INTRODUCTION TO SOFTWARE ENGINEERING

PART II – SOFTWARE PROCESSES

Software Engineering Dept.
College of Information and Communication Technology, CTU

CONTENT

- Requirements analysis & specification (Definition)
- Software design
- Implementation (Programming)
- Testing
- Delivery and Maintenance

SOFTWARE PROCESSES

Part II.4 – Testing

3

Nội dung

Session 1 – Testing the Programs

- Session 2 – Testing the System

4

Session 1 - Testing the Programs

- Software Faults and Failures
- Testing Issues
- Unit Testing
- Integration Testing
- Test Planning
- Automated Testing Tools

5

Software Faults and Failures

Why Does Software Fail?

- Wrong requirement: not what the customer wants
- Missing requirement
- Requirement impossible to implement
- Faulty design
- Faulty code
- Improperly implemented design

Software Faults and Failures Objective of Testing

- Objective of testing: discover faults
- A test is successful only when a fault is discovered
  - Fault identification is the process of determining what fault caused the failure
  - Fault correction is the process of making changes to the system so that the faults are removed

Software Faults and Failures Types of Faults

- Algorithmic fault
- Computation and precision fault
  - a formula's implementation is wrong
- Documentation fault
  - Documentation doesn't match what program does
- Capacity or boundary faults
  - System's performance not acceptable when certain limits are reached
- Timing or coordination faults
- Performance faults
  - System does not perform at the speed prescribed
- Standard and procedure faults

Software Faults and Failures
Typical Algorithmic Faults

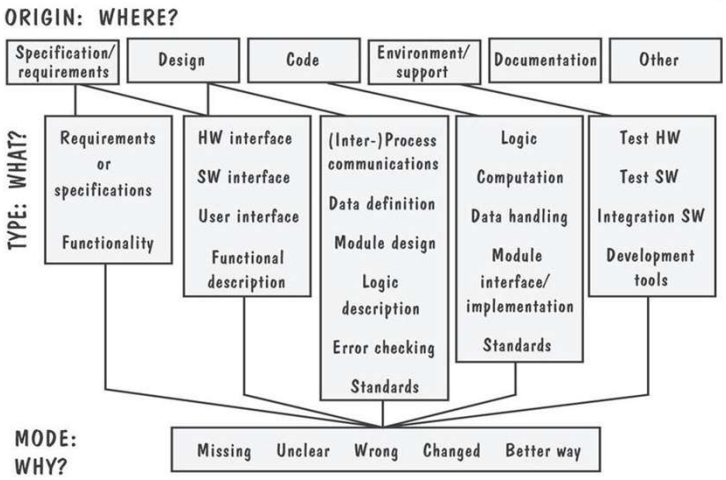
- An algorithmic fault occurs when a component's algorithm or logic does not produce proper output
  - Branching too soon
  - Branching too late
  - Testing for the wrong condition
  - Forgetting to initialize variable or set loop invariants
  - Forgetting to test for a particular condition
  - Comparing variables of inappropriate data types
- Syntax faults

Software Faults and Failures
Orthogonal Defect Classification

Fault Type	Meaning
Function	Fault that affects capability, end-user interface, product interface with hardware architecture, or global data structure
Interface	Fault in interacting with other component or drivers via calls, macros, control blocks, or parameter lists
Checking	Fault in program logic that fails to validate data and values properly before they are used
Assignment	Fault in data structure or code block initialization
Timing/serialization	Fault in timing of shared and real-time resources
Build/package/merge	Fault that occurs because of problems in repositories, management changes, or version control
Documentation	Fault that affects publications and maintenance notes
Algorithm	Fault involving efficiency or correctness of algorithm or data structure but not design

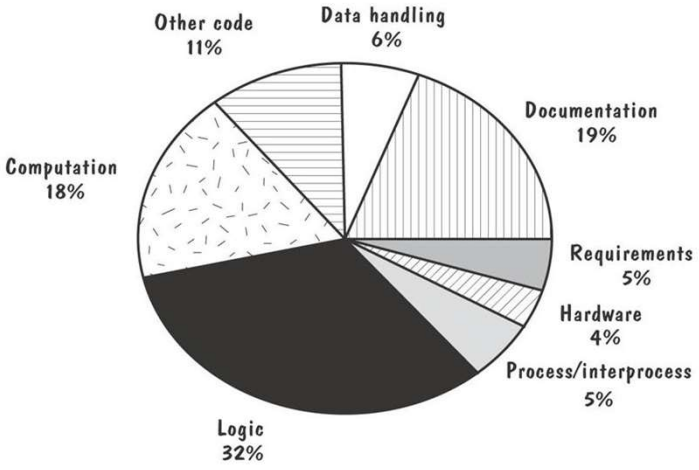
8.1 Software Faults and Failures

Sidebar 8.1 Hewlett-Packard's Fault Classification



Software Faults and Failures

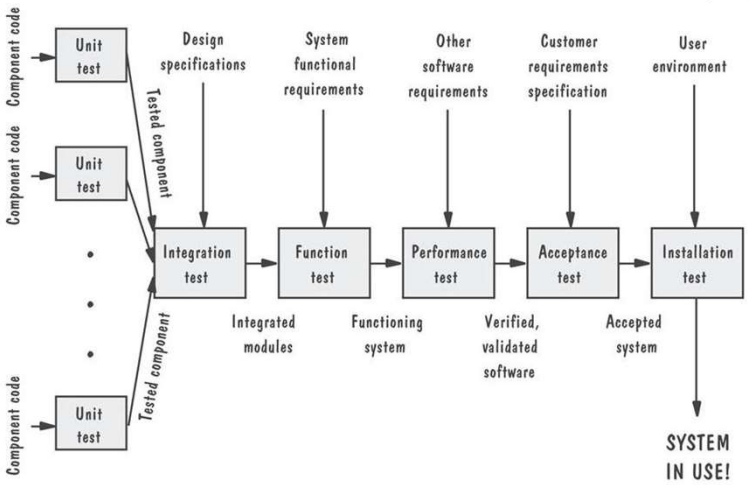
Faults for one Hewlett-Packard Division



Testing Issues
Testing Organization

- Module testing, component testing, or unit testing
- Integration testing
- Function testing
- Performance testing
- Acceptance testing
- Installation testing

Testing Issues
Testing Organization Illustrated



Testing Issues
Attitude Toward Testing

- Egoless programming: programs are viewed as components of a larger system, not as the property of those who wrote them

Testing Issues
Who Performs the Test?

- Independent test team
  - avoid conflict
  - improve objectivity
  - allow testing and coding concurrently

Testing Issues

Views of the Test Objects

- Closed box or black box: functionality of the test objects.
  - **Black-box testing** is a method of software **testing** that examines the functionality of an application without peering into its internal structures or workings
- Clear box or white box: structure of the test objects
  - **White-box testing** (also known as clear **box testing**, glass **box testing**, transparent **box testing**, and structural **testing**) is a method of **testing** software that **tests** internal structures or workings of an application

Các vấn đề trong kiểm thử

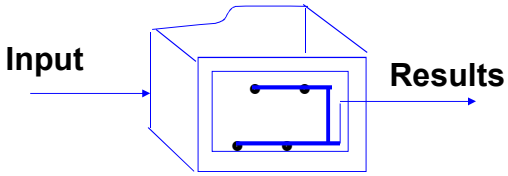
- Kiểm thử hộp đen
  - Đặc điểm
    - Đối tượng kiểm thử như một hộp đen, thông qua giao diện để đưa dữ liệu vào và nhận dữ liệu ra.

  - Độc lập với các ràng buộc bị tác động bởi cấu trúc bên trong và tính logic của đối tượng.

18

Các vấn đề trong kiểm thử

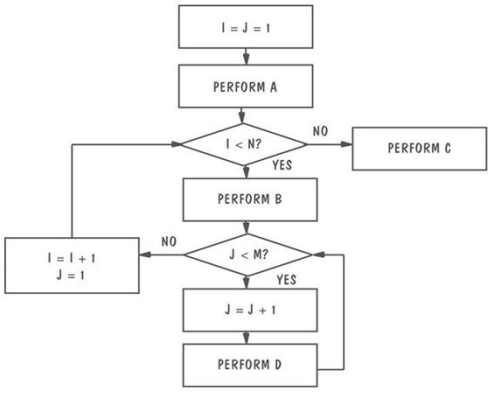
- Kiểm thử hộp trắng
  - Phương pháp thiết kế trường hợp kiểm thử sử dụng cấu trúc điều khiển của thiết kế thuật toán để dẫn ra các trường hợp kiểm thử.



19

Testing Issues
Clear Box

- Example of logic structure



Testing Issues
Sidebar 8.2 Box Structures

- Black box: external behavior description
- State box: black box with state information
- White box: state box with a procedure

Testing Issues
Factors Affecting the Choice of Test Philosophy

- The number of possible logical paths
- The nature of the input data
- The amount of computation involved
- The complexity of algorithms

Unit Testing

- **Unit testing** is a **software testing** method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.
- The steps of Unit test
  - Code Review.
  - Compile code and remove syntax error
  - Develop the test case involve input and it's expected output.

23

Unit Testing

Steps in Choosing Test Cases

- Determining test objectives
- Selecting test cases
- Defining a test

Unit Testing
Test Thoroughness

- Statement testing
- Branch testing
- Path testing
- Definition-use testing
- All-uses testing
- All-predicate-uses/some-computational-uses testing
- All-computational-uses/some-predicate-uses testing

Unit test
Path testing

Path testing : Test all possible directions of path

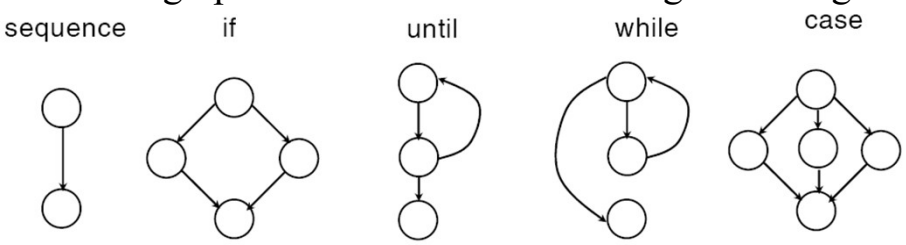
- Create ***Program path (flow path)***.
- List the basic independent execution paths.
- Generate test cases for those execution paths.

26

Path testing

Program graph is a **directed graph**

- Each circle node represents one or more actions.
- **edges** represent **flow of control**.
- Flow graph were constructed from algorithm logs.

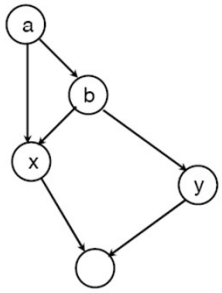


27

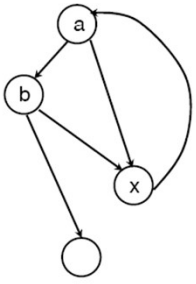
Path testing

Create ***flow path***

- All complex conditions must be decomposed into single conditions.



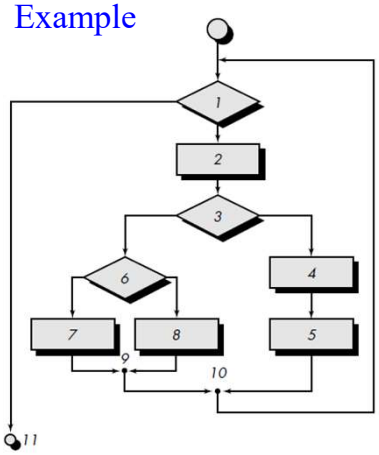
`if a and b then y else x`



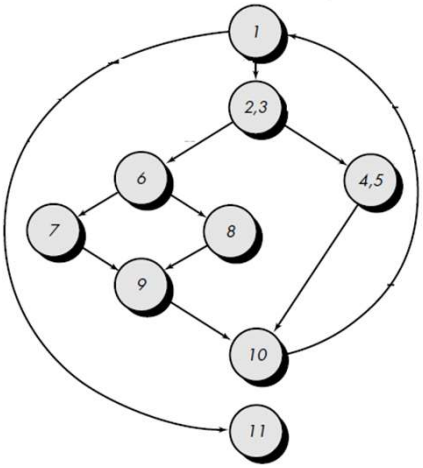
`while a or b do x`

Path testing

- Example



Flow chart



Flow graph

29

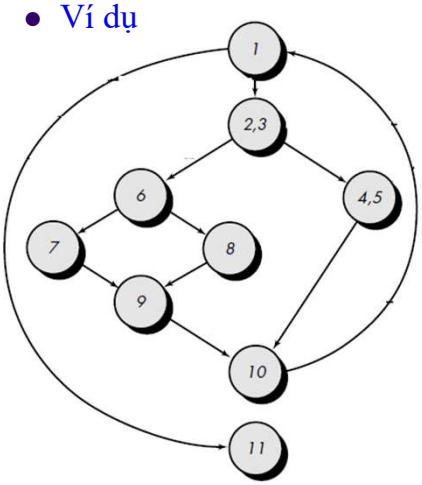
Path testing

- Listing the basic independence path
  - From the start node to the end node, the basic execution paths are listed in a certain order to ensure that the current listing path traverse at least a edge that is unlisted in a previous path
  - The total number of independent paths is calculated of **V = P + 1** where P is the number of branch nodes.

30

Path testing

- Ví dụ



Đồ thị dòng chảy

basic independence paths:

1. 1 - 11
2. 1 – 2,3 – 4,5 – 10 – 1 - 11
3. 1 – 2,3 – 6 – 7 – 9 – 10 – 1 - 11
4. 1 – 2,3 – 6 – 8 – 9 – 10 – 1 -11

31

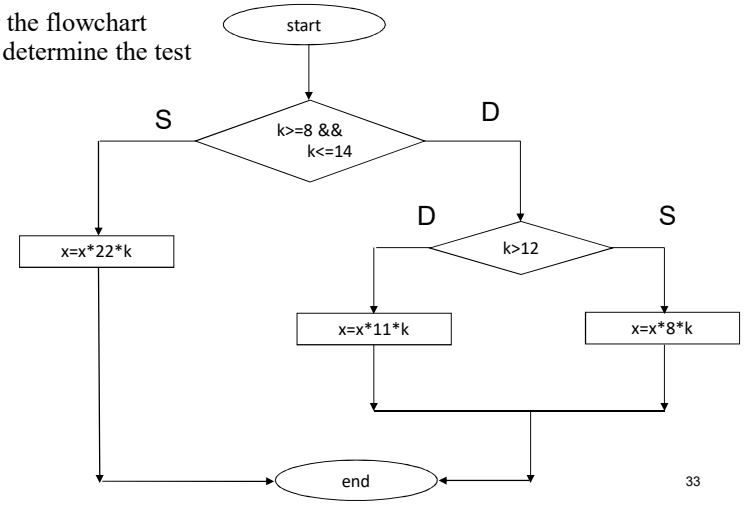
Path testing

- Develop test-case
  - Set up a test case for each basic independence path.
  - Based on the algorithm to
    - Give out input data.
    - Calculates the output data or the expected output of the algorithm.
  - Can not create test case for basic independence path.

32

Path testing

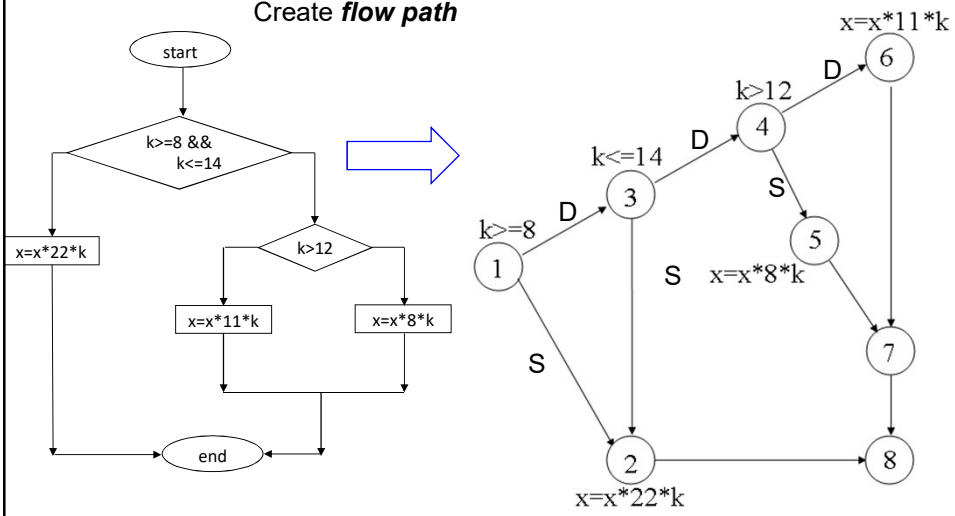
For example, for the flowchart shown in below, determine the test cases.



33

Path testing

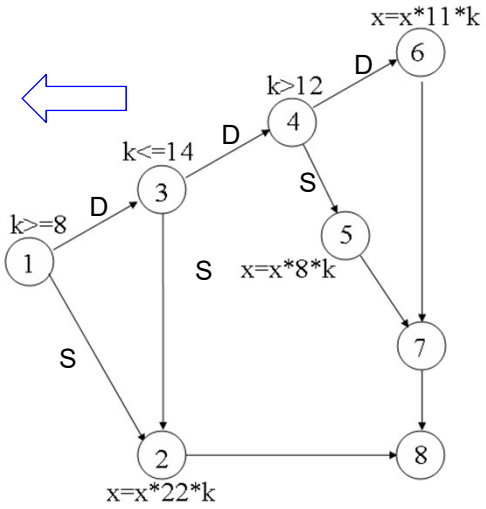
Create ***flow path***



Path testing

Develop the test cases

Path	Input	Output
1-2-8	k=5, x=10	1100
1-3-2-8	k=15, x=10	3300
1-3-4-5-7-8	k=10, x=10	800
1-3-4-6-7-8	k=13, x=10	1430



Integration Testing

Integrated testing to detect errors involving interactions between components.

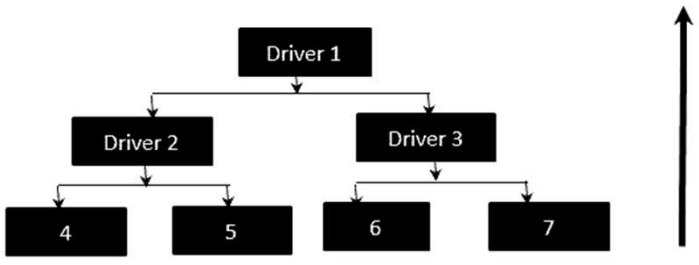
- Bottom-up
- Top-down
- Big-bang
- Sandwich testing
- Modified top-down
- Modified sandwich

Integration Testing
Terminology

- **Component Driver:** a routine that calls a particular component and passes a test case to it
- **Stub:** a special-purpose program to simulate the activity of the missing component

Integration Testing

- Test Driver
  - Test Drivers are the modules that act as temporary replacement for a calling module and give the same output as that of the actual product.
  - **Test Drivers** are used during Bottom-up integration testing in order to simulate the behaviour of the upper level modules that are not yet integrated.
  - Drivers are also used when the software needs to interact with an external system

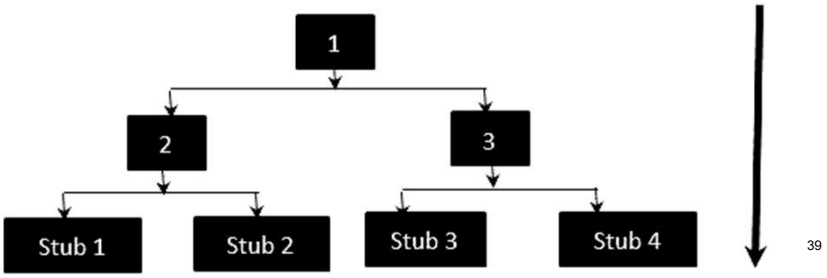


38

Integration Testing

Test Stub

- Stubs are used during Top-down integration testing, in order to simulate the behaviour of the lower-level modules that are not yet integrated.
- Stubs are the modules that act as temporary replacement for a called module and give the same output as that of the actual product.
- Stubs are also used when the software needs to interact with an external system.

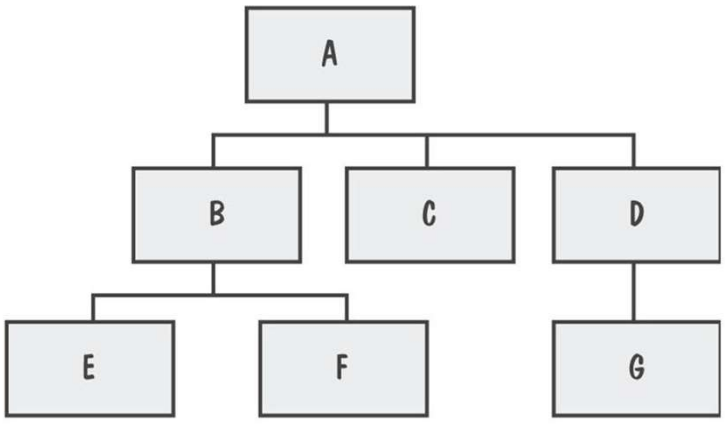


39

Integration Testing

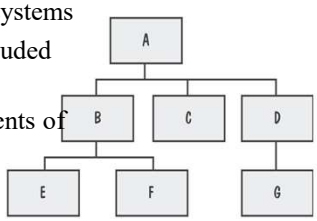
View of a System

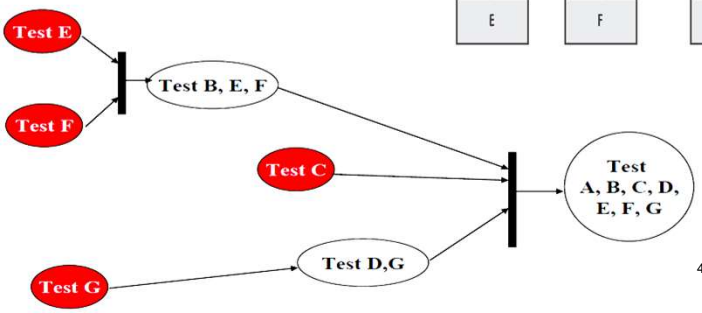
- System viewed as a hierarchy of components



Kiểm thử tích hợp

- Bottom-Up Integration Example
  - The subsystem in the lowest layer of the call hierarchy are tested individually
  - Then the next subsystems are integrated and tested from the next layer up that call the previously tested subsystems
  - This is done repeatedly until all subsystems are included in the testing
  - Only Test Drivers are used to simulate the components of higher layers



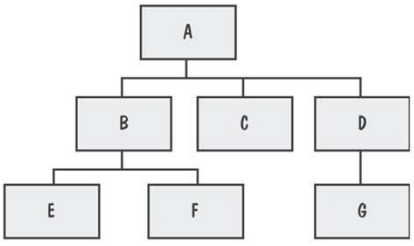


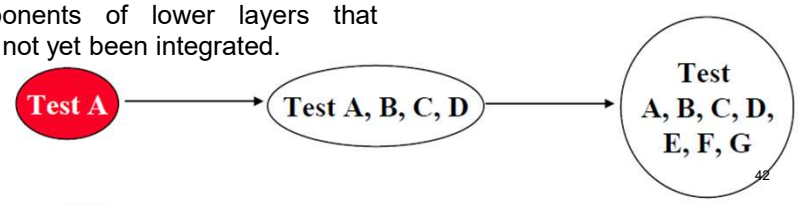
41

Kiểm thử tích hợp

- Top-Down Integration Example

- Test the top layer or the controlling subsystem first
- Then combine all the subsystems that are called by the tested subsystems and test the resulting collection of subsystems
- Do this until all subsystems are incorporated into the test
- Test Stubs are used to simulate the components of lower layers that have not yet been integrated.



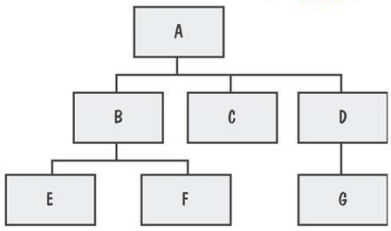


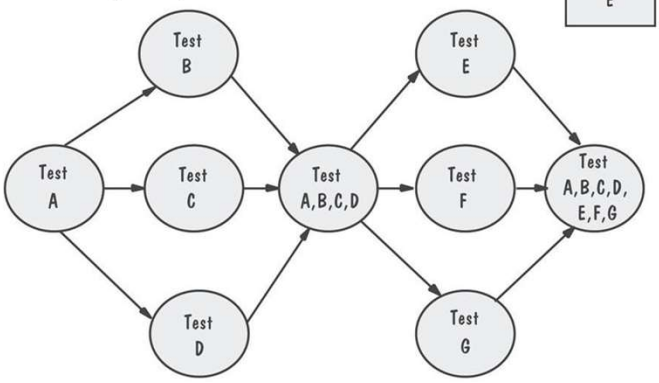
42

Kiểm thử tích hợp

Modified Top-Down Integration Example

- Each level's components individually tested before the merger

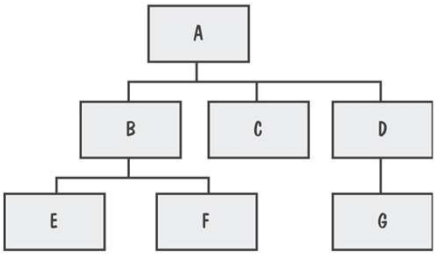


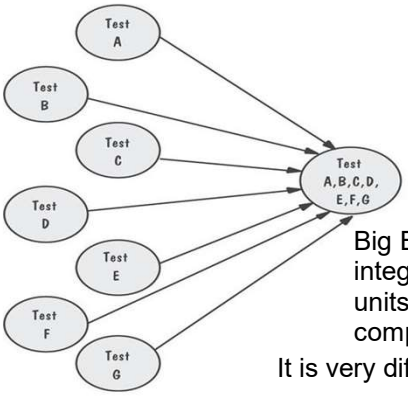


43

Kiểm thử tích hợp

- Big-Bang Integration Example





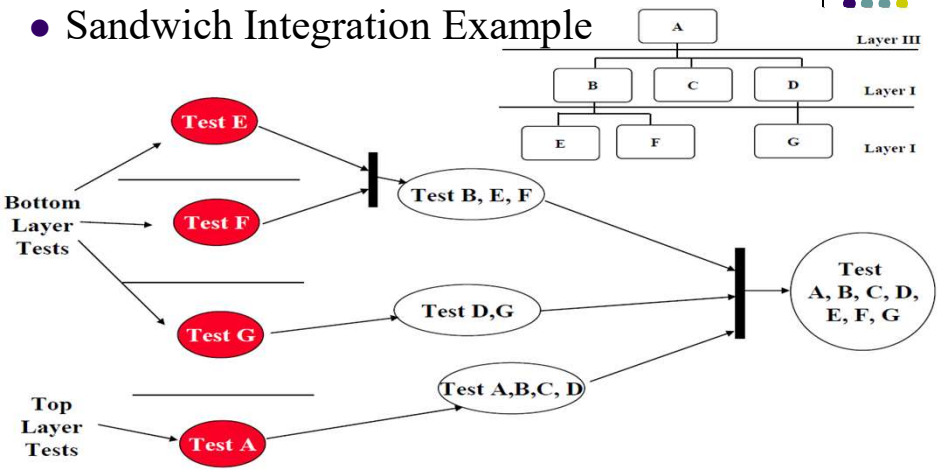
Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system

It is very difficult to isolate the defects found.

44

Kiểm thử tích hợp

- Sandwich Integration Example



Multi-level component hierarchy is divided into three levels with the test target being in the middle:

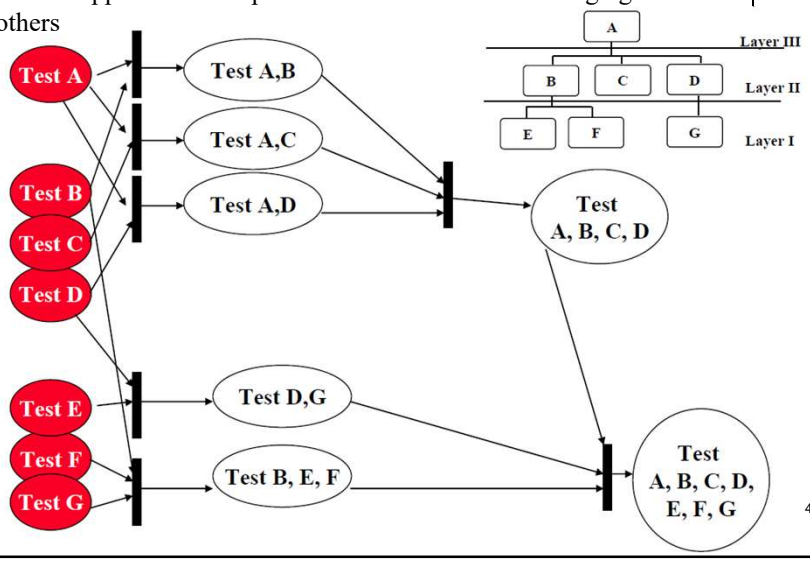
- Top-down approach is used in the top layer;
- Bottom-up approach used in the lower layer.

45

Kiểm thử tích hợp

Modified Sandwich Integration Example

- Allows upper-level components to be tested before merging them with others



46

Test Planning

- Establish test objectives
- Design test cases
- Write test cases
- Test test cases
- Execute tests
- Evaluate test results

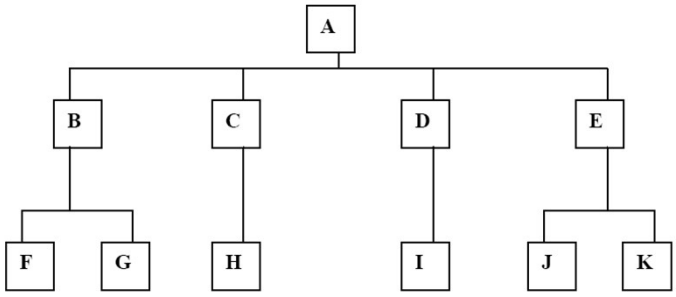
Test Planning
Purpose of the Plan

- Test plan explains
  - who does the testing
  - why the tests are performed
  - how tests are conducted
  - when the tests are scheduled

Test Planning
Contents of the Plan

- What the test objectives are
- How the test will be run
- What criteria will be used to determine when the testing is complete

Exercise



Xác định sơ đồ các dạng kiểm thử tích hợp?

50

Reading

51

Lập kế hoạch kiểm thử

- Thiết lập các mục tiêu kiểm thử
- Thiết kế các trường hợp kiểm thử
- Viết các trường hợp kiểm thử
- Kiểm tra các trường hợp kiểm thử
- Thực thi các kiểm thử
- Đánh giá các kết quả kiểm thử

52

Lập kế hoạch kiểm thử

- Mục đích của kế hoạch

  Kế hoạch kiểm thử giải thích:
  - Ai thực hiện kiểm thử
  - Tại sao các kiểm thử được thực hiện
  - Cách thức các kiểm thử được kiểm soát
  - Khi nào các kiểm thử được thực hiện

53

Lập kế hoạch kiểm thử

- Nội dung của kế hoạch:
  - Các mục tiêu của kiểm thử
  - Cách thực hiện kiểm thử
  - Chuẩn được sử dụng để xác định khi nào kiểm thử hoàn thành

54

Công cụ kiểm thử tự động

- Công cụ phân tích mã lệnh
  - Phân tích tĩnh
    - Bộ phân tích mã lệnh
    - Bộ kiểm tra cấu trúc
    - Bộ phân tích dữ liệu
    - Bộ kiểm tra trình tự
  - Phân tích động
    - Các bộ giám sát chương trình: xem và báo cáo lại các hoạt động của chương trình

55

Công cụ kiểm thử tự động

- Công cụ thực hiện kiểm thử
  - Capture và replay
  - Các stub và driver
  - Môi trường kiểm thử tự động
- Bộ sinh các trường hợp kiểm thử

56

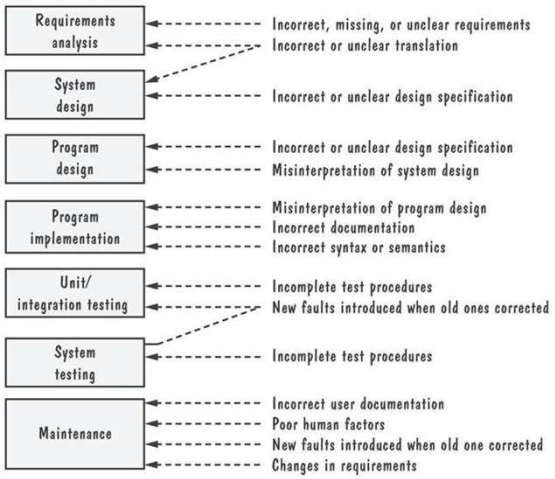
Phần 2 - Nội dung

- Các nguyên lý của kiểm thử hệ thống
- Kiểm thử chức năng
- Kiểm thử sự thực thi
- Tính tin cậy, tính sẵn có và tính có thể bảo trì
- Kiểm thử chấp nhận
- Kiểm thử sự cài đặt
- Tài liệu kiểm thử

57

Các nguyên lý của kiểm thử hệ thống

- Nguồn gốc của các lỗi phần mềm trong suốt quá trình phát triển



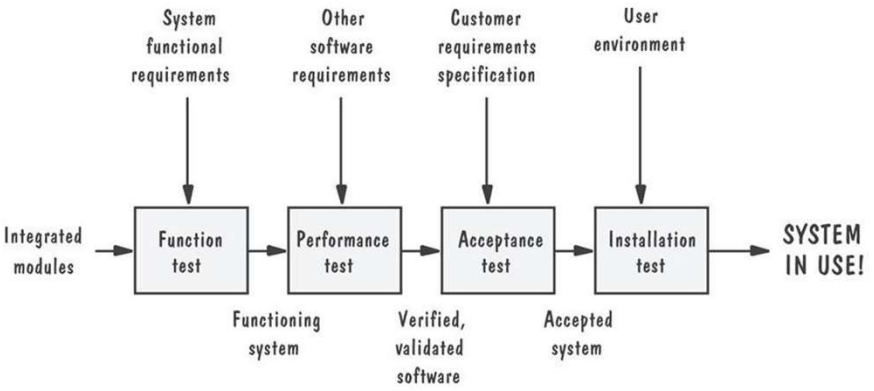
Các nguyên lý của kiểm thử hệ thống

- Quy trình kiểm thử hệ thống
  - Kiểm thử chức năng: hệ thống được tích hợp có thực hiện như được cam kết trong đặc tả yêu cầu?
  - Kiểm thử thực hiện: các yếu tố phi chức năng có được đáp ứng?
  - Kiểm thử chấp nhận: hệ thống có phải là cái mà khách hàng mong đợi?
  - Kiểm thử sự cài đặt: hệ thống có vận hành ở chỗ khách hàng không?

59

Các nguyên lý của kiểm thử hệ thống

- Các bước trong quy trình kiểm thử hệ thống



Các nguyên lý của kiểm thử hệ thống

- Những kỹ thuật được sử dụng trong kiểm thử hệ thống
  - Kế hoạch tích hợp hay xây dựng.
  - Kiểm thử hồi quy.
  - Quản lý cấu hình.

61

Các nguyên lý của kiểm thử hệ thống

- Kế hoạch tích hợp hay xây dựng
  - Mô tả các hệ thống con (spin) được kiểm thử.
  - Mô tả cách thức, nơi chốn, thời gian và người thực hiện các kiểm thử.

62

Các nguyên lý của kiểm thử hệ thống

- Ví dụ về kế hoạch xây dựng cho hệ thống viễn thông

Spin	Chức năng	Bắt đầu kiểm thử	Kết thúc kiểm thử
0	Exchange	1 September	15 September
1	Area code	30 September	15 October
2	State/province/district	25 October	5 November
3	Country	10 November	20 November
4	International	1 December	15 December

63

Các nguyên lý của kiểm thử hệ thống

- Kiểm thử hồi quy
  - Nhận dạng các lỗi mới mà chúng được gây ra bởi các lỗi hiện tại được hiệu chỉnh.
  - Kiểm tra phiên bản hay phát hành mới để xác nhận rằng nó vẫn thực hiện cùng các chức năng theo cùng cách như phiên bản hay phát hành cũ.

64

Các nguyên lý của kiểm thử hệ thống

- Các bước kiểm thử hồi quy
  - Chèn vào mã lệnh mới.
  - Kiểm thử các chức năng được biết sẽ bị ảnh hưởng bởi mã lệnh mới.
  - Kiểm thử các chức năng cần thiết của phiên bản m để xác nhận rằng chúng vẫn hoạt động chính xác.
  - Tiếp tục kiểm thử chức năng của phiên bản m + 1.

65

Các nguyên lý của kiểm thử hệ thống

- Nhóm kiểm thử
  - Kiểm thử viên chuyên nghiệp: tổ chức và thực hiện các kiểm thử.
  - Nhà phân tích: người đã tạo ra các đặc tả.
  - Nhà thiết kế hệ thống: hiểu giải pháp được đề nghị.
  - Chuyên gia quản lý cấu hình: giúp kiểm soát các sửa chữa.
  - Người dùng: đánh giá các phát sinh.

66

Kiểm thử chức năng

- Mục đích và vai trò
  - So sánh sự thực hiện thực tế của hệ thống với các yêu cầu của nó.
  - Phát triển các trường hợp kiểm thử dựa trên tài liệu yêu cầu.

67

Kiểm thử chức năng

- Tạo ra các trường hợp kiểm thử chức năng
  - Vẽ đồ thị nhân quả từ các yêu cầu
  - Chuyển đồ thị thành bảng quyết định
  - Mỗi cột trong bảng quyết định tương ứng với một trường hợp kiểm thử chức năng

68

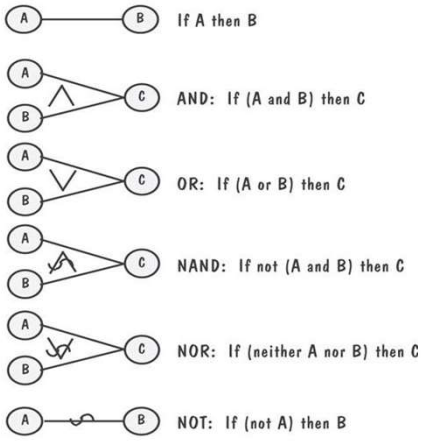
Kiểm thử chức năng

- Đồ thị nhân quả
  - Một đồ thị logic phản ánh quan hệ logic giữa nguyên nhân (các input) và kết quả (output hay sự biến đổi)
  - Vẽ đồ thị nhân quả từ các yêu cầu:
    - Các yêu cầu được phân tách sao cho mỗi yêu cầu mô tả một chức năng.
    - Mô tả các nguyên nhân, các kết quả (đó là các nút trong đồ thị với các nhân nằm bên trái và các quả nằm bên phải).
    - Vẽ mối quan hệ giữa chúng.

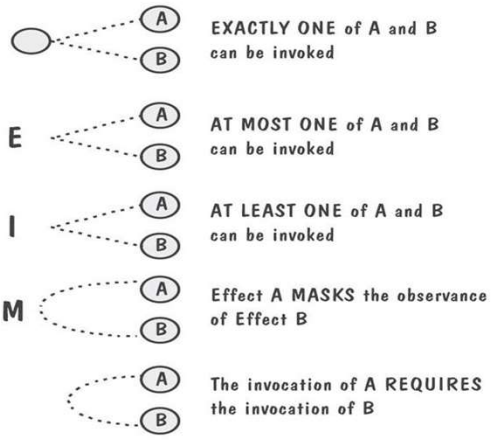
69

Kiểm thử chức năng

- Ký hiệu của đồ thị nhân quả



- Ký hiệu bổ sung của đồ thị nhân quả



Kiểm thử sự thực thi

- Mục đích và vai trò
  - Được sử dụng để kiểm tra:
    - Sự tính toán
    - Tốc độ đáp ứng
    - Độ chính xác của kết quả
    - Khả năng truy cập dữ liệu
  - Được thiết kế và quản lý bởi nhóm kiểm thử.

71

Kiểm thử sự thực thi

- Các loại kiểm thử sự thực thi
  - Kiểm thử hiệu suất
  - Kiểm thử dung lượng CTDL
  - Kiểm thử cấu hình
  - Kiểm thử tính tương thích
  - Kiểm thử hồi quy
  - Kiểm thử sự bảo mật
  - Kiểm thử sự điều hòa thời gian
  - Kiểm thử môi trường
  - Kiểm thử chất lượng
  - Kiểm thử sự hồi phục
  - Kiểm thử bảo trì
  - Kiểm thử tài liệu
  - Kiểm thử tính dễ sử dụng

72

Tính tin cậy, tính sẵn có và tính có thể bảo trì

- Định nghĩa
  - Tính tin cậy của phần mềm: vận hành mà không có lỗi dưới một điều kiện xác định trong một khoảng thời gian cho trước.
  - Tính sẵn có của phần mềm: vận hành thành công theo sự đặc tả tại một điểm thời gian xác định.
  - Tính có thể bảo trì của phần mềm: với một điều kiện sử dụng xác định, một hoạt động bảo trì có thể được thực hiện trong khoảng thời gian, thủ tục và tài nguyên xác định.

73

Tính tin cậy, tính sẵn có và tính có thể bảo trì

- Các mức độ khác nhau về lỗi
  - Thảm khốc: gây ra sự chết chóc hoặc sự thất bại của hệ thống.
  - Then chốt: gây ra tổn hại rất xấu hay sự hư hại hệ thống chính dẫn đến sự thất bại về nhiệm vụ.
  - Lề: gây ra tổn hại phụ hay sự hư hại hệ thống phụ dẫn đến sự ngừng trệ, mất tính sẵn có hay giảm sút trong công việc.
  - Thứ yếu: không đủ nghiêm trọng để gây ra tổn hại hay sự hư hại hệ thống nhưng sẽ dẫn đến việc bảo trì và sửa chữa không có kế hoạch.

74

Kiểm thử chấp nhận

- Mục đích và vai trò
  - Cho phép khách hàng và người dùng xác định xem hệ thống được xây dựng có đáp ứng được yêu cầu và sự mong đợi của họ hay không.
  - Được viết, quản lý và đánh giá bởi khách hàng.

75

Kiểm thử chấp nhận

- Các loại kiểm thử chấp nhận
  - Kiểm thử thử nghiệm (pilot): cài đặt hệ thống trên cơ sở thực nghiệm.
    - Kiểm thử alpha: kiểm thử của người dùng trong tổ chức hay công ty phát triển phần mềm.
    - Kiểm thử beta: kiểm thử của khách hàng.
  - Kiểm thử song song: một hệ thống mới vận hành song song với hệ thống cũ.

76

Kiểm thử sự cài đặt

- Trước khi kiểm thử
  - Cấu hình hệ thống
  - Gắn số và loại thiết bị
  - Thiết lập sự giao tiếp với hệ thống khác
- Kiểm thử
  - Kiểm thử hồi quy: kiểm tra rằng hệ thống được cài đặt một cách chính xác và hoạt động.

77

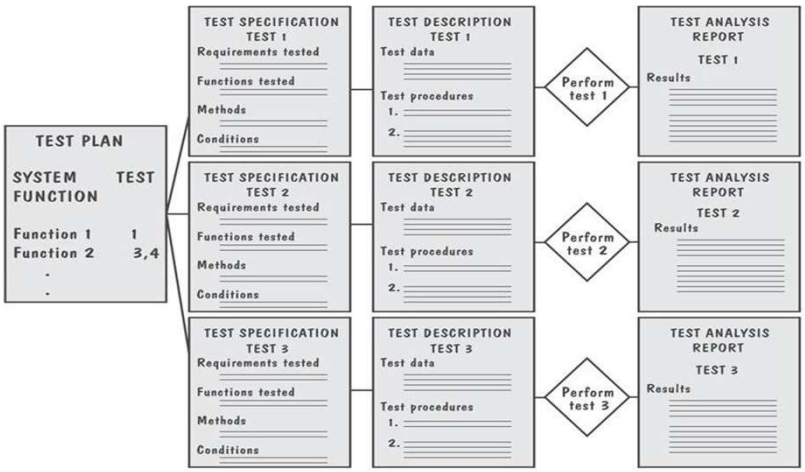
Tài liệu kiểm thử

- Kế hoạch kiểm thử: mô tả hệ thống và kế hoạch để kiểm tra tất cả các chức năng và các đặc trưng.
- Đặc tả và đánh giá kiểm thử: chi tiết từng kiểm thử và định nghĩa tiêu chí để đánh giá từng đặc điểm.
- Mô tả kiểm thử: thủ tục và dữ liệu kiểm thử cho từng kiểm thử.
- Báo cáo phân tích kiểm thử: các kết quả của từng kiểm thử.

78

Tài liệu kiểm thử

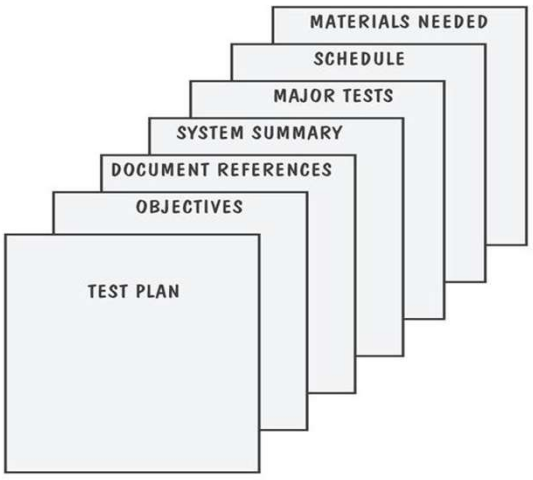
- Các tài liệu được tạo ra trong suốt sự kiểm thử



79

Tài liệu kiểm thử

- Các phần của một kế hoạch kiểm thử



80

Tài liệu kiểm thử

- Sự mô tả kiểm thử gồm:
  - Phương tiện kiểm soát
  - Dữ liệu
  - Thủ tục

81

Tài liệu kiểm thử

- Báo cáo phân tích kiểm thử
  - Ghi kết quả kiểm thử.
  - Cung cấp thông tin cần để sao chép lại sự thất bại, định vị và sửa tận gốc vấn đề.
  - Cung cấp thông tin cần thiết để xác định xem dự án có hoàn thành hay không.
  - Thiết lập sự tin cậy trong sự thực thi của hệ thống.

82

Tài liệu kiểm thử

- Biểu mẫu báo cáo vấn đề
  - Vị trí: vấn đề đã xuất hiện ở đâu?
  - Thời gian: nó đã xuất hiện khi nào?
  - Dấu hiệu: cái được quan sát?
  - Kết quả cuối cùng: các hệ quả?
  - Kỹ thuật: nó đã xuất hiện như thế nào?
  - Nguyên nhân: tại sao nó xuất hiện?
  - Tính khốc liệt: mức độ mà người dùng hay doanh nghiệp bị ảnh hưởng?
  - Chi phí: nó tốn bao nhiêu?

83

TIẾN TRÌNH PHẦN MỀM

PHẦN II.5 – TRIỂN KHAI & BẢO TRÌ

84

Nội dung

- Triển khai hệ thống
- Bảo trì

85

Triển khai hệ thống

- Triển khai hệ thống:
  - Không chỉ đặt hệ thống vào vị trí.
  - Còn giúp người dùng hiểu và cảm thấy thoải mái với hệ thống.
    - Huấn luyện (phân loại đối tượng, loại huấn luyện, phương tiện trợ giúp, nguyên tắc huấn luyện)
    - Tài liệu (phân loại độc giả, loại tài liệu)

86

Huấn luyện

- Phân loại người sử dụng hệ thống
  - Người dùng: sử dụng các chức năng của hệ thống chính.
  - Điều hành viên: thực hiện các chức năng bổ sung.
    - Tạo ra các bản sao dự phòng của các tập tin dữ liệu.
    - Xác định những người truy xuất vào hệ thống.

87

Huấn luyện

- Các chức năng của người dùng và điều hành viên

Chức năng của người dùng	Chức năng của điều hành viên
Thao tác trên các tập tin dữ liệu	Chấp nhận truy xuất của người dùng
Mô phỏng các hoạt động	Chấp nhận truy xuất tập tin
Phân tích dữ liệu	Thực hiện sao lưu
Giao tiếp dữ liệu	Cài đặt các thiết bị mới
Vẽ đồ thị và biểu đồ	Cài đặt phần mềm mới
	Phục hồi các tập tin hư

88

Huấn luyện

- Các kiểu huấn luyện
  - Huấn luyện người dùng
  - Huấn luyện điều hành viên
  - Yêu cầu huấn luyện đặc biệt

89

Huấn luyện

- Huấn luyện người dùng
  - Giới thiệu các chức năng cơ bản
    - Quản lý hồ sơ: tạo, xóa, truy lục, sắp xếp hồ sơ.
    - Sự điều hướng qua hệ thống.
    - Không cần kỹ thuật bên trong (giải thuật, cấu trúc dữ liệu, …).
  - Liên hệ cách các chức năng được thực hiện hiện tại, cách thực hiện sau đó bằng hệ thống mới.
    - Cần chú ý đến tính khó của học hỏi chuyển giao.

90

Huấn luyện

- Huấn luyện điều hành viên
  - Tập trung vào việc huấn luyện điều hành viên quen với các chức năng hỗ trợ của hệ thống và xác định cách thức hệ thống làm việc (hơn là cái mà hệ thống làm).
  - Thực hiện huấn luyện theo hai mức
    - Cách đưa ra và thực thi hệ thống mới.
    - Cách hỗ trợ người dùng.

91

Huấn luyện

- Yêu cầu huấn luyện đặc biệt
  - Người sử dụng thường xuyên và không thường xuyên.
  - Người sử dụng mới.

92

Huấn luyện

- Phương tiện trợ giúp huấn luyện
  - Tài liệu
    - Tài liệu hình thức, sách hướng dẫn.
    - Một tỷ lệ nhỏ người dùng đọc chúng.
  - Trợ giúp trực tuyến và biểu tượng
    - Biểu tượng cho đối tượng và chức năng.
    - Sách hướng dẫn trực tuyến cung cấp các liên kết siêu văn bản.
  - Các thể hiện và các lớp
    - Tổ chức lớp học; Sử dụng thiết bị đa phương tiện (nghe, nhìn).
  - Người dùng thành thạo (và người được huấn luyện)

93

Huấn luyện

- Nguyên tắc huấn luyện
  - Hiểu sở thích cá nhân, kiểu làm việc và áp lực tổ chức.
  - Cần xem xét các kiểu học viên khác nhau.
    - Hệ thống được đặc thù hóa
  - Chia nội dung huấn luyện thành các bài ở dạng ngắn.
  - Xác định kiểu huấn luyện dựa trên vị trí của học viên.

94

Tài liệu

- Hiểu thính giả
  - Phân nhóm thính giả
    - Người dùng
    - Điều hành viên
    - Nhân viên của khách hàng
    - Các thành viên khác của nhóm phát triển
  - Thiết kế tài liệu khác nhau cho các nhóm thính giả khác nhau.

95

Tài liệu

- Các loại tài liệu
  - Sách hướng dẫn người dùng
  - Sách hướng dẫn điều hành viên
  - Hướng dẫn hệ thống chung
  - Hướng dẫn học
  - Tài liệu khác: hướng dẫn lập trình viên

96

Tài liệu

- Sách hướng dẫn người dùng
  - Bắt đầu bằng mục tiêu chung, tiến tới mô tả chức năng chi tiết.
    - Mục tiêu và mục đích của hệ thống.
    - Các chức năng và khả năng của hệ thống.
    - Các đặc điểm, đặc trưng và thuận lợi của hệ thống.

97

Tài liệu

- Sách hướng dẫn điều hành viên
  - Cấu hình phần cứng, phần mềm.
  - Các phương pháp chấp nhận và từ chối truy xuất đối với người dùng.
  - Các thủ tục thêm để bổ sung và loại bỏ các ngoại vi ra khỏi hệ thống.
  - Các kỹ thuật để sao chép và dự phòng các tập tin và tài liệu.

98

Tài liệu

- Hướng dẫn hệ thống chung
  - Hệ thống chi tiết hóa theo cách mà khách hàng có thể hiểu.
  - Cấu hình phần cứng và phần mềm của hệ thống.

99

Tài liệu

- Hướng dẫn học
  - Hướng dẫn học được tự động hóa, từng bước một bằng đa phương tiện.

100

Tài liệu

- Hướng dẫn lập trình viên
  - Khái quát về cách thức phần cứng, phần mềm được cấu hình.
  - Các thành phần của phần mềm được chi tiết hóa và các chức năng của chúng được thực hiện.
  - Các chức năng hỗ trợ hệ thống.
  - Các cải tiến của hệ thống.

101

Tài liệu

- Gỡ rối và giúp đỡ người dùng
  - Các chỉ dẫn khi hệ thống không thực hiện.
  - Các tài liệu tham khảo.
  - Các tập tin giúp đỡ trực tuyến.

102

Bảo trì

- Định nghĩa
- Phân loại bảo trì
- Framework của bảo trì phần mềm (BTPM)
- Các vấn đề then chốt trong BTPM
- Quy trình BTPM
- Các kỹ thuật và công cụ trong BTPM

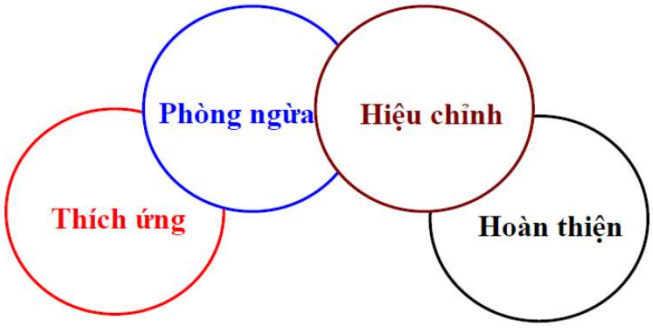
103

Định nghĩa

- Bảo trì phần mềm (Software Maintenace)
  - [IEEE 1219] Bảo trì phần mềm là sự sửa đổi một sản phẩm phần mềm sau khi phát hành nhằm hiệu chỉnh lỗi, cải thiện sự thực thi hay các đặc tính khác, hay làm cho sản phẩm thích ứng với môi trường bị thay đổi.
  - [ISO/IEC/IEEE 14764] Bảo trì phần mềm là sản phẩm phần mềm phải trải qua sự sửa đổi về mã lệnh và các tài liệu liên quan do có vấn đề hay nhu cầu cải tiến. Mục đích là sửa đổi sản phẩm phần mềm hiện có mà vẫn giữ được tính toàn vẹn của nó.

104

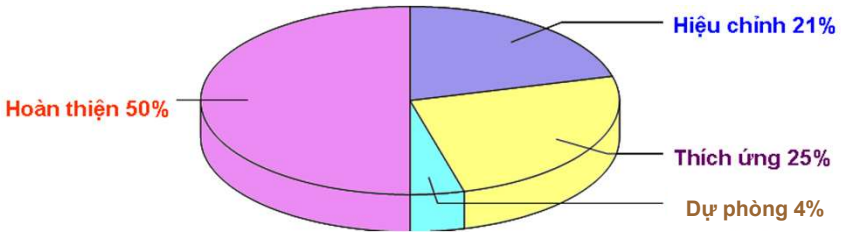
Các loại bảo trì phần mềm



105

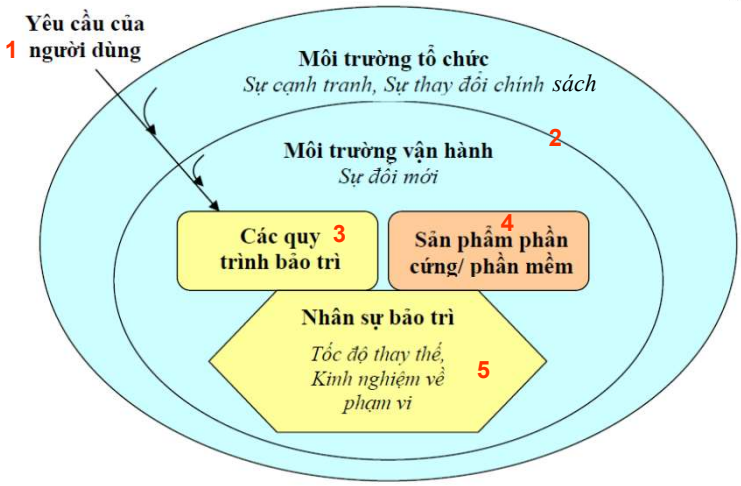
Các loại bảo trì phần mềm

- Công sức bảo trì



106

Framework của bảo trì phần mềm



Framework của bảo trì phần mềm

107

Các vấn đề then chốt trong BTPM

- Các vấn đề về kỹ thuật: Sự hiểu biết có giới hạn, Kiểm thử, Phân tích sự tác động, Tính có thể bảo trì.
- Các vấn đề trong quản lý: Bố trí nhân sự, Các vấn đề về quy trình, Chọn nhóm bảo trì, Gia công bảo trì phần mềm.
- Dự đoán chi phí phần mềm
- Phép đo bảo trì phần mềm

108

Quy trình bảo trì

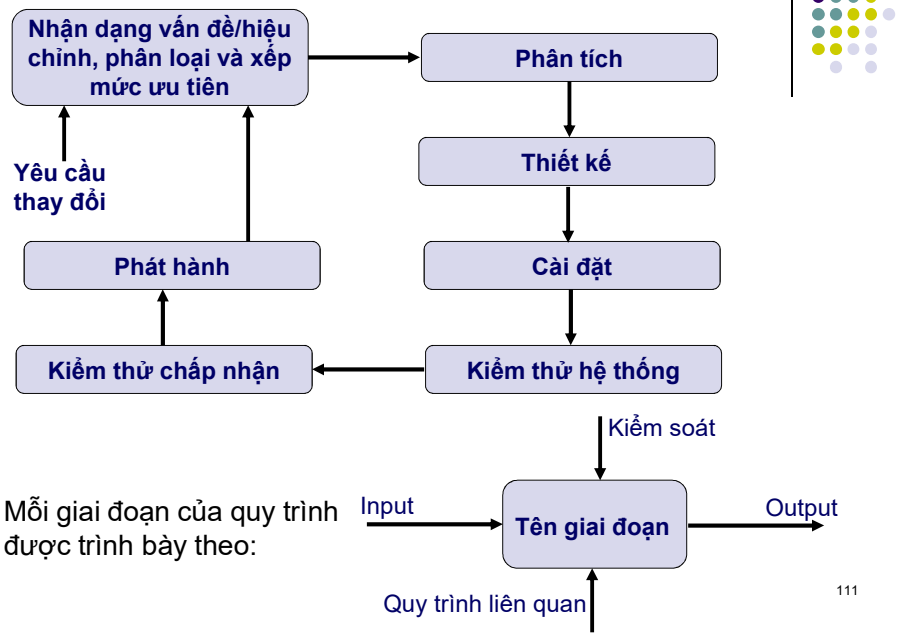
- *Quy trình bảo trì phần mềm* (Software Maintenance Process): Các chuỗi các hoạt động được thực hiện để đem lại sự thay đổi trong suốt sự bảo trì.
- Những quy trình bảo trì phổ biến được mô tả trong các chuẩn
  - IEEE 1219
  - ISO/IEC/IEEE 14764

109

Phần đọc thêm

110

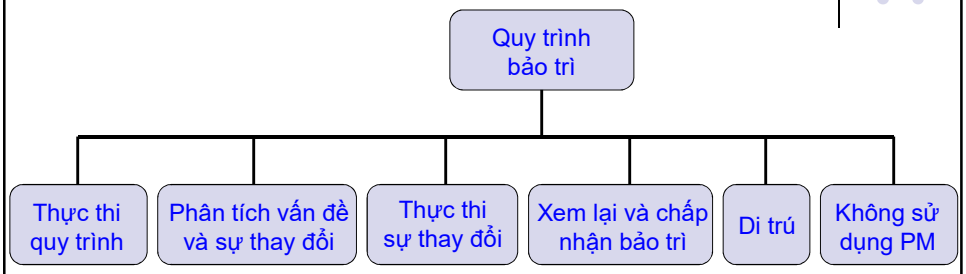
Quy trình BT theo chuẩn IEEE 1219

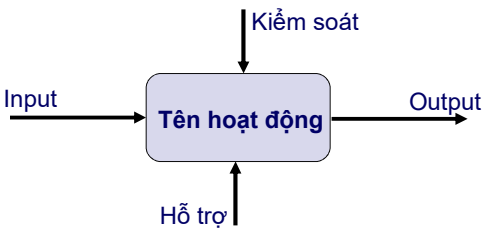


Mỗi giai đoạn của quy trình được trình bày theo:

111

Quy trình BT theo chuẩn ISO/IEC 14764





112

Kỹ thuật và công cụ bảo trì

- Sự hiểu biết về chương trình
- Kỹ thuật đảo ngược (Reverse Engineering)
- Kỹ thuật tái kiến tạo (ReEngineering)
- Các công cụ bảo trì

113

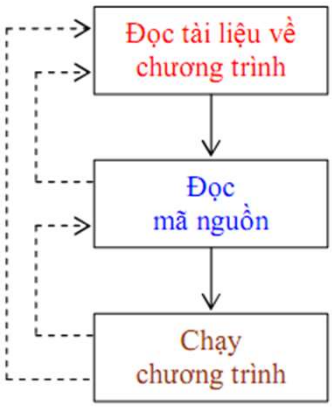
Sự hiểu biết về chương trình

- Mục đích cuối cùng của việc đọc và hiểu chương trình là có thể thực hiện thành công các thay đổi được yêu cầu.
- Điều này đòi hỏi phải có kiến thức về những điểm đặc trưng của một hệ thống phần mềm:
  - Phạm vi vấn đề
  - Ảnh hưởng của sự thực hiện
  - Quan hệ nhân - quả
  - Quan hệ sản phẩm - môi trường
  - Các điểm đặc trưng hỗ trợ việc ra quyết định

114

Sự hiểu biết về chương trình

- Các hoạt động liên quan đến việc hiểu chương trình



115

Định nghĩa

- Kỹ thuật đảo ngược là quy trình phân tích một hệ thống chủ thể để:
  - Nhận dạng các thành phần của hệ thống và các mối quan hệ giữa chúng
  - Tạo ra các biểu diễn của hệ thống ở một dạng khác hay ở các mức trừu tượng cao hơn

116

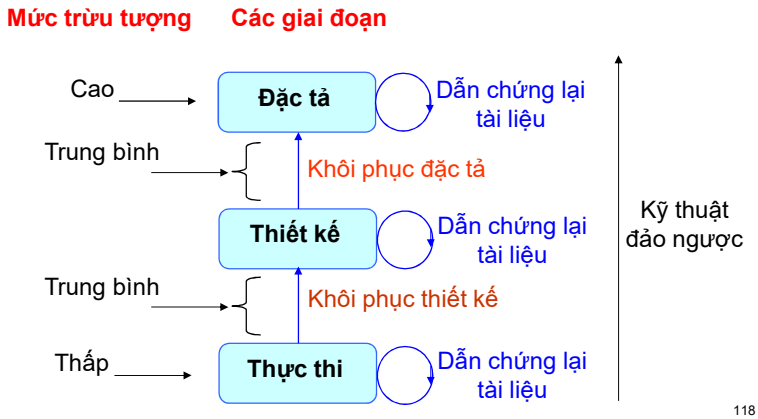
Kỹ thuật đảo ngược

- Kỹ thuật đảo ngược là quy trình phân tích một hệ thống chủ thể để:
  - Nhận dạng các thành phần của hệ thống và các mối quan hệ giữa chúng
  - Tạo ra các biểu diễn của hệ thống ở một dạng khác hay ở các mức trừu tượng cao hơn

117

Kỹ thuật đảo ngược

Các mức của kỹ thuật đảo ngược



118

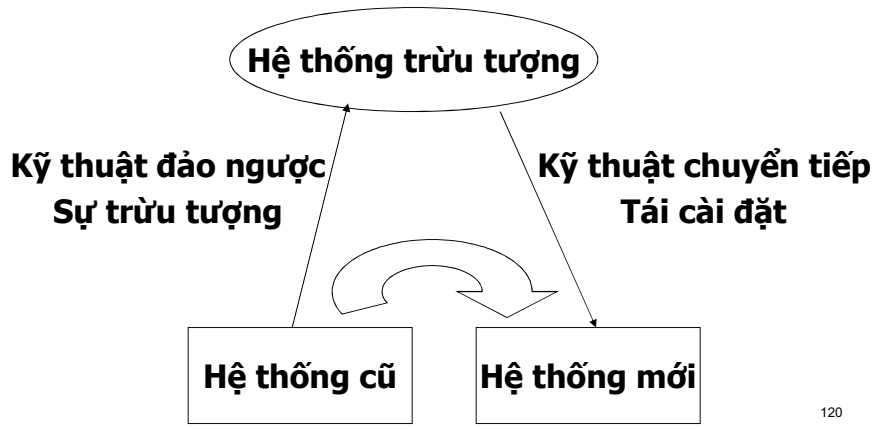
Kỹ thuật tái kiến tạo

- Kỹ thuật tái kiến tạo
  - Là quy trình kiểm tra và sửa đổi hệ thống đích để thực thi các thay đổi mong muốn.
  - Gồm hai bước:
    - Kỹ thuật đảo ngược
    - Kỹ thuật chuyển tiếp

119

Kỹ thuật tái kiến tạo

- Kỹ thuật tái kiến tạo



120

Các loại công cụ bảo trì

- Về nguyên tắc, ta có thể phân loại công cụ bảo trì phần mềm dựa trên công việc mà chúng hỗ trợ.
- Các công việc giúp phân loại công cụ bảo trì:
  - Hiểu chương trình, kỹ thuật đảo ngược
  - Kiểm thử
  - Quản lý cấu hình
  - Lập tài liệu và đo lường
- Trong thực tế, ta khó có được phân loại tốt vì trạng thái tự nhiên đa dạng hóa và có liên quan với nhau của các hoạt động bảo trì phần mềm

121