



## PHẦN V: KIỂM THỬ VÀ BẢO TRÌ



### I. Kiểm thử

1. Khái niệm kiểm thử
2. Phương pháp thử
3. Kỹ thuật thiết kế trường hợp thử
4. Kiểm thử module
5. Kiểm thử hệ thống
6. Kiểm thử chấp nhận

### II. Bảo trì

1



## 1. Khái niệm kiểm thử



### I. Kiểm thử

- Là mẫu chốt của đảm bảo chất lượng phần mềm
- Là tiến trình (và là nghệ thuật) nhằm phát hiện lỗi bằng việc xem xét lại đặc tả, thiết kế và mã nguồn.
- Kiểm thử thành công là phát hiện ra lỗi; kiểm thử không phát hiện ra lỗi là kiểm thử dở

2



## Khó khăn

- Nâng cao chất lượng phần mềm nhưng không vượt quá chất lượng khi thiết kế: chỉ phát hiện các lỗi tiềm tàng và sửa chúng
- Phát hiện lỗi bị hạn chế do thủ công là chính
- Dễ bị ảnh hưởng tâm lý khi kiểm thử
- Khó đảm bảo tính đầy đủ của kiểm thử

3

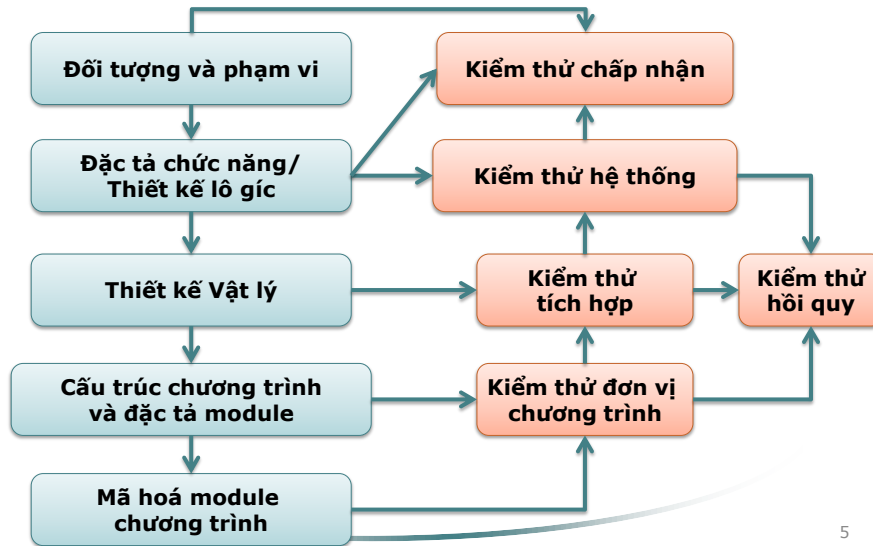


## Lưu ý khi kiểm thử

1. Chất lượng phần mềm do khâu thiết kế quyết định là chủ yếu, chứ không phải khâu kiểm thử
2. Tính dễ kiểm thử phụ thuộc vào cấu trúc chương trình
3. Người kiểm thử và người phát triển nên khác nhau
4. Dữ liệu thử cho kết quả bình thường thì không có ý nghĩa nhiều, cần có những dữ liệu kiểm thử mà phát hiện ra lỗi
5. Khi thiết kế trường hợp thử, không chỉ dữ liệu kiểm thử nhập vào, mà phải thiết kế trước cả dữ liệu kết quả sẽ có
6. Khi phát sinh thêm trường hợp thử thì nên thử lại những trường hợp thử trước đó để tránh ảnh hưởng lan truyền sóng

4

## Tương ứng giữa vòng đời dự án và kiểm thử



5

## 2.1. Kiểm thử tĩnh

I. Kiểm thử  
1. Khái niệm kiểm thử  
2. Phương pháp thử

- Kiểm thử trên bàn: giấy và bút trên bàn, kiểm tra logic, lần từng chi tiết ngay sau khi lập trình xong.
- Đi xuyên suốt (walk through)
- Thanh tra (inspection)

6



## 2.2. Kiểm thử trên máy

- Gỡ lỗi bằng máy (machine debug) hay kiểm thử động: Dùng máy chạy chương trình để điều tra trạng thái từng động tác của chương trình
- 9 bước của trình tự kiểm thử bằng máy:

7



## Trình tự kiểm thử bằng máy

1. Thiết kế trường hợp thử theo thử trên bàn
2. Trường hợp thử phải có cả kết quả kỳ vọng sẽ thu được
3. Dịch chương trình nguồn và tạo module tải để thực hiện
4. Khi trường hợp thử có xử lý tệp vào-ra, phải làm trước trên bàn việc xác định miền của các tệp
5. Nhập dữ liệu đã thiết kế cho trường hợp kiểm thử
6. Điều chỉnh môi trường thực hiện module tải (tạo thủ tục đưa các tệp truy cập tệp vào chương trình)
7. Thực hiện module tải và ghi nhận kết quả
8. Xác nhận kết quả với kết quả kỳ vọng
9. Lặp lại thao tác (5)-(8)

8



### 3. Kỹ thuật thiết kế trường hợp thử

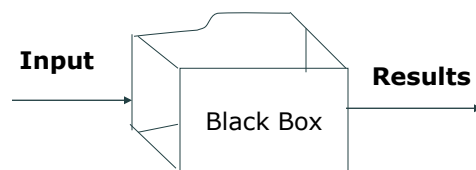
- Kỹ thuật thiết kế trường hợp thử dựa trên đặc tả bề ngoài của chương trình: Kiểm thử hộp đen (Black box test): WHAT ?
- Kỹ thuật thiết kế trường hợp thử dựa trên đặc tả bên trong của chương trình: Kiểm thử hộp trắng (white box test): HOW ?
- Kiểm thử Top-Down hay Bottom-Up

9



#### 3.1. Kiểm thử hộp đen

- Phân đoạn tương đương
- Phân tích giá trị biên
- Đoán lỗi
- Và 1 số kỹ thuật khác



**Black box Data Testing Strategy**

10



## a. Phương pháp phân đoạn tương đương (Equivalence Partition)

- Mục đích: giảm số lượng test bằng cách chọn các tập dữ liệu đại diện
- Thực hiện: Chia dữ kiện vào thành các đoạn, mỗi đoạn đại diện cho một số dữ liệu => việc kiểm thử chỉ thực hiện trên đại diện đó
- Ưu điểm: Test theo mức trừu tượng hơn là trường.
- Áp dụng: màn hình, menu hay mức quá trình.
- Ví dụ:

11



## b. Phương pháp phân tích giá trị biên (Boundary value analysis)

- Là 1 trường hợp riêng của phân đoạn
- Thí dụ: nếu miền dữ liệu là tháng thì giá trị 0 hay >12 là không hợp lệ
- Thường sử dụng trong kiểm thử module

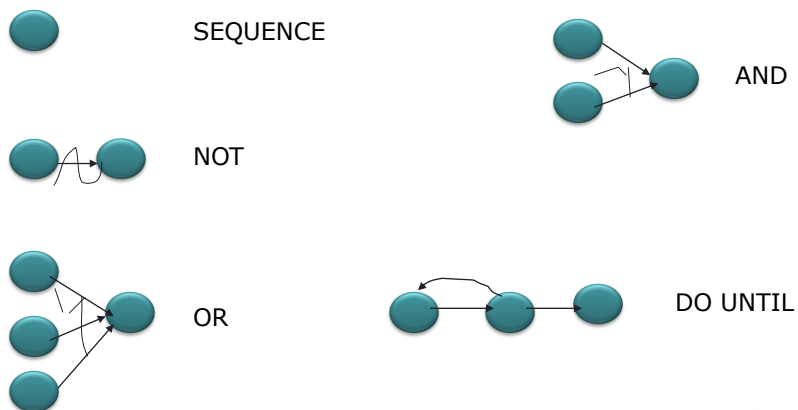
12

### c. Phương pháp đoán lỗi (Error Guessing)

- Dựa vào trực giác và kinh nghiệm
- Thí dụ lỗi chia cho 0. Nếu module có phép chia thì phải kiểm thử lỗi này
- Nhược điểm: không phát hiện hết lỗi

13

### d. Phương pháp đồ thị nguyên nhân - kết quả (Cause-effect Graphing)

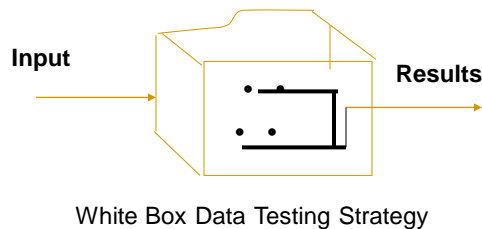


14



## 3.2. Kiểm thử hộp trắng

- Là phương pháp kiểm thử dựa vào cấu trúc điều khiển của các thủ tục để thiết kế các trường hợp kiểm thử.



15



## Kiểm thử hộp trắng (tiếp)

- Người KSPM có thể đảm bảo:
  - Kiểm tra tất cả các lộ trình độc lập bên trong 1 mô đun ít nhất 1 lần
  - Kiểm tra tất cả các nhánh đúng/sai của lựa chọn
  - Kiểm tra việc thực hiện của vòng lặp tại các biên và bên trong vòng lặp
  - Kiểm tra các cấu trúc dữ liệu để đảm bảo tính hợp thức.
- Các kỹ thuật:
  - Kiểm thử theo lộ trình (Basis path testing)
  - Kiểm thử theo cấu trúc điều khiển.

16





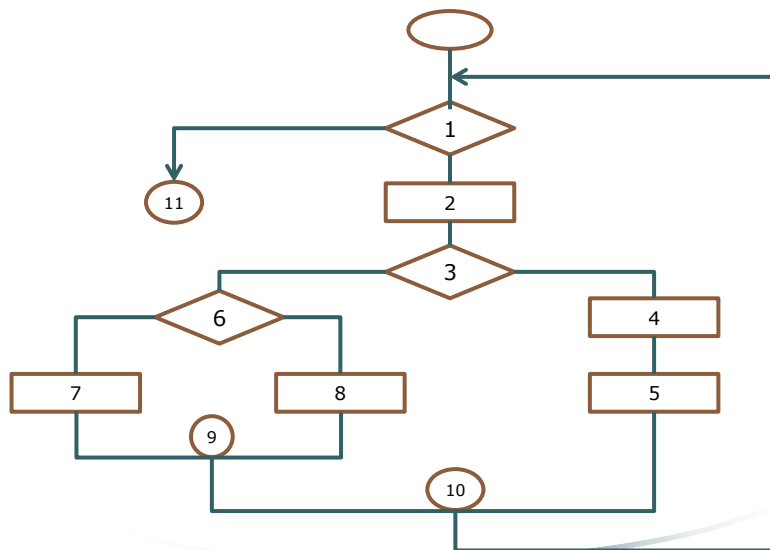
## a. Kiểm thử theo lộ trình

- Là kỹ thuật do Tom McCabe đề xuất cho phép nhà kiểm thử tiến hành 1 số đo về độ phức tạp logic của các thủ tục và số đo này được sử dụng để giúp cho việc định nghĩa các lộ trình cơ bản sao cho các lệnh trong chương trình được thực hiện ít nhất 1 lần trong quá trình kiểm thử.
- Sử dụng Ký pháp đồ họa luồng/ đồ thị chương trình:
  - Mỗi nút đồ thị biểu diễn 1 lệnh/ 1 dãy lệnh liên tiếp
  - Cung của đồ thị biểu diễn luồng điều kiện (trình tự thực hiện).

17



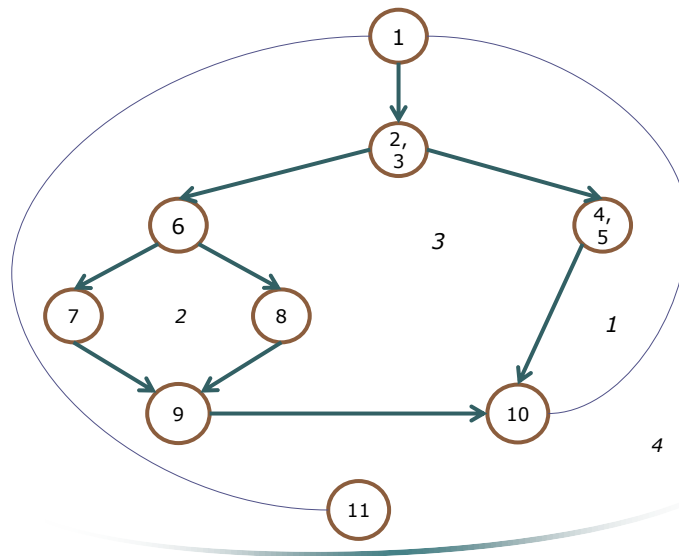
## Ví dụ: lưu đồ khối chương trình



18



## Ví dụ: Đồ thị chương trình



19



## Chú ý

- Một cung bao giờ cũng phải kết thúc tại 1 nút (có thể nút này không tương ứng với bất kỳ lệnh nào trong thủ tục).
- Vùng bao bởi các cung và nút gọi là Region (khi tính, ta phải tính cả vùng bao ngoài).
- Thí dụ đồ thị chương trình ở slide trước gồm 4 vùng (các số in nghiêng).
- Với điều kiện phức tạp (nhiều hơn 1 phép so sánh) thì mỗi so sánh lại tách thành 1 nút riêng.
- Thí dụ: `If a OR b then X else Y Endif`

20

## Độ phức tạp lặp (Cyclomatic Complexity)

- Độ phức tạp lặp là 1 số đo phần mềm, cung cấp 1 đơn vị đo định lượng về độ phức tạp lô gic của CT.
- Trong ngữ cảnh áp dụng kiểm thử theo lộ trình, giá trị này sẽ cung cấp số lượng các lộ trình (path) độc lập trong 1 chương trình và đó được coi như là cận trên của số lượng test phải tiến hành để đảm bảo mọi lệnh đều được thực hiện ít nhất 1 lần.
- Lộ trình độc lập? 1 phần của CT bao gồm ít nhất 1 tập lệnh hay 1 điều kiện mới.
- Đồ thị CT trên có 4 lộ trình độc lập: 1-11; 1-2-3-4-5-10-1-11; 1-2-3-6-8-9-10-1-11; 1-2-3-6-7-9-10-1-11

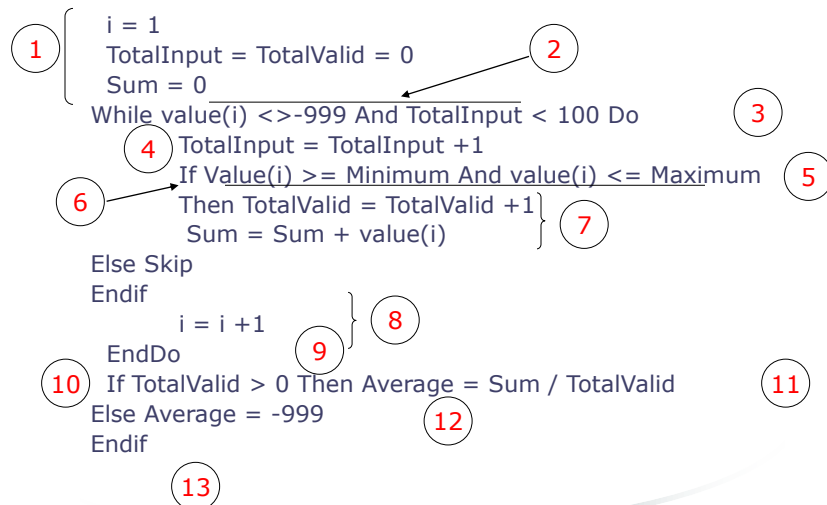
21

## Độ phức tạp lặp

- Có 3 cách tính độ phức tạp lặp ký hiệu  $V(G)$ :
  - $V(G) = E - N + 2$ , với  $E$  là số cung,  $N$  là số nút của  $G$
  - $V(G) = \text{số vùng (region)}$
  - $V(G) = P + 1$ , với  $P$  là số lượng nút Predicat (nút giả định, không có thật).

22

## Thí dụ: chương trình viết bằng PDL



LastUpdate 8-07

□ Dent. of SE, 2004

SE-V.23

## Lời giải

- Số lộ trình độc lập (độ phức tạp lặp) = 6
  - 1-2-10-11-13; 1-2-10-12-13
  - 1-2-3-10-11-13; 1-2-3-4-5-8-9-2 ...
  - 1-2-3-4-5-6-8-9-2...; 1-2-3-4-5-6-7-8-9-2...
  - ...: có nghĩa là phần tiếp theo còn lại đồ thị là chấp nhận được.
- Đồ thị chương trình ?
- Số test phải thực hiện: 6



### 3.3. Trình tự thiết kế

- Kiểm thử module
- Kiểm thử tích hợp
  - Kiểm thử tích hợp trên xuống
  - Kiểm thử tích hợp dưới lên
  - Kiểm thử hồi qui

25



### 4. Kiểm thử module

- Kiểm thử tích hợp module
  - Kiểm thử dưới lên (Bottom-up Test)
  - Kiểm thử trên xuống (Top-down Test)
  - Kiểm thử cột trụ (Big bang Test)
  - Kiểm thử kẹp (Sandwich Test)

26



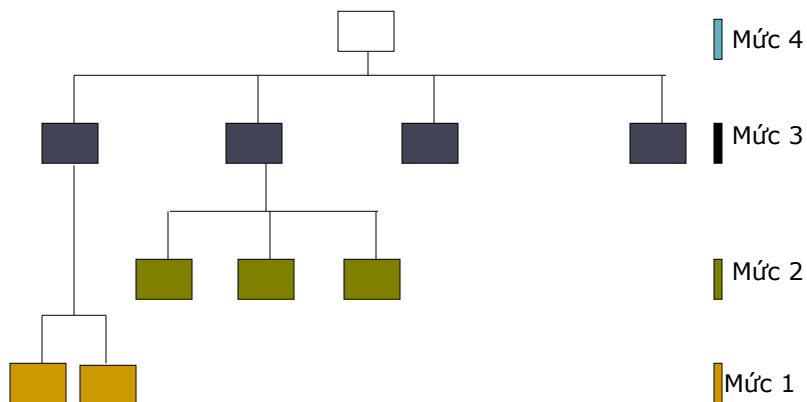
## a. Bottom-up Test

- Các module mức thấp được tổ hợp vào các chùm thực hiện một chức năng con
- Viết trình điều khiển phối hợp vào/ ra và kiểm thử
- Kiểm thử chùm/bó
- Loại bỏ trình điều khiển và chuyển lên mức trên

27



## Bottom-up Test (Tiếp)



28



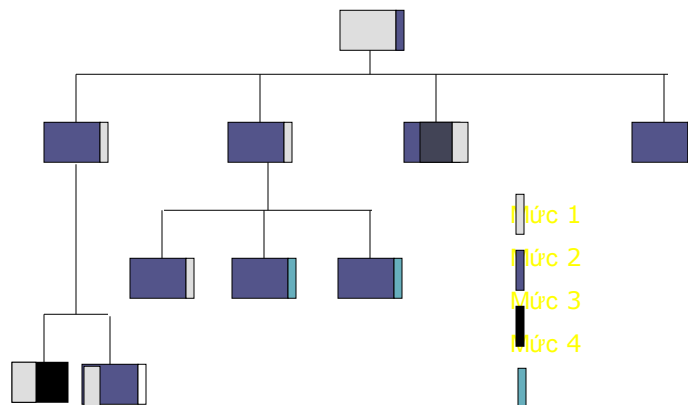
## b. Top-down Test

- module điều khiển chính được dùng như trình điều khiển kiểm thử, gắn các nút con trực tiếp vào nó
- Thay các nút con bằng các module thực tại (theo chiều sâu / ngang)
- Kiểm thử từng module được gắn vào
- Các 1 nút thử xong được thử tiếp nút khác
- Kiểm thử hồi quy

29



## Top-down Test (tiếp)



30



### c. Big bang Test

- Tích hợp không tăng dần
- Tất cả các module đều được tổ hợp trước
- Toàn bộ chương trình được kiểm thử tổng thể
- Khó khăn: khó cô lập lỗi, khi chữa xong lỗi này có thể lỗi mới lại phát sinh

31



### d. Sandwich Test

- Tích hợp trên xuống cho các mức trên cấu trúc chương trình
- Tích hợp dưới lên cho các mức phụ thuộc

32





## 5. Kiểm thử hệ thống

- Kiểm thử phục hồi: bắt buộc phần mềm hỏng nhiều cách để kiểm chứng phục hồi
- Kiểm thử an toàn: kiểm chứng cơ chế bảo vệ
- Kiểm thử gay cấn
- Kiểm thử hiệu năng

33

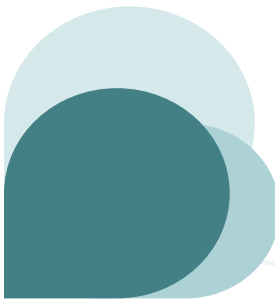


## 6. Kiểm thử chấp nhận

- Mục đích: để bàn giao PM cho khách hàng
- Đối tượng: Cần có sự tham gia của ND
- Trình tự: Dựa vào Yêu cầu PM

34

## PHẦN V: KIỂM THỬ VÀ BẢO TRÌ

- 
- I. Kiểm thử
  - II. Bảo trì
    - 1. Khái niệm
    - 2. Quy trình nghiệp vụ
    - 3. Các vấn đề còn tồn tại
    - 4. Bảo trì trong các phương pháp phát triển phần mềm

35

### 1. Khái niệm

- Bảo trì là công việc tu sửa, thay đổi phần mềm đã được phát triển (chương trình, dữ liệu, JCL, các loại tư liệu đặc tả, . . .) theo những lý do nào đó.
- Các hình thái bảo trì: bảo trì để
  - Tu chỉnh
  - Thích nghi
  - Cải tiến
  - Phòng ngừa

36



## a. Bảo trì để tu sửa

- Là bảo trì khắc phục những khiếm khuyết có trong phần mềm.
- Một số nguyên nhân điển hình
  - Kỹ sư phần mềm và khách hiểu nhầm nhau.
  - Lỗi tiềm ẩn của phần mềm do sơ ý của lập trình hoặc khi kiểm thử chưa bao quát hết.
  - Vấn đề tính năng của phần mềm: không đáp ứng được yêu cầu về bộ nhớ, tệp, . . . .  
Thiết kế sai, biên tập sai . . .
  - Thiếu chuẩn hóa trong phát triển phần mềm (trước đó).
- Kỹ nghệ ngược (Reverse Engineering): dò lại thiết kế để tu sửa.
- Những lưu ý
  - Mức trừu tượng
  - Tính đầy đủ
  - Tính tương tác
  - Tính định hướng

37



## b. Bảo trì để thích hợp

- Là tu chỉnh phần mềm theo thay đổi của môi trường bên ngoài nhằm duy trì và quản lý phần mềm theo vòng đời của nó.
- Thay đổi phần mềm thích nghi với môi trường: công nghệ phần cứng, môi trường phần mềm.
- Những nguyên nhân chính:
  - Thay đổi về phần cứng (ngoại vi, máy chủ, . . .)
  - Thay đổi về phần mềm (môi trường): đổi OS
  - Thay đổi cấu trúc tệp hoặc mở rộng CSDL

38



## c. Bảo trì để cải tiến

- Là việc tu chỉnh hệ phần mềm theo các yêu cầu ngày càng hoàn thiện hơn, đầy đủ hơn, hợp lý hơn.
- Những nguyên nhân chính:
  - Do muốn nâng cao hiệu suất nên thường hay cải tiến phương thức truy cập tệp.
  - Mở rộng thêm chức năng mới cho hệ thống.
  - Cải tiến quản lý kéo theo cải tiến tư liệu vận hành và trình tự công việc.
  - Thay đổi người dùng hoặc thay đổi thao tác.
- Còn gọi là tái kỹ nghệ (re-engineering)
- Mục đích: đưa ra một thiết kế cùng chức năng nhưng có chất lượng cao hơn.
- Các bước thực hiện:
  - Xây dựng lưu đồ phần mềm
  - Suy dẫn ra biểu thức Bun cho từng dãy xử lý
  - Biên dịch bằng chân lí
  - Tái cấu trúc phần mềm

39



## d. Bảo trì để phòng ngừa

- Là công việc tu chỉnh chương trình có tính đến tương lai của phần mềm đó sẽ mở rộng và thay đổi như thế nào.
- Thực ra trong khi thiết kế phần mềm đã phải tính đến tính mở rộng của nó, nên thực tế ít khi ta gặp bảo trì phòng ngừa nếu như phần mềm được thiết kế tốt.
- Mục đích: sửa đổi để thích hợp với yêu cầu thay đổi sẽ có của người dùng.
- Thực hiện những thay đổi trên thiết kế không tưởng minh.
- Hiểu hoạt động bên trong chương trình
- Thiết kế / lập trình lại.
- Sử dụng công cụ CASE

40



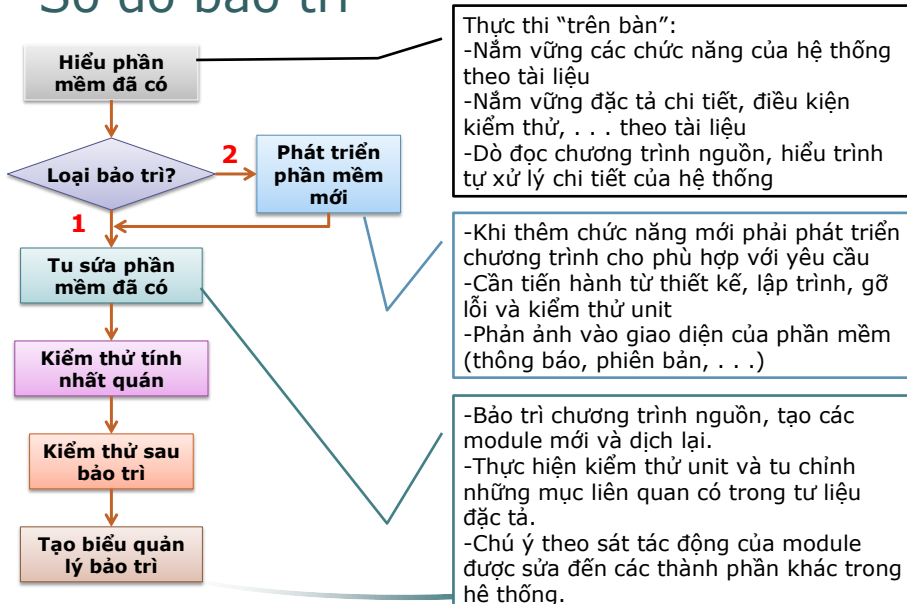
## 2. Quy trình nghiệp vụ

- Quy trình bảo trì: quá trình trong vòng đời của phần mềm, cũng tuân theo các pha phân tích, thiết kế, phát triển và kiểm thử từ khi phát sinh vấn đề cho đến khi giải quyết xong.
- Các nhiệm vụ bảo trì:
  - Phân tích/cô lập: phân tích tác động, phân tích những giá trị lợi ích, và cô lập các thành phần cần bảo trì
  - Thiết kế: thiết kế lại hệ thống (phải biết cách tu sửa, thay đổi).
  - Thực thi: thay thế mã nguồn và kiểm soát từng đơn vị thành phần hệ thống, có tính đến thời gian lập trình.
- Thao tác bảo trì: Gồm 2 loại
  - Tu chỉnh cái đã có (loại 1)
  - Thêm cái mới (loại 2)

41

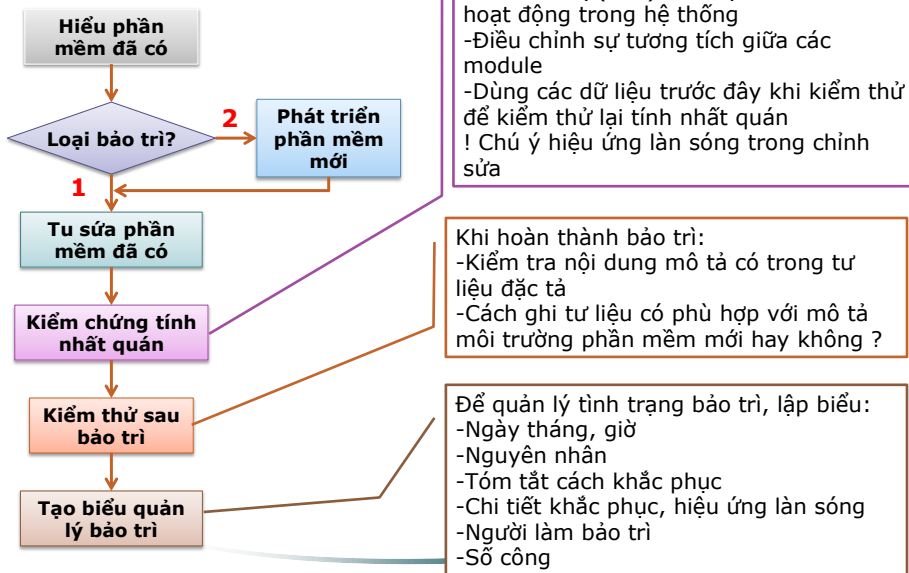


## Sơ đồ bảo trì





## Sơ đồ bảo trì



## 3. Các vấn đề còn tồn tại

- Phương pháp cải tiến thao tác bảo trì:
  - Sáng kiến trong quy trình phát triển phần mềm
  - Sáng kiến trong quy trình bảo trì phần mềm
  - Phát triển những kỹ thuật mới cho bảo trì

## a. Sáng kiến trong quy trình phát triển phần mềm

- Chuẩn hóa mọi khâu trong phát triển phần mềm
- Người bảo trì chủ chốt tham gia vào giai đoạn phân tích và thiết kế
- Thiết kế để dễ bảo trì

45

## b. Sáng kiến trong quy trình bảo trì phần mềm

- Sử dụng các công cụ hỗ trợ phát triển phần mềm
- Chuẩn hóa thao tác bảo trì và thiết bị môi trường bảo trì
- Lưu lại những thông tin sử dụng bảo trì
- Dự án nên cử một người chủ chốt của mình làm công việc bảo trì sau khi dự án kết thúc giai đoạn phát triển.

46



## c. Phát triển những kỹ thuật mới cho bảo trì

- Công cụ phần mềm hỗ trợ bảo trì
- Cơ sở dữ liệu cho bảo trì
- Quản lý tài liệu, quản lý dữ liệu, quản lý chương trình nguồn, quản lý dữ liệu thử, quản lý sử bảo trì
- Trạm bảo trì tính năng cao trong hệ thống mạng lưới bảo trì với máy chủ thông minh.

47



## Bài tập về nhà

- Tìm hiểu SMMM – Software Maintenance Maturity Model.

48



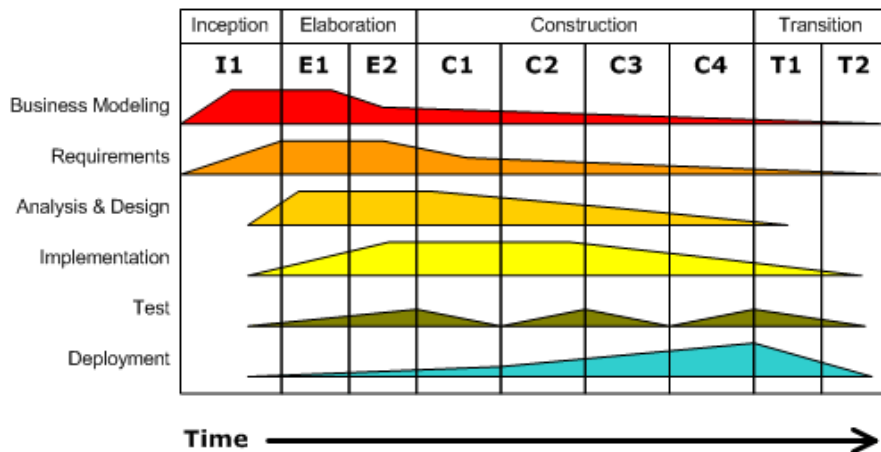
## a. Phát triển lặp



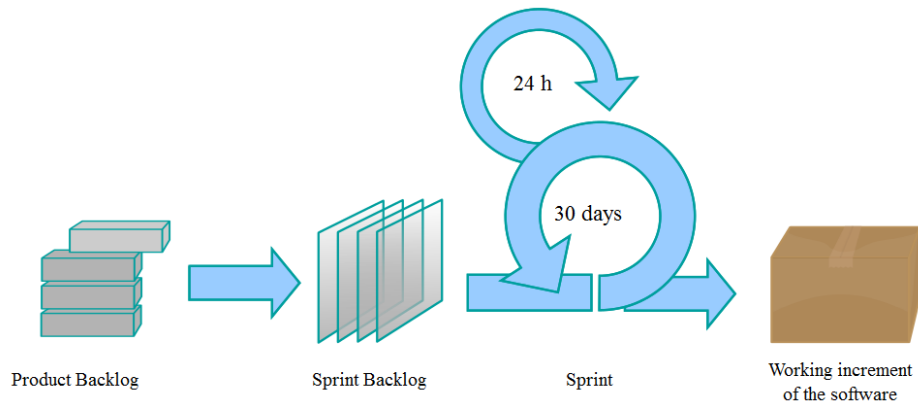
## RUP

### Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.

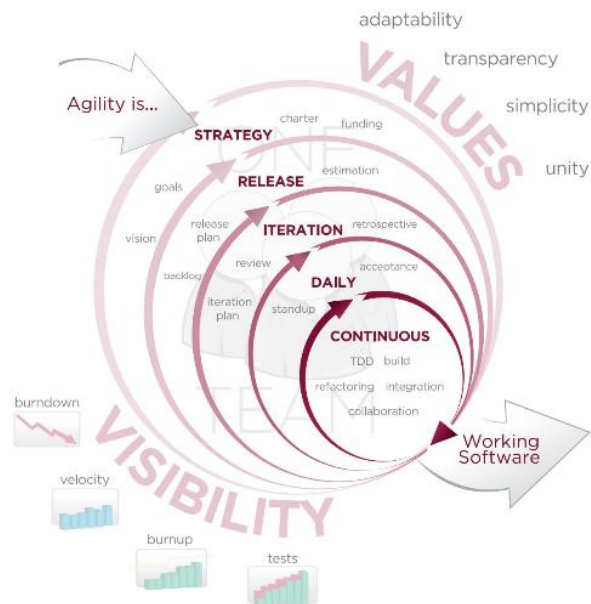


## Scrum



51

## Agile



52



## b. Hướng thành phần

- Những hoạt động bảo trì chính ở CBSD
  - Gắn kết hóa và gói hóa
  - May đo hóa
  - Phát hiện lỗi và cô lập
  - Cập nhật cấu hình thành phần
  - Theo dõi và kiểm tra các hành vi hệ thống
  - Kiểm thử các thành phần

53



## c. Mã nguồn mở

- Sự khác biệt với bảo trì theo phương pháp truyền thống
  - Phiên bản ngày tháng
  - Chờ đợi dịch vụ

54