

58

## Black-box testing

- ❑ Bao gồm các kĩ thuật:
  - Equivalence partitioning
  - Boundary value analysis
  - State transition testing
  - Decision table testing

Đảm bảo chất lượng phần mềm

59

## Phân hoạch lớp tương đương

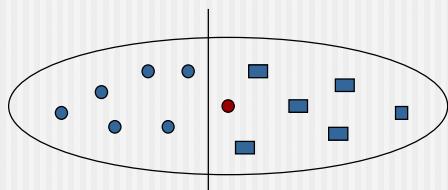
- 
- A horizontal blue line represents a range of values. Two vertical red lines mark the boundaries of partitions. The first partition is labeled 'invalid' at both ends. The second partition is labeled 'valid' at both ends. The third partition is labeled 'invalid' at both ends. Red numbers '0', '1', '100', and '101' are placed on the line to indicate specific points. Below the line, there are two sets of labels: 'invalid' and 'valid' with arrows pointing to the respective regions, and '0', '1', '100', and '101' with arrows pointing to their corresponding positions on the line.
- ❑ Phân chia tập hợp các điều kiện test (dữ liệu đầu vào) thành những vùng/lớp tương đương nhau.
  - ❑ Lớp tương đương(equivalence partitions)
    - 2 tests thuộc cùng lớp tương đương nếu kết quả mong đợi của nó giống nhau
    - → thực hiện nhiều tests thuộc cùng 1 lớp tương đương thì dẫn đến dư thừa

Đảm bảo chất lượng phần mềm

60

## Phân tích giá trị biên

- Biên(boundary)** là điểm chuyển từ lớp tương đương này sang lớp tương đương khác
- Tìm ra được lỗi gần biên



Đảm bảo chất lượng phần mềm

61

## Ví dụ:

|                       |                      |                        |
|-----------------------|----------------------|------------------------|
| Customer Name         | <input type="text"/> | 2-64 chars.            |
| Account number        | <input type="text"/> | 6 digits, 1st non-zero |
| Loan amount requested | <input type="text"/> | £500 to £9000          |
| Term of loan          | <input type="text"/> | 1 to 30 years          |
| Monthly repayment     | <input type="text"/> | Minimum £10            |
| Term:                 |                      |                        |
| Repayment:            |                      |                        |
| Interest rate:        |                      |                        |
| Total paid back:      |                      |                        |

Đảm bảo chất lượng phần mềm

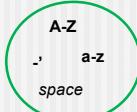
62

## Customer name

Số lượng kí tự:



Các kí tự hợp lệ:



| Conditions    | Valid Partitions             | Invalid Partitions                       | Valid Boundaries    | Invalid Boundaries             |
|---------------|------------------------------|--|---------------------|--------------------------------|
| Customer name | 2 to 64 chars<br>valid chars | < 2 chars<br>> 64 chars<br>invalid chars | 2 chars<br>64 chars | 1 chars<br>65 chars<br>0 chars |

Đảm bảo chất lượng phần mềm

63

## Account number

Chữ số đầu tiên:

valid: non-zero  
invalid: zero

Số lượng chữ số:

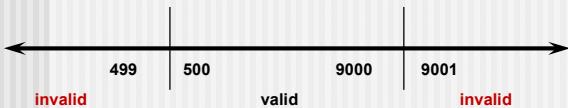


| Conditions     | Valid Partitions         | Invalid Partitions                                     | Valid Boundaries | Invalid Boundaries               |
|----------------|--------------------------|--|------------------|----------------------------------|
| Account number | 6 digits<br>1st non-zero | < 6 digits<br>> 6 digits<br>1st digit = 0<br>non-digit | 100000<br>999999 | 5 digits<br>7 digits<br>0 digits |

Đảm bảo chất lượng phần mềm

64

## Loan amount



| Conditions  | Valid Partitions | Invalid Partitions                         | Valid Boundaries | Invalid Boundaries |
|-------------|------------------|--|------------------|--------------------|
| Loan amount | 500 - 9000       | < 500<br>>9000<br>0<br>non-numeric<br>null | 500<br>9000      | 499<br>9001        |

Đảm bảo chất lượng phần mềm

65

## Điều kiện test

| Conditions     | Valid Partitions            | Tag      | Invalid Partitions                                     | Tag                           | Valid Boundaries    | Tag      | Invalid Boundaries               | Tag            |
|----------------|-----------------------------|----------|--|-------------------------------|---------------------|----------|----------------------------------|----------------|
| Customer name  | 2 - 64 chars<br>valid chars | V1<br>V2 | < 2 chars<br>> 64 chars<br>invalid char                | X1<br>X2<br>X3                | 2 chars<br>64 chars | B1<br>B2 | 1 char<br>65 chars<br>0 chars    | D1<br>D2<br>D3 |
| Account number | 6 digits<br>1st non-zero    | V3<br>V4 | < 6 digits<br>> 6 digits<br>1st digit = 0<br>non-digit | X4<br>X5<br>X6<br>X7          | 100000<br>999999    | B3<br>B4 | 5 digits<br>7 digits<br>0 digits | D4<br>D5<br>D6 |
| Loan amount    | 500 - 9000                  | V5       | < 500<br>>9000<br>0<br>non-integer<br>null             | X8<br>X9<br>X10<br>X11<br>X12 | 500<br>9000         | B5<br>B6 | 499<br>9001                      | D7<br>D8       |

Đảm bảo chất lượng phần mềm

66

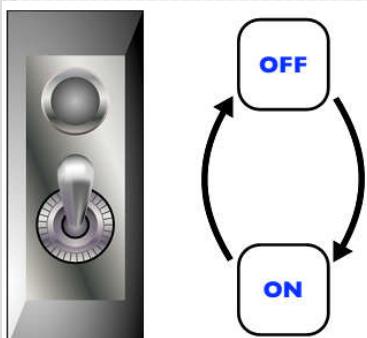
## Thiết kế test cases

| Test Case | Input   | Expected Outcome   | New Tags Covered               |
|-----------|---|--|--------------------------------|
| 1         | Name: John Smith<br>Acc no: 123456<br>Loan: 2500<br>Term: 3 years | Term: 3 years<br>Repayment: 79.86<br>Interest rate: 10%<br>Total paid: 2874.96 | V1, V2,<br>V3, V4,<br>V5 ..... |
| 2         | Name: AB<br>Acc no: 100000<br>Loan: 500<br>Term: 1 year           | Term: 1 year<br>Repayment: 44.80<br>Interest rate: 7.5%<br>Total paid: 537.60  | B1, B3,<br>B5, .....           |

Đảm bảo chất lượng phần mềm

67

## Ví dụ 1:



| TEST 1                     | TEST 2                      |
|----------------------------|-----------------------------|
| start state:<br><b>off</b> | start state:<br><b>on</b>   |
| input:<br><b>switch on</b> | input:<br><b>switch off</b> |
| output:<br><b>light on</b> | output:<br><b>light off</b> |
| finish state:<br><b>on</b> | finish state:<br><b>off</b> |

Đảm bảo chất lượng phần mềm

69

## Ví dụ 2: Media Player

| Test Case No. | Start State | Event/Input        | End State/Exp Output |
|---------------|-------------|--------------------|----------------------|
| 1             | Stopped     | Press play button  | Play                 |
| 2             | Stopped     | Press pause button | Stopped              |
| 3             | Play        | Press stop button  | Stopped              |
| 4             | Play        | Press pause button | Paused               |
| 5             | Paused      | Press stop button  | Stopped              |
| 6             | Paused      | Press play button  | Play                 |

Đảm bảo chất lượng phần mềm

71

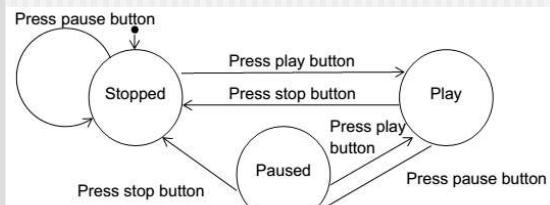
## Chuyển đổi trạng thái (state transition testing)

- Phân tích mối quan hệ giữa các trạng thái, sự kiện, hành động(action) gây ra sự chuyển đổi từ trạng thái này sang trạng thái khác
- Các bước thực hiện**
  - Xác định tất cả các trạng thái
  - Với mỗi test, xác định
    - Start state
    - Input
    - Expected Output/End state

Đảm bảo chất lượng phần mềm

68

## Ví dụ 2: Media Player



Đảm bảo chất lượng phần mềm

70

## Bảng quyết định (Decision Table)

### ▫ Gồm 3 phần:

- Conditions**
  - Các điều kiện đầu vào để ra quyết định
- Actions**
  - Kết quả của sự kết hợp các điều kiện đầu vào
- Rules**
  - Luật kết hợp các điều kiện

Đảm bảo chất lượng phần mềm

72

## Bảng quyết định (Decision Table)

### ❑ Các bước thực hiện tạo bảng:

1. Liệt kê tất cả các **conditions/inputs** và xác định giá trị của nó
2. Liệt kê tất cả các **actions**
3. Tính số **rules lớn nhất**
4. Xác định các **rules** từ sự tổ hợp giá trị của các **conditions/inputs**
5. Đánh dấu "X" vào mỗi **actions** tương ứng với mỗi **rule**
6. Rút gọn bảng (nếu có thể)

Đảm bảo chất lượng phần mềm

73

## Ví dụ 1:

Một ngân hàng sử dụng những luật sau để phân loại tài khoản mới của người gửi tiền:

- Nếu tuổi  $\geq 21$  và số tiền gửi là  $\geq 100$  nghìn, thì loại tài khoản là A
- Nếu tuổi  $< 21$  và số tiền gửi là  $\geq 100$  nghìn, thì loại tài khoản là B
- Nếu tuổi  $\geq 21$  và số tiền gửi là  $< 100$  nghìn, thì loại tài khoản là C
- Nếu tuổi  $< 21$  và số tiền gửi là  $< 100$  nghìn, thì không mở tài khoản

Đảm bảo chất lượng phần mềm

74

## Ví dụ 1:

|    | Condition/Action      | R1 | R2 | R3 | R4 |
|----|-----------------------|----|----|----|----|
| C1 | Tuổi $\geq 21$        | Y  | Y  | N  | N  |
| C2 | Tiền $\geq 100$ nghìn | Y  | N  | Y  | N  |
| A1 | A                     | X  | -  | -  | -  |
| A2 | B                     | -  |    | X  | -  |
| A3 | C                     | -  | X  | -  | -  |
| A4 | Ko thẻ mở TK          | -  | -  | -  | X  |

Đảm bảo chất lượng phần mềm

75

## Ví dụ 2:

### ❑ Để tính thuế thu nhập, người ta có mô tả sau:

- Người vô gia cư nộp 4% thuế thu nhập
  - Người có nhà ở nộp thuế theo bảng sau:
- |                       |      |
|-----------------------|------|
| Tổng thu nhập         | Thuế |
| $\leq 5.000.000$ đồng | 4%   |
| $> 5.000.000$ đồng    | 6%   |

Đảm bảo chất lượng phần mềm

76

## Ví dụ 2:

|    | Condition/Action            | R1 | R2 | R3 | R4 |
|----|-----------------------------|----|----|----|----|
| C1 | Người có nhà ở              | Y  | Y  | N  | N  |
| C2 | Tổng thu nhập $\leq 500000$ | Y  | N  | Y  | N  |
| A1 | Nộp thuế 4%                 | X  | -  | X  | X  |
| A2 | Nộp thuế 6%                 | -  | X  | -  | -  |

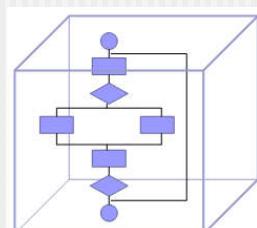
|    | Condition/Action            | R1 | R2 | R3 |
|----|-----------------------------|----|----|----|
| C1 | Người có nhà ở              | Y  | Y  | N  |
| C2 | Tổng thu nhập $\leq 500000$ | Y  | N  | -  |
| A1 | Nộp thuế 4%                 | X  | -  | X  |
| A2 | Nộp thuế 6%                 | -  | X  | -  |

Đảm bảo chất lượng phần mềm

77

## White-box testing

### ❑ Là phương pháp kiểm thử dựa trên cấu trúc logic của PM



Đảm bảo chất lượng phần mềm

78

## White-box testing

- ❑ Control Flow Testing
  - Statement testing
  - Branch / Decision testing
  - Branch condition testing
  - Branch condition combination testing
- ❑ Data flow testing

Đảm bảo chất lượng phần mềm

79

## Kĩ thuật độ bao phủ cấu trúc (Coverage techniques)

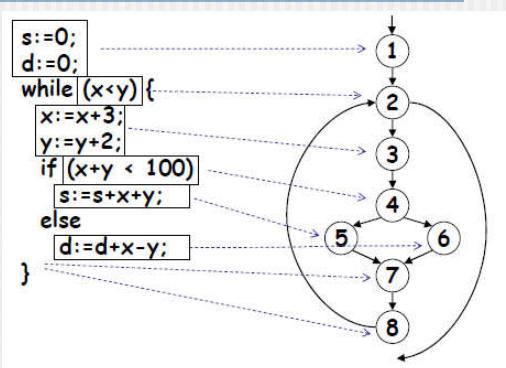
- ❑ Đánh giá độ bao phủ của các thành phần cấu trúc
- ❑ Thiết kế các test bổ sung → có thể đạt được mức bao phủ 100%

Độ bao phủ 100%  
không có nghĩa là  
100% được test

Đảm bảo chất lượng phần mềm

81

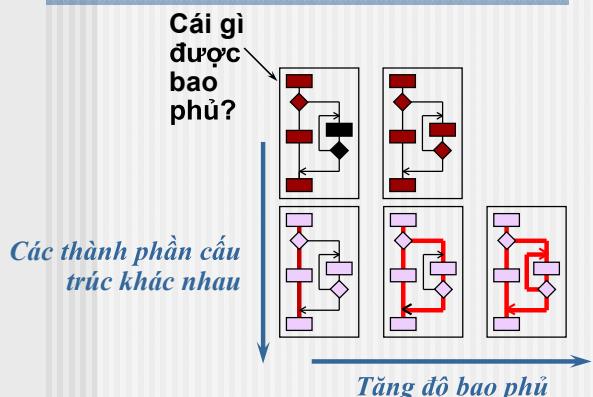
## Ví dụ 1:



Đảm bảo chất lượng phần mềm

83

## Kĩ thuật độ bao phủ cấu trúc (Coverage techniques)



Đảm bảo chất lượng phần mềm

80

## Kiểm thử luồng điều khiển (Control Flow Testing)

- ❑ Bước 1:
  - Xây dựng đồ thị luồng từ source code
  - Đồ thị bao gồm nút, cạnh
- ❑ Bước 2:
  - Thiết kế các Test cases để bao phủ hết các thành phần của đồ thị

Đảm bảo chất lượng phần mềm

82

## Thành phần của CFG

- ❑ Có 3 loại nút
  - Statement nodes
  - Decision nodes
  - Auxiliary nodes

Đảm bảo chất lượng phần mềm

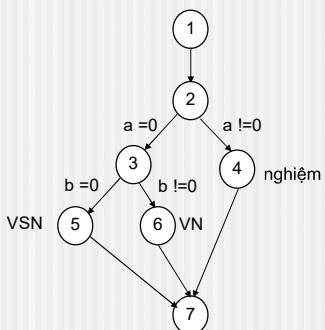
84

## Ví dụ 2:

Đồ thị luồng điều khiển của đoạn chương trình giải phương trình bậc nhất

```

1. if(a == 0)
2.   if(b == 0)
3.     cout<<"Vo so nghiem";
4.   else
5.     cout<<"Vo nghiem";
6. else
7.   cout<<"x = "<<-b/a;
    
```



Đảm bảo chất lượng phần mềm

85

## Độ phức tạp “Cyclomatic”

$$\blacksquare V(G) = E - N + 2$$

$$\blacksquare V(G) = P + 1$$

$$\blacksquare V(G) = R$$

Trong đó:

- E: số cạnh
- N: số nút
- P: số nút điều kiện
- R: số miền

Đảm bảo chất lượng phần mềm

86

## Vẽ đồ thị luồng

```

1. void prime(int n){
2.   cout<<"Cac so nguyen to nho hon n la:"<<endl;
3.   for(int i = 2; i < n;i++){
4.     int flag = 1;
5.     for(int j = 2; j <= sqrt(i);j++)
6.       if(i%j==0)
7.         { flag = 0; break;}
8.     if(flag == 1){
9.       cout<<i<<"\t";
10.    }
11.  }
12. }
    
```

Đảm bảo chất lượng phần mềm

87

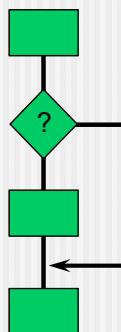
## Bao phủ câu lệnh (Statement coverage)

- Tỉ lệ phần trăm các câu lệnh được thực thi bởi bộ test

$$= \frac{\text{Số câu lệnh thực thi}}{\text{Tổng số câu lệnh}}$$

- Ví dụ:

- Chương trình có 100 câu lệnh
- tests thực thi 87 câu lệnh
- Bao phủ câu lệnh = 87%



Đảm bảo chất lượng phần mềm

88

## Ví dụ 1:

```

1 cin>>a;
2 if (a > 6)
3   b = a;
4 cout<<b;
    
```

Số câu lệnh

| Test case | Input | Expected output |
|-----------|-------|-----------------|
| 1         | 7     | 7               |

Với 1 test case này thì tất cả 4 câu lệnh đều được thực thi  
→ đạt độ bao phủ dòng lệnh 100%

Đảm bảo chất lượng phần mềm

89

## Bài tập 1

Đoạn chương trình giải phương trình bậc nhất  $ax+b=0$

```

1. {
2.   if(a == 0)
3.     if(b == 0)
4.       cout<<"Vo so nghiem";
5.     else
6.       cout<<"Vo nghiem";
7.   else
8.     cout<<"x = "<<-b/a;
9. }
    
```

1. Có bao nhiêu câu lệnh?
2. Test case ( $a=0, b=8$ ) bao phủ bao nhiêu % câu lệnh ?
3. Cần tối thiểu bao nhiêu test case để bao phủ 100% các câu lệnh ?

Đảm bảo chất lượng phần mềm

90

## Bài tập 2

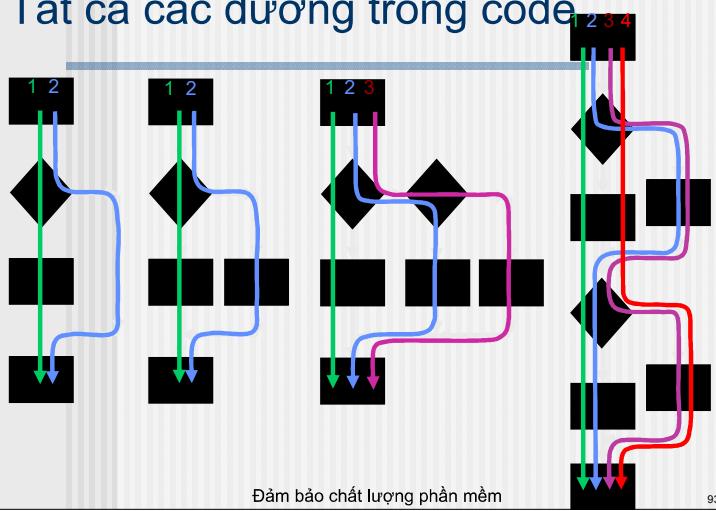
```
1. void func(int A, int B, int X)
2. {
3.     if (A > 1 && B == 0)
4.         X = X / A;
5.     if (A == 2 || X > 1)
6.         X = X + 1;
7. }
```

1. Có bao nhiêu câu lệnh ?
2. Test case (A=2,B=1,X=3) bao phủ bao nhiêu % câu lệnh ?
3. Test case (A=3,B=0,X=0) bao phủ bao nhiêu % câu lệnh ?
4. Tối thiểu bao nhiêu test case để bao phủ 100% các câu lệnh ?

Đảm bảo chất lượng phần mềm

91

## Tất cả các đường trong code



Đảm bảo chất lượng phần mềm

93

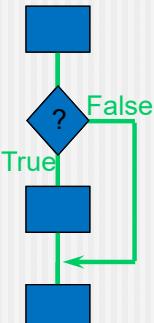
## Bao phủ nhánh (Decision/Branch coverage)

- ❑ Tỉ lệ phần trăm các nhánh được thực thi bởi bộ test

$$= \frac{\text{Số nhánh được thực thi}}{\text{Tổng số nhánh}}$$

- ❑ Ví dụ:

- Chương trình có 120 nhánh
- tests thực thi 60 nhánh
- Bao phủ nhánh = 50%



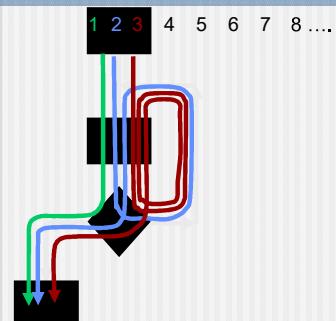
True

False

Đảm bảo chất lượng phần mềm

92

## Các đường trong vòng lặp



Đảm bảo chất lượng phần mềm

94

## Ví dụ 1:

Đoạn chương trình giải phương trình bậc nhất  $ax+b=0$

```
1. {
2.     if(a == 0)
3.         if(b == 0)
4.             cout<<"Vo so nghiem";
5.         else
6.             cout<<"Vo nghiem";
7.     else
8.         cout<<"x =" ,-b/a;
9. }
```

1. Có bao nhiêu nhánh ?
2. Test case (a=0,b=8),(a=0,b=0) bao phủ bao nhiêu % nhánh ?
3. Tối thiểu bao nhiêu test case để bao phủ 100% các nhánh ?

Đảm bảo chất lượng phần mềm

95

## Ví dụ 2:

```
void func(int A, int B, int X)
```

```
1. {
2.     if ( A > 1 && B == 0 )
3.         X = X / A;
4.     if ( A == 2 || X > 1 )
5.         X = X + 1;
6. }
```

1. Có bao nhiêu nhánh ?
2. Test case (A=2,B=0,X=3) bao phủ bao nhiêu % nhánh ?
3. Tối thiểu bao nhiêu test case để bao phủ 100% các nhánh ?

Đảm bảo chất lượng phần mềm

96

## Ví dụ

```

1. cin>>A;
2. cin>> B;
3. if (A > 0)
4.   if (B == 0)
5.     cout<< "No values";
6.   else
7.   {
8.     cout<<"B"<<endl;
9.     if (A > 21)
10.      cout<<"A";
11. }

```

Đảm bảo chất lượng phần mềm

98

## Bao phủ điều kiện (Condition coverage)

Tỉ lệ phần trăm các điều kiện được thực thi bởi bộ test

$$= \frac{\text{Số điều kiện được thực thi}}{\text{Tổng số điều kiện}}$$

Đảm bảo chất lượng phần mềm

100

## Ví dụ: Bao phủ điều kiện

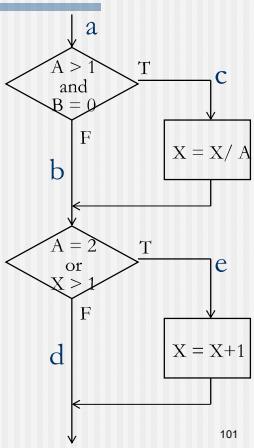
⦿ Cần thiết kế các test cases bao phủ hết các điều kiện sau:

A>1, A<=1, B=0, B !=0  
A=2, A !=2, X >1, X<=1

⦿ Test cases:

- 1) A = 1, B = 0, X = 3 (abe)
- 2) A = 2, B = 1, X = 1 (abe)

⦿ Không thỏa mãn bao phủ nhánh



101

## Bài tập 1

```

1. float A, B, C;
2. printf("Enter three values\n");
3. scanf("%f%f%f", &A, &B, &C);
4. printf("\n largest value is: ");
5. if( A>B)
6. {
7.   if (A>C) printf("%f\n",A);
8.   else printf("%f\n",C);
9. }
10. else
11. {
12.   if (C>B) printf("%f\n",C);
13.   else printf("%f\n",B);
14. }

```

Đảm bảo chất lượng phần mềm

102

## Bài tập 2

```

1. void PTA(int a[], int n)
2. {
3.   int i = 0;
4.   while ((i < n) && (a[i] >= 0))
5.     i++;
6.   if (i < n)
7.     printf("Phan tu am a[%d] = %d", i, a[i]);
8.   else
9.     printf("Khong co phan tu am.");
10. }

```

Đảm bảo chất lượng phần mềm

103

## Bài tập 3

```

1. const int SCALENE = 1;
2. const int ISOSCELES = 2;
3. const int EQUILATERAL = 3;
4. const int ERROR = 4;
5. int TriangleType(int x, int y, int z){
6.   if (x<=0 || y<=0 || z<=0)
7.     return ERROR;
8.   if ((x + y <= z) || (x + z <= y) || (z + y <= x))
9.     return ERROR;
10.  if (x == y && y == z)
11.    return EQUILATERAL;
12.  if (x == y || y == z || z == x)
13.    return ISOSCELES;
14.  return SCALENE;
15. }

```

Đảm bảo chất lượng phần mềm

104

## Bao phủ nhánh – điều kiện

### ❑ Bao phủ nhánh - điều kiện

- Viết các TCs sao cho mỗi điều kiện (đơn) trong mỗi nhánh nhận được 2 giá trị(T và F) ít nhất 1 lần và mỗi nhánh được thực hiện ít nhất 1 lần

### ❑ Bao phủ tổ hợp các điều kiện

- Viết các TCs để thực thi được tất cả các tổ hợp giá trị(T và F) của các điều kiện (đơn) trong 1 nhánh

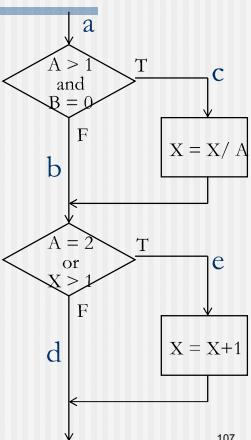
Đảm bảo chất lượng phần mềm

105

## Ví dụ: Bao phủ tổ hợp điều kiện

### ❑ TCs phải bao phủ các điều kiện

- 1)  $A > 1, B = 0$
- 2)  $A > 1, B \neq 0$
- 3)  $A \leq 1, B = 0$
- 4)  $A \leq 1, B \neq 0$
- 5)  $A = 2, X > 1$
- 6)  $A = 2, X \leq 1$
- 7)  $A \neq 2, X > 1$
- 8)  $A \neq 2, X \leq 1$



Đảm bảo chất lượng phần mềm

107

## Đoán lỗi(Error guessing)

- ❑ Được sử dụng sau khi áp dụng các kĩ thuật hình thức khác
- ❑ Có thể tìm ra 1 số lỗi mà các kĩ thuật khác có thể bỏ lỡ
- ❑ Không có qui tắc chung

Đảm bảo chất lượng phần mềm

109

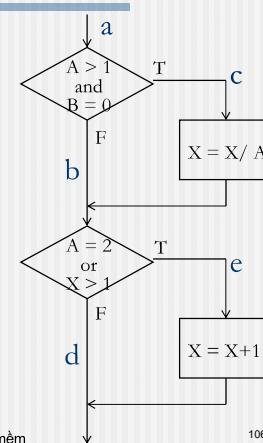
## Ví dụ: Bao phủ nhánh-điều kiện

### ❑ Các TCs cần bao phủ tất cả các điều kiện

$A > 1, A \leq 1, B = 0, B \neq 0$   
 $A = 2, A \neq 2, X > 1, X \leq 1$   
và  $(A > 1 \text{ and } B = 0) \quad T, F$   
 $(A = 2 \text{ or } X > 1) \quad T, F$

### ❑ Test cases:

- 1)  $A = 2, B = 0, X = 4$  (ace)
- 2)  $A = 1, B = 1, X = 1$  (abd)



Đảm bảo chất lượng phần mềm

106

## Kĩ thuật kiểm thử dựa trên kinh nghiệm

- ❑ Error guessing
- ❑ Exploratory testing

Đảm bảo chất lượng phần mềm

108

## Kiểm thử thăm dò (Exploratory testing)

- ❑ Hoạt động thiết kế test và thực thi test được thực hiện song song
- ❑ Các chú ý sẽ được ghi chép lại để báo cáo sau này
- ❑ Khía cạnh chủ chốt: learning(cách sử dụng, điểm mạnh, điểm yếu của PM)

Đảm bảo chất lượng phần mềm

110

## Chọn kĩ thuật kiểm thử

- ❑ Mỗi kĩ thuật riêng lẻ chỉ hiệu quả với 1 nhóm lỗi cụ thể
- ❑ Các yếu tố ảnh hưởng đến việc chọn kĩ thuật kiểm thử:
  - Loại hệ thống
  - Tiêu chuẩn quy định
  - Yêu cầu khách hàng hoặc hợp đồng
  - Mức độ rủi ro, loại rủi ro
  - Mục tiêu kiểm thử
  - Tài liệu có sẵn

Đảm bảo chất lượng phần mềm

111

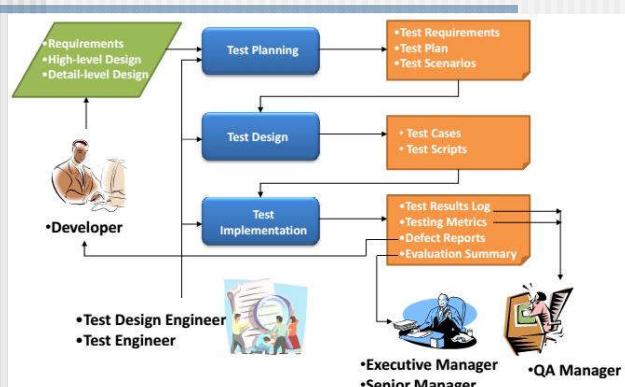
## Chọn kĩ thuật kiểm thử

- Kiến thức của testers
- Thời gian và ngân sách
- Mô hình phát triển PM
- Kinh nghiệm về các loại lỗi đã được tìm ra trước đó

Đảm bảo chất lượng phần mềm

112

## D.Quy trình kiểm thử



113

## Quy trình kiểm thử

### ❑ Bao gồm các hoạt động:

1. Lập kế hoạch
2. Phân tích xác định các điều kiện, thiết kế TCs
3. Thực thi tests
4. Đánh giá các tiêu chuẩn và báo cáo
5. Kiểm tra hoàn thành tests

Đảm bảo chất lượng phần mềm

114

## Quy trình kiểm thử

### Bước 1: Vạch kế hoạch

- Thiết lập chiến lược test
- Xác định những người tham gia: testers, QA, development, support,...
- Xác định các yêu cầu, đặc tả về chức năng
- Chuẩn bị cơ sở hạ tầng cho test

Đảm bảo chất lượng phần mềm

115

## Quy trình kiểm thử

### Bước 2: Phân tích, thiết kế test

- Xác định điều kiện test, độ ưu tiên
- Thiết kế các Test cases
- Chuẩn bị dữ liệu kịch bản

Đảm bảo chất lượng phần mềm

116

## Quy trình kiểm thử

### Bước 3: Thực thi tests

- Thực thi các TCs đã được thiết kế:
  - Thực hiện trước các TCs quan trọng nhất
  - Có thể thực thi bằng tay hoặc tự động
  - Không thực thi tất cả các TCs nếu:
    - Có quá nhiều lỗi ở các TCs trước
    - Áp lực thời gian

Đảm bảo chất lượng phần mềm

117

## Quy trình kiểm thử

### Bước 4: Đánh giá và báo cáo

- Ghi lại các phiên bản của PM trong khi test
- Với mỗi TCs, ghi lại kết quả thực tế, độ bao phủ test
- So sánh kết quả thực tế với kết quả mong đợi
- Sau khi lỗi được fix thì cần phải test lại
- Viết báo cáo trạng thái kiểm thử

Đảm bảo chất lượng phần mềm

118

## Quy trình kiểm thử

### Bước 5: Kiểm tra hoàn thành tests

- Tiêu chuẩn kết thúc test đã được đưa vào test plan
- Nếu các tiêu chuẩn chưa được đáp ứng thì thực hiện lại từ việc thiết kế thêm các TCs
- Các tiêu chuẩn có thể ứng dụng với tất cả các mức test

Đảm bảo chất lượng phần mềm

119

## D.Tổ chức kiểm thử

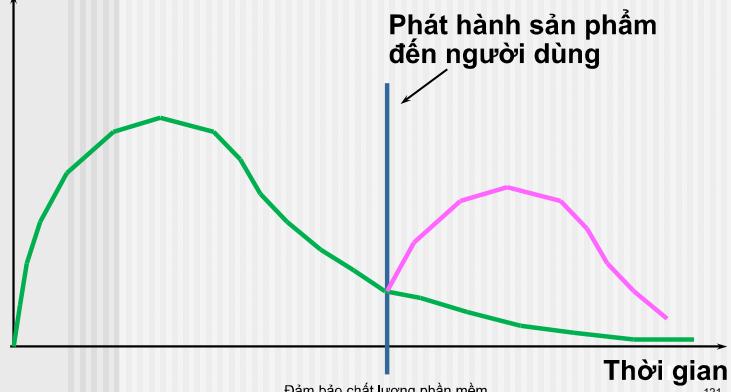
- ❑ Tầm quan trọng của tính độc lập trong kiểm thử
- ❑ Cấu trúc tổ chức kiểm thử

Đảm bảo chất lượng phần mềm

120

## Tầm quan trọng của tính độc lập trong kiểm thử

### Số lượng lỗi



## Cấu trúc tổ chức kiểm thử

1. Trách nhiệm của mỗi developer
2. Trách nhiệm của cả nhóm phát triển(Development team )
3. Tester nằm trong nhóm phát triển
4. Nhóm testers riêng biệt
5. Có vấn cho kiểm thử
6. Nhóm testers thuộc công ty thứ 3

Đảm bảo chất lượng phần mềm

122

## Mỗi developer thực hiện kiểm thử

### ❖ Ưu điểm

- Hiểu rõ code
- Tìm được các lỗi kỹ thuật mà tester có thể bỏ lỡ
- Tìm lỗi và sửa lỗi ít tốn ngân sách

### ❖ Nhược điểm

- Đánh giá chủ quan
- Khó phá hủy công việc của chính họ
- Xu hướng thấy được kết quả mong đợi hơn là kết quả thực tế

Đảm bảo chất lượng phần mềm

123

## Nhóm phát triển thực hiện kiểm thử

### ❑ Ưu điểm

- Có sự độc lập nhất định
- Tìm được các lỗi kỹ thuật mà tester có thể bỏ lỡ
- Thân thiện, dễ dàng giao tiếp

### ❑ Nhược điểm

- Thiếu kỹ năng test
- Bị áp lực công việc

Đảm bảo chất lượng phần mềm

124

## Tester nằm trong nhóm phát triển

### ❑ Ưu điểm

- Tester chỉ có trách nhiệm kiểm thử
- Độc lập trong kiểm thử
- Tester là thành viên của nhóm → có chung 1 mục đích

### ❑ Nhược điểm

- Ít được tôn trọng
- Làm việc 1 mình
- Quan điểm đưa ra đơn lẻ

Đảm bảo chất lượng phần mềm

125

## Đội ngũ tester độc lập

### ❑ Ưu điểm

- Gồm những tester có kinh nghiệm, chỉ làm nhiệm vụ kiểm thử
- Kiểm thử khách quan hơn và nhất quán hơn

### ❑ Nhược điểm

- Phụ thuộc vào đội ngũ tester → test thiếu
- Cố lập với nhóm phát triển

Đảm bảo chất lượng phần mềm

126

## Chuyên gia tư vấn kiểm thử nội bộ

### ❑ Ưu điểm

- Gồm các chuyên gia trong lĩnh vực kiểm thử → cung cấp, hỗ trợ giúp cải tiến quá trình kiểm thử
- Vạch kế hoạch, ước lượng, điều khiển quá trình kiểm thử tốt hơn

### ❑ Nhược điểm

- Mức chuyên gia đã đủ chưa??
- Lời khuyên, hướng dẫn chỉ ảnh hưởng đến thủ tục test, chứ không quyết định hoàn toàn
- Một số chuyên gia cũng phải thực thi test

Đảm bảo chất lượng phần mềm

127

## Nhóm testers thuộc công ty thứ 3

### ❑ Ưu điểm

- Gồm các chuyên gia trong lĩnh vực kiểm thử
- Độc lập với các chính sách nội bộ

### ❑ Nhược điểm

- Chi phí đắt
- Thiếu kiến thức về sản phẩm

Đảm bảo chất lượng phần mềm

128

## Những kỹ năng cần thiết trong đội ngũ kiểm thử

- ❑ Chuyên gia kỹ thuật
- ❑ Chuyên gia về kiểm thử tự động
- ❑ Chuyên gia database
- ❑ Kỹ năng business
- ❑ Hiểu biết tốt về tính khả dụng
- ❑ Chuyên gia về môi trường test
- ❑ Người quản lý test

Đảm bảo chất lượng phần mềm

130

## Vai trò của Test leader

- ❑ Lập kế hoạch, giám sát và kiểm soát các hoạt động kiểm thử
- ❑ Lập các mục tiêu, tổ chức chính sách, chiến lược kiểm thử
- ❑ Uớc tính test cần thực hiện và đàm phán với quản lý để có được các nguồn lực cần thiết
- ❑ Nhận biết thời điểm cần test tự động, chọn lựa tools, tổ chức training
- ❑ Tham khảo ý kiến với nhóm khác

Đảm bảo chất lượng phần mềm

131

## Vai trò của Test leader

- Hướng dẫn, giám sát việc phân tích, thiết kế, thực thi các TCs, thủ tục kiểm thử, kịch bản kiểm thử
- Đảm bảo quản lý cấu hình testware
- Đảm bảo thực hiện môi trường test
- Lập lịch thực hiện kiểm thử
- Theo dõi, đo lường, kiểm soát và báo cáo về tiến độ kiểm thử, tình trạng chất lượng sản phẩm và kết quả kiểm thử
- Viết báo cáo tóm tắt về trạng thái test

Đảm bảo chất lượng phần mềm

132

## Vai trò của tester

- ❑ Rà soát và góp ý cho kế hoạch kiểm thử
- ❑ Phân tích, đánh giá các yêu cầu của người dùng, các đặc tả thiết kế.
- ❑ Xác định điều kiện test, thiết kế TCs, thủ tục test, hỗ trợ test tự động
- ❑ Cài đặt môi trường test

Đảm bảo chất lượng phần mềm

133

## Vai trò của tester

- ❑ Thực hiện kiểm thử trên tất cả các mức test
- ❑ Thực thi, ghi nhận, đánh giá kết quả test, lập tài liệu báo cáo lỗi
- ❑ Rà soát đặc tả test, các báo cáo lỗi, kết quả test.
- ❑ Theo dõi quá trình kiểm thử, môi trường test sử dụng các tools

Đảm bảo chất lượng phần mềm

134