ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Nhập môn Công nghệ Phần mềm

(Introduction to Software Engineering)

CHƯƠNG 9

Đảm bảo chất lượng phần mềm

Software Quality Assurance

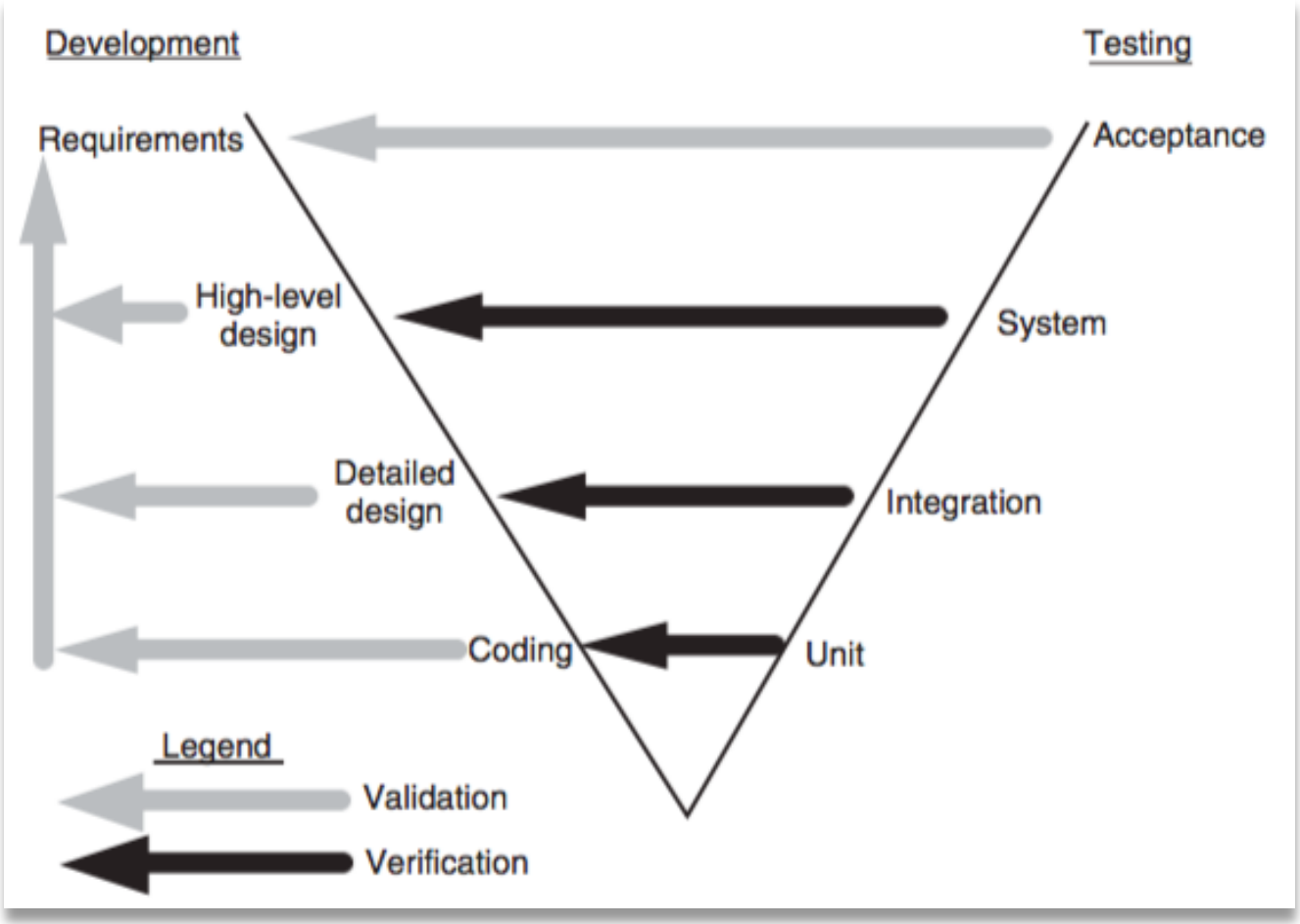
Nội dung

1. Mô hình V&V
2. Các thuật ngữ
3. Phương pháp kiểm thử hộp trắng
4. Phương pháp kiểm thử hộp đen
5. Quản lý chất lượng phần mềm
6. Bảo trì phần mềm

1. Mô hình V & V

- Mô hình V&V: Verification&Validation
- Verification : xác minh
  - Verify if the system is built right
- Validation: kiểm chứng
  - Validate if we built a right system
- Verification là hoạt động đánh giá hệ thống phần mềm bằng cách xác định xem hệ thống có thoả mãn các yêu cầu nêu ra ban đầu hay không (from perspective of development team)
- Validation là hoạt động kiểm chứng xem sản phẩm phần mềm có thoả mãn các yêu cầu của người dùng hay không (from perspective of users)

1. Mô hình V & V


Development
Testing
Requirements
Acceptance
High-level design
System
Detailed design
Integration
Coding
Unit
Legend
Validation
Verification

Mô hình V & V

- Kiểm thử được thực hiện ở các mức khác nhau tương ứng với các phase trong quá trình phát triển phần mềm
- Kiểm thử đơn vị
- Kiểm thử tích hợp
- Kiểm thử hệ thống
- Kiểm thử xác nhận

2. Một số thuật ngữ cơ bản

- Failure: là một sự kiện xảy ra bất cứ khi nào hệ thống có một hành vi không tương ứng với đặc tả
- Fault: hay Bug là lỗi hay những bất thường trong phần mềm
- Error: là một trạng thái của hệ thống gây ra bởi một Fault
- Software Testing: là một quy trình phân tích và đánh giá một thành phần phần mềm (software item) để phát hiện sự khác biệt giữa những gì tồn tại và những yêu cầu đặt ra

2. Một số thuật ngữ cơ bản

- Test case: ca kiểm thử là một cặp <input, expected outcome>
  - Khi thực hiện một ca kiểm thử người ta sẽ đưa input tương ứng vào chương trình và kiểm tra kết quả trả về của hệ thống
  - Passed test: là trạng thái sau khi thực hiện một ca kiểm thử thu được kết quả trả về tương ứng với kết quả mong đợi
  - Failed test: ngược lại
- Test suite: là một tập hợp các test case được xây dựng để kiểm thử cho 1 chức năng hoặc 1 nhóm chức năng của phần mềm

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

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 trướcđó để tránh ảnh hưởng lan truyền sóng

Các kĩ thuật kiểm thử

- Có 2 kỹ thuật kiểm thử cơ bản:
  - Kiểm thử hộp đen – black box testing
  - Kiểm thử hộp trắng – white box testing
- Cả hai kỹ thuật này đều yêu cầu phải thực thi chương trình để xem xét kết quả
- Giúp dễ dàng thiết kế các ca kiểm thử để có thể phát hiện nhiều nhất các lỗi có thể trong phần mềm
- Sử dụng kỹ thuật nào phụ thuộc vào các giai đoạn (các mức kiểm thử khác nhau) trong quá trình phát triển phần mềm

3. Kỹ thuật kiểm thử hộp trắng

- Kiểm thử hộp trắng là 1 chiến lược kiểm thử dựa trên cấu trúc của chức năng cần kiểm thử để thiết kế các ca kiểm thử
- Structural Testing
- Thường được thực hiện bởi lập trình viên và có thể được triển khai bởi các kiểm thử viên
- Thực hiện chủ yếu ở giai đoạn kiểm thử đơn vị và kiểm thử tích hợp

Các phương pháp thiết kế ca kiểm thử

- Kiểm thử luồng điều khiển
  - Bao phủ tất cả đường dẫn chương trình
  - Bao phủ lệnh
  - Bao phủ nhánh
- Kiểm thử luồng dữ liệu
  - Bao phủ tất cả các điểm định nghĩa của biến
  - Bao phủ tất cả các điểm sử dụng của biến
  - ...

Các phương pháp thiết kế ca kiểm thử

- **Kiểm thử luồng điều khiển**
  - Bao phủ tất cả đường dẫn chương trình
  - Bao phủ lệnh
  - Bao phủ nhánh
- Kiểm thử luồng dữ liệu
  - Bao phủ tất cả các điểm định nghĩa của biến
  - Bao phủ tất cả các điểm sử dụng của biến
  - ...

Đồ thị luồng điều khiển
Control Flow Graph

- Biểu diễn đồ thị cấu trúc của một đơn vị chương trình
- 1 đường dẫn (path) của chương trình
  - Là một chuỗi các câu lệnh (statements) từ điểm bắt đầu cho đến điểm kết thúc của chương trình
  - Với mỗi một input data, chương trình có thể thực thi theo các đường dẫn khác nhau (execution path)
- Mục đích của việc thiết kế các ca kiểm thử trong kiểm thử cấu trúc:
  - Tìm ra những đường dẫn (path) tại đó lỗi hay xảy ra hoặc tiềm ẩn nhiều lỗi nhất có thể mà không cần phải bao phủ tất cả các path của chương trình

Đồ thị luồng điều khiển
Control Flow Graph

- Có 3 loại statements:
  - tuần tự
  - rẽ nhánh
  - lặp
- Các tiêu chí lựa chọn đường dẫn chương trình
  - Bao phủ toàn bộ path ➔ vét cạn
    - Chương trình có 10 lệnh rẽ nhánh true/false ➔ Số đường dẫn tối đa là $2^{10} = 1024$ đường dẫn ➔ 1024 test case!!!
  - Bao phủ các lệnh
    - Đảm bảo mọi câu lệnh trong chương trình đều được test
  - Bao phủ các nhánh
    - Đảm bảo mọi nhánh rẽ true/false trong chương trình đều được test

Example

- Chương trình có source code là hàm AccClient với input là tuổi và giới tính như sau:

Life Insurance Example

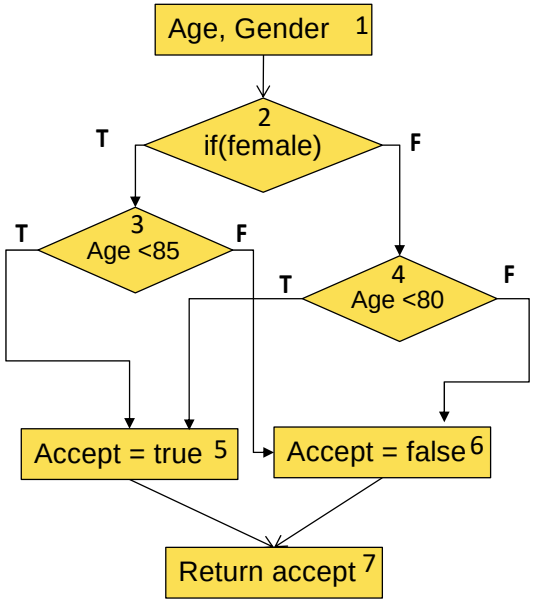
```
bool AccClient(agetype
  age; gndrtype gender)
bool accept
  if(gender=female)
     accept := age < 85;
   else
     accept := age < 80;
return accept
```

Example

- Vẽ sơ đồ luồng điều khiển

Life Insurance Example

```
bool AccClient(agetype
  age; gndrtype gender)
bool accept
  if(gender=female)
     accept := age < 85;
   else
     accept := age < 80;
return accept
```

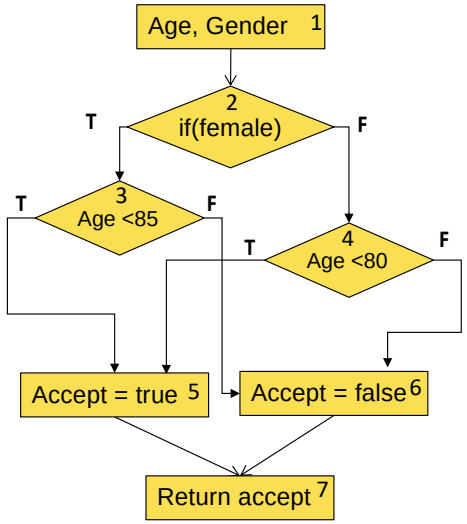


All path coverage

- Lựa chọn tất cả các path có thể trong chương trình
- Mỗi path sẽ tương ứng với 1 test case

All paths

Female	Age < 85	Age < 80
Yes	Yes	Yes
Yes	Yes	No
~~Yes~~	~~No~~	~~Yes~~
Yes	No	No
No	Yes	Yes
No	Yes	No
~~No~~	~~No~~	~~Yes~~
No	No	No

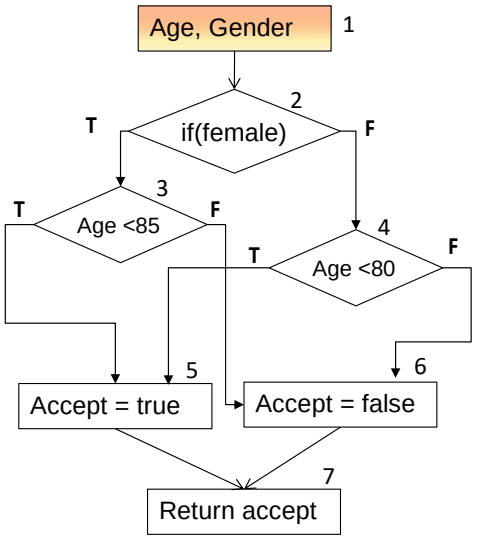


<Yes,Yes,*> 1-2(T)-3(T)-5-7
<Yes,No,No> 1-2(T)-3(F)-6-7
<No,Yes,Yes> 1-2(F)-4(T)-5-7
<No,*,No> 1-2(F)-4(F)-6-7

Statement Coverage

- Đảm bảo mỗi một câu lệnh đều được thực thi ít nhất 1 lần ➔ Đảm bảo mọi đỉnh của đồ thị đều được xuất hiện ít nhất 1 lần trong các đường dẫn

```
bool AccClient(agetype
  age; gndrtype gender)
bool accept
  if(gender=female)
    accept := age < 85;
   else
    accept := age < 80;
return accept
```



AccClient(83, female)->accept

AccClient(83, male) ->reject

Branch coverage

- Mục đích: đảm bảo mỗi một điểm quyết định trên đồ thị (điểm tại đó có nhánh rẽ) đều được kiểm thử ít nhất 1 lần với mỗi nhánh true/false

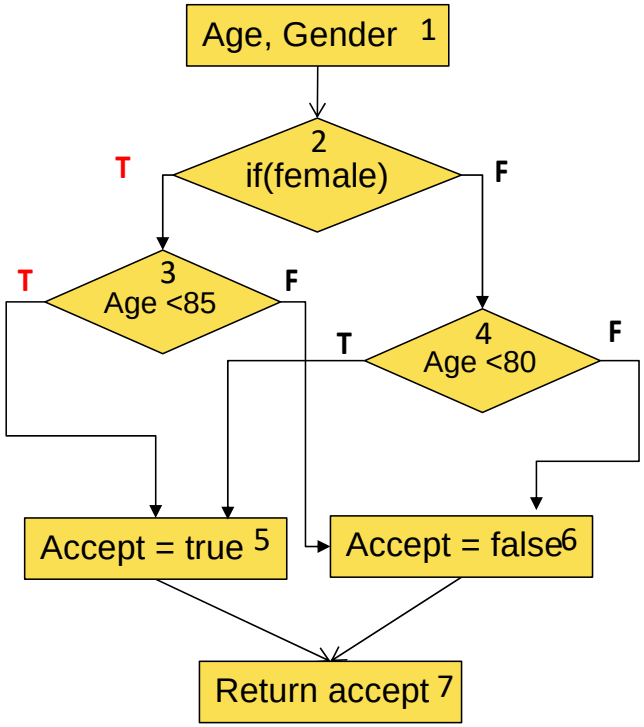
Example

Branch Coverage /1

AccClient(83, female)->accept

```
bool AccClient(agetype
  age; gndrtype gender)
bool accept
  if(gender=female)
     accept := age < 85;
   else
     accept := age < 80;
return accept
```

true



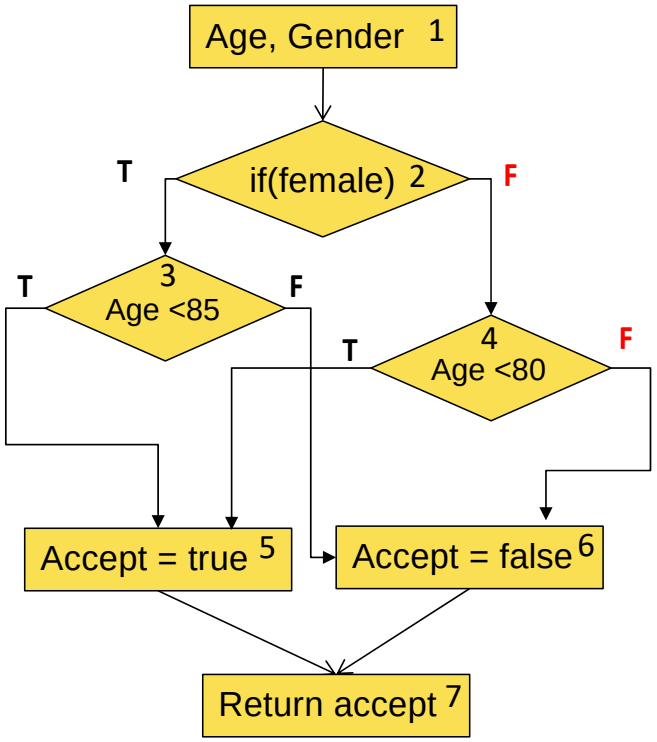
Example

Branch Coverage /2

AccClient(83, male)
->reject

```
bool AccClient(agetype
  age; gndrtype gender)
bool accept
  if(gender=female)
     accept := age < 85;
   else
     accept := age < 80;
return accept
```

false



Example

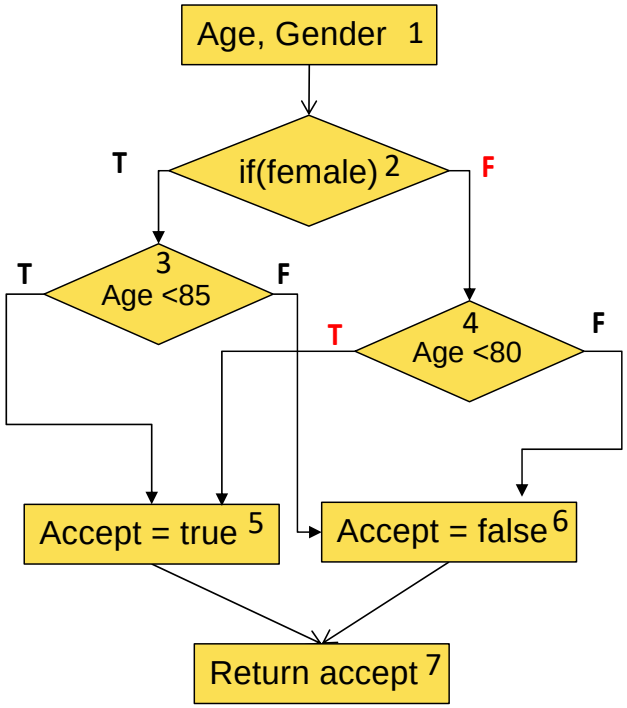
Branch Coverage /3

AccClient(78, male)->accept

```
bool AccClient(agetype
  age; gndrtype gender)
bool accept
  if(gender=female)
     accept := age < 85;
   else
     accept := age < 80;
return accept
```

true

true



Example

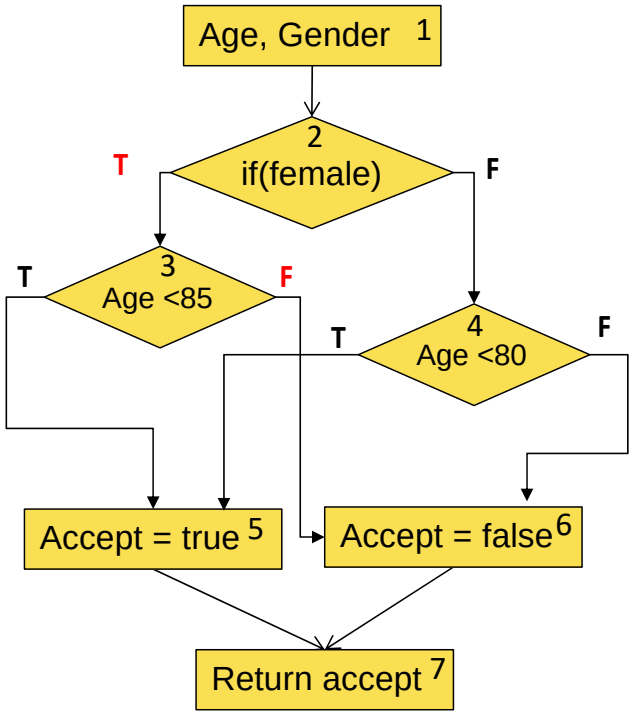
Branch Coverage /4

AccClient(88, female) ->reject

```
bool AccClient(agetype
  age; gndrtype gender)
bool accept
  if(gender=female)
     accept := age < 85;
   else
     accept := age < 80;
return accept
```

false

false



So sánh các tiêu chí

- Tiêu chí mạnh nhất: bao phủ mọi đường dẫn
- Tiêu chí yếu nhất: bao phủ mọi câu lệnh
  - Các câu lệnh rẽ nhánh có thể chỉ cần kiểm thử 1 nhánh true/false
- Tiêu chí bao phủ điểm quyết định thường được sử dụng.

4. Kỹ thuật kiểm thử hộp đen

- Kiểm thử hộp đen là 1 chiến lược kiểm thử
  - Thực hiện dựa trên yêu cầu và đặc tả chức năng
  - Xác minh đầu ra của chức năng mà không quan tâm tới cấu trúc bên trong
- Còn được gọi với tên khác là kiểm thử chức năng (functional testing)
- Thường được thực hiện bởi các kiểm thử viên, kĩ sư đảm bảo chất lượng, và chủ yếu được sử dụng ở giai đoạn kiểm thử hệ thống và kiểm thử chấp nhận

Các phương pháp thiết kế ca kiểm thử

- Inputs của kiểm thử hộp đen là những đặc tả chi tiết của chức năng
- Outputs là các hành xử của hệ thống tương ứng với từng loại dữ liệu đầu vào và kịch bản kiểm thử
- Các phương pháp thiết kế ca kiểm thử:
  - Phân chia lớp tương đương
  - Phân tích giá trị biên
  - Sử dụng bảng quyết định
  - Kiểm thử trạng thái

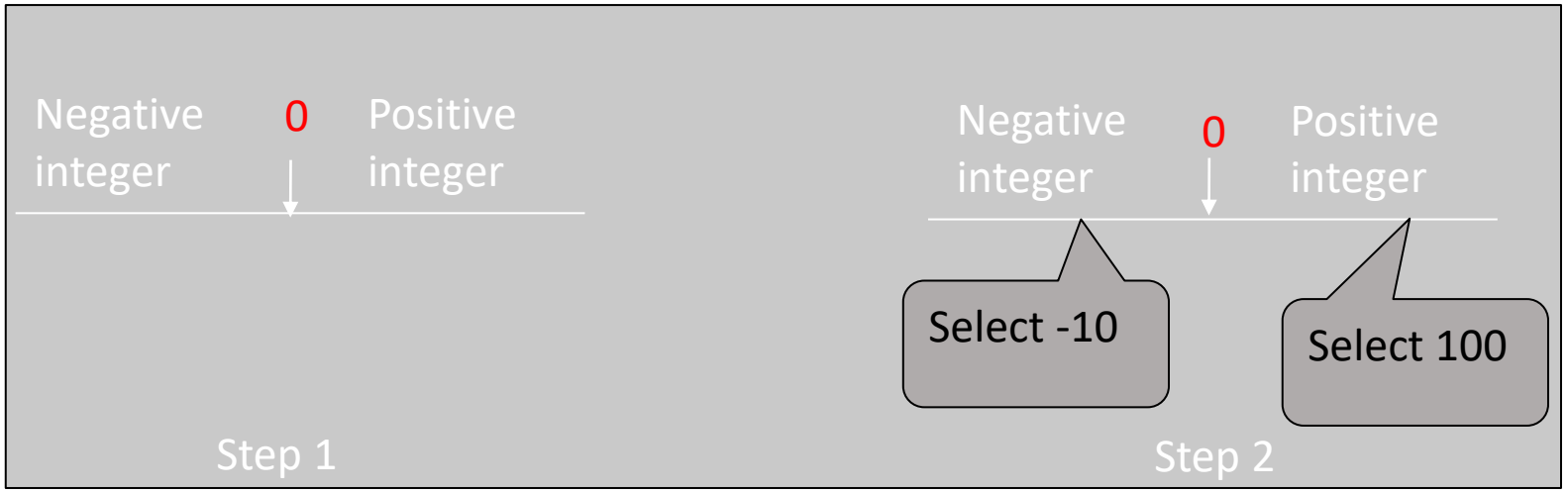
Các phương pháp thiết kế ca kiểm thử

- Inputs của kiểm thử hộp đen là những đặc tả chi tiết của chức năng
- Outputs là các hành xử của hệ thống tương ứng với từng loại dữ liệu đầu vào và kịch bản kiểm thử
- Các phương pháp thiết kế ca kiểm thử:
  - **Phân chia lớp tương đương**
  - Phân tích giá trị biên
  - Sử dụng bảng quyết định
  - Kiểm thử trạng thái

Kĩ thuật phân chia lớp tương đương

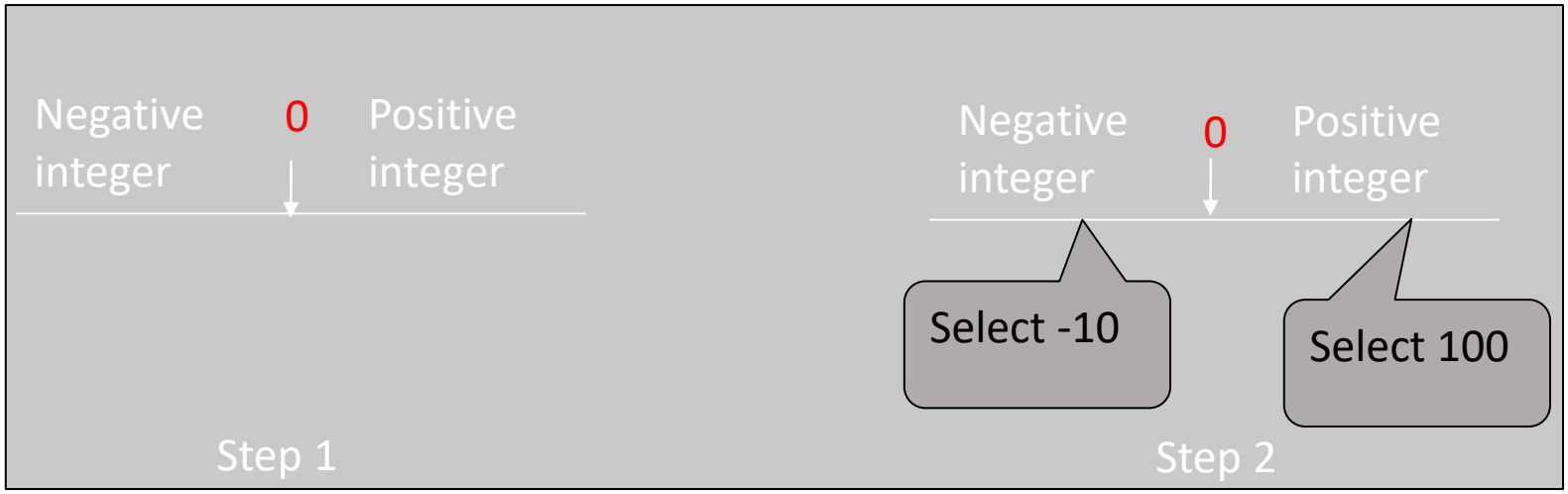
- Dựa vào miền dữ liệu trong đặc tả của inputs và outputs để xác định các phân lớp tương đương
- 1 phân lớp tương đương được định nghĩa là 1 miền dữ liệu (input hoặc output) trong đó với mọi điểm dữ liệu thuộc miền, chức năng được test thực hiện cùng một hành vi

Kĩ thuật phân chia lớp tương đương



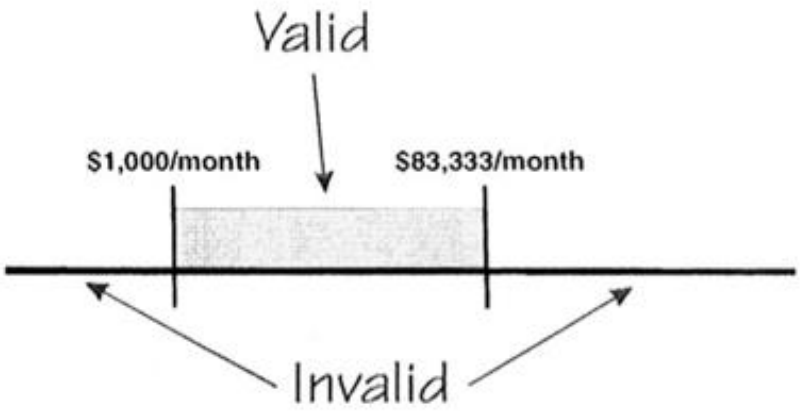
- Một hàm thực hiện kiểm tra xem dữ liệu số nguyên truyền vào là âm hay dương.
- Input: số nguyên Output: true/false

Kĩ thuật phân chia lớp tương đương



- Step 1: Xác định được 2 lớp tương đương ứng với input >=0 và input < 0
- Step 2: Ở mỗi phân lớp tương đương lấy 1 giá trị đại diện để xây dựng ca kiểm thử ➔ 2 test cases

Ưu điểm



Giảm số lượng test case, thay vì phải quét toàn bộ miền dữ liệu của input

Đảm bảo mỗi một lớp đều được test

Nhược điểm

- Việc phân chia lớp tương đương như thế nào phụ thuộc vào kinh nghiệm của kiểm thử viên
  - Một số kiểm thử viên có thể bỏ qua các phân lớp dữ liệu không hợp lệ (invalid data)
- Không xem xét các giá trị tại biên của các lớp
  - Kết hợp với phân tích giá trị biên để xây dựng các ca kiểm thử
- Khi dữ liệu có mối quan hệ với nhau không còn độc lập nữa thì việc phân chia rõ ràng các lớp tương đương sẽ khó
  - Kết hợp với bảng quyết định để xây dựng các ca kiểm thử

5. Quy trình đảm bảo chất lượng phần mềm

- Chất lượng phần mềm được định nghĩa là
  - Mức độ một hệ thống hay một thành phần đạt được các yêu cầu đặt ra
  - Mức độ một hệ thống hay một thành phần đạt được những mong đợi của khách hàng hay người dùng hệ thống đó
- Chất lượng được nhìn nhận theo 2 quan điểm
  - Quan điểm của người phát triển
  - Quan điểm của người dùng

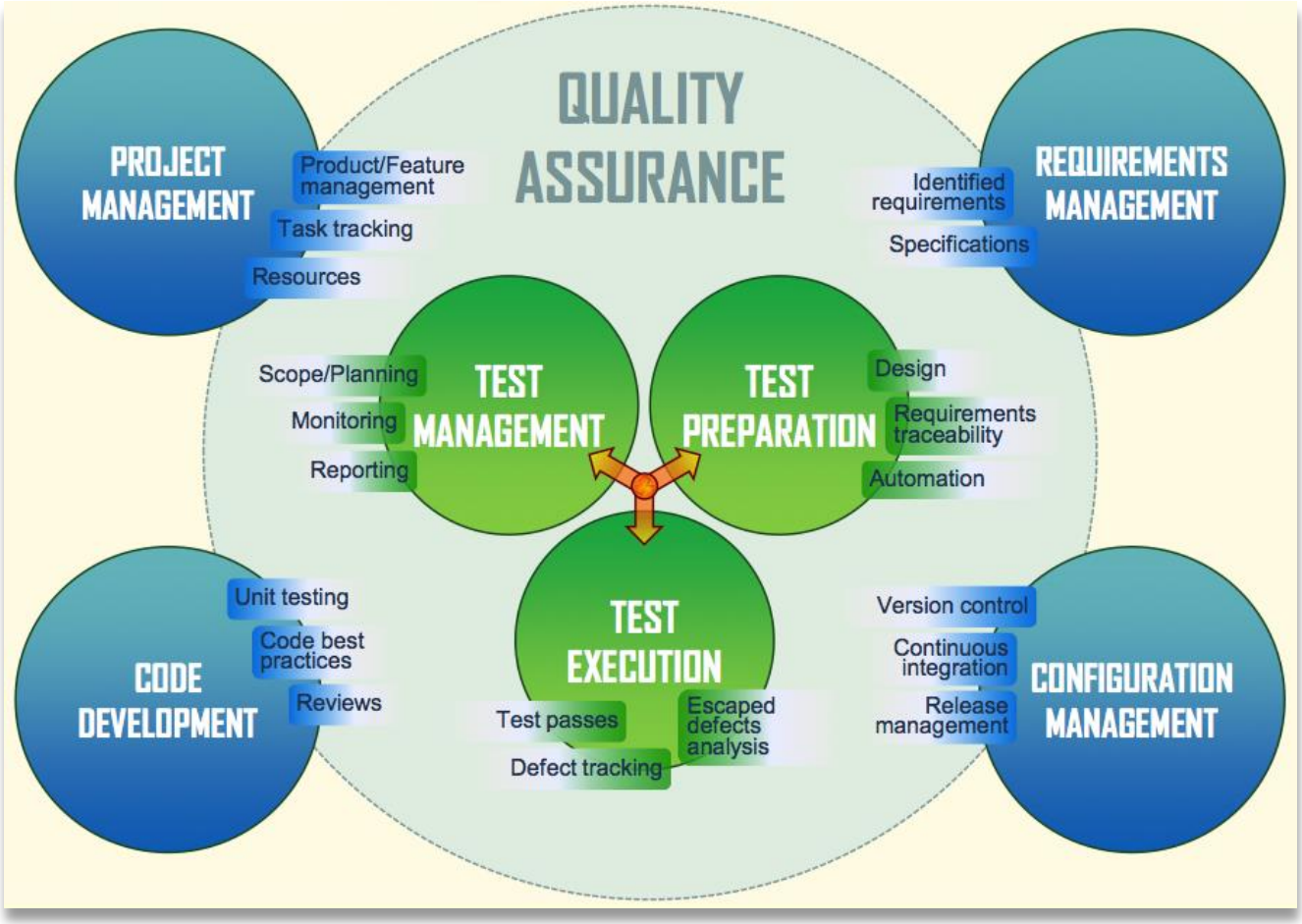
Các vấn đề trong việc đảm bảo chất lượng

- Quan điểm về chất lượng giữa người dùng và đội phát triển khác nhau
  - Đội phát triển quan tâm đến khía cạnh phát triển hệ thống: tái sử dụng, dễ kiểm thử, dễ bảo trì
  - Người dùng quan tâm đến hiệu quả, hiệu suất và độ tin cậy của phần mềm
  - Đôi khi để đảm bảo cho code sạch, dễ tái sử dụng, dễ bảo trì mà các tiêu chí chất lượng khác như hiệu quả, độ tin cậy có thể giảm đi
- Đặc tả phần mềm không bao giờ là đầy đủ và nhất quán ngay từ đầu ➔ Rất khó để quy chuẩn được chất lượng phần mềm ngay từ đầu

Quy trình đảm bảo chất lượng

- Quy trình đảm bảo chất lượng là một quy trình mang tính hệ thống được thực hiện để kiểm chứng xem sản phẩm hoặc dịch vụ đang được phát triển có đạt được các yêu cầu đặc tả ban đầu hay không?

Quy trình đảm bảo chất lượng



Quy trình đảm bảo chất lượng

- Tích hợp với tất cả các quy trình khác trong toàn bộ vòng đời phát triển của phần mềm
  - Quản lý dự án
  - Quản lý yêu cầu
  - Quản lý cấu hình
  - Quản lý mã nguồn
- Quy trình kiểm thử là nền tảng của quy trình QA: 3 hoạt động chính
  - Quản lý kiểm thử
  - Chuẩn bị kiểm thử / Kế hoạch kiểm thử
  - Thực thi kiểm thử

6. Bảo trì phần mềm

- 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

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

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

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 bảng chân lí
  - Tái cấu trúc phần mềm

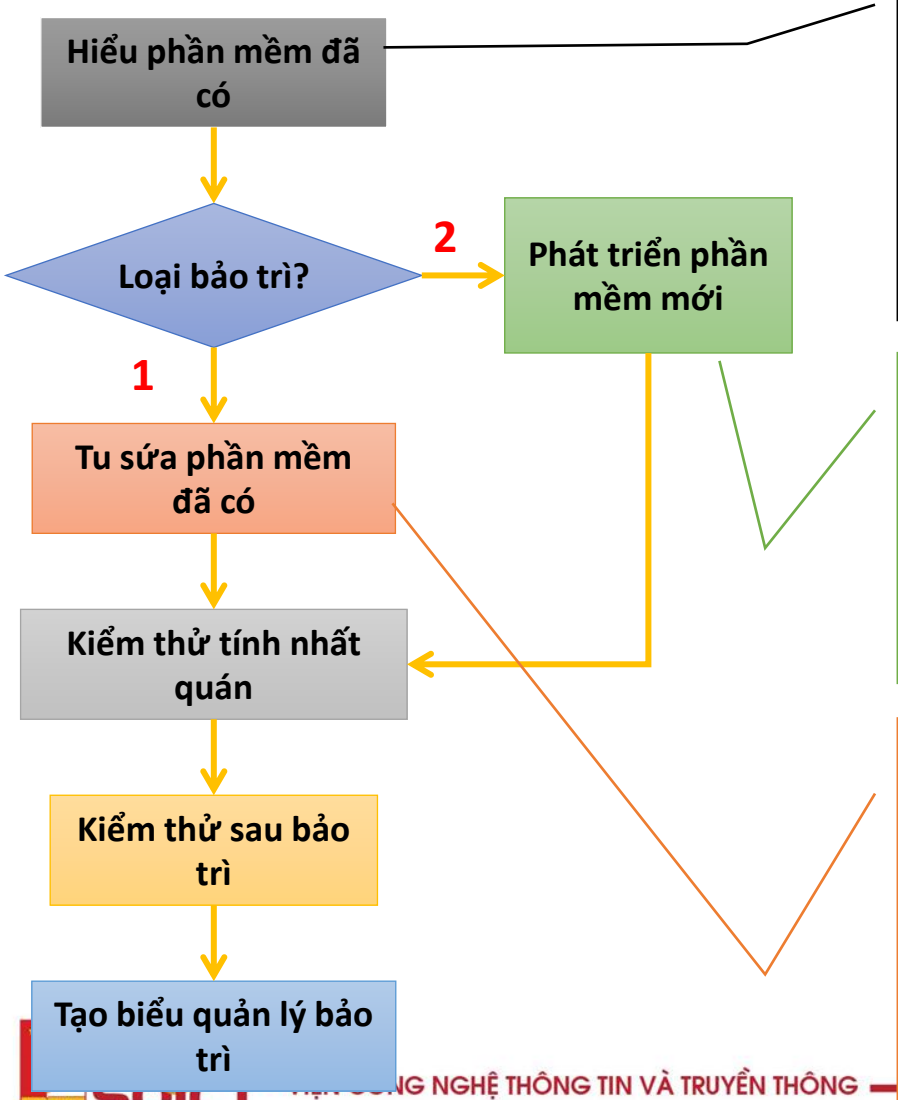
d. Bảo trì để phòng ngừa

- 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 tường minh.
- Hiểu hoạt động bên trong chương trình
- Thiết kế / lập trình lại.
- Sử dụng công cụ CASE

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 cải đã có (loại 1)
  - Thêm cái mới (loại 2)

Sơ đồ bảo trì

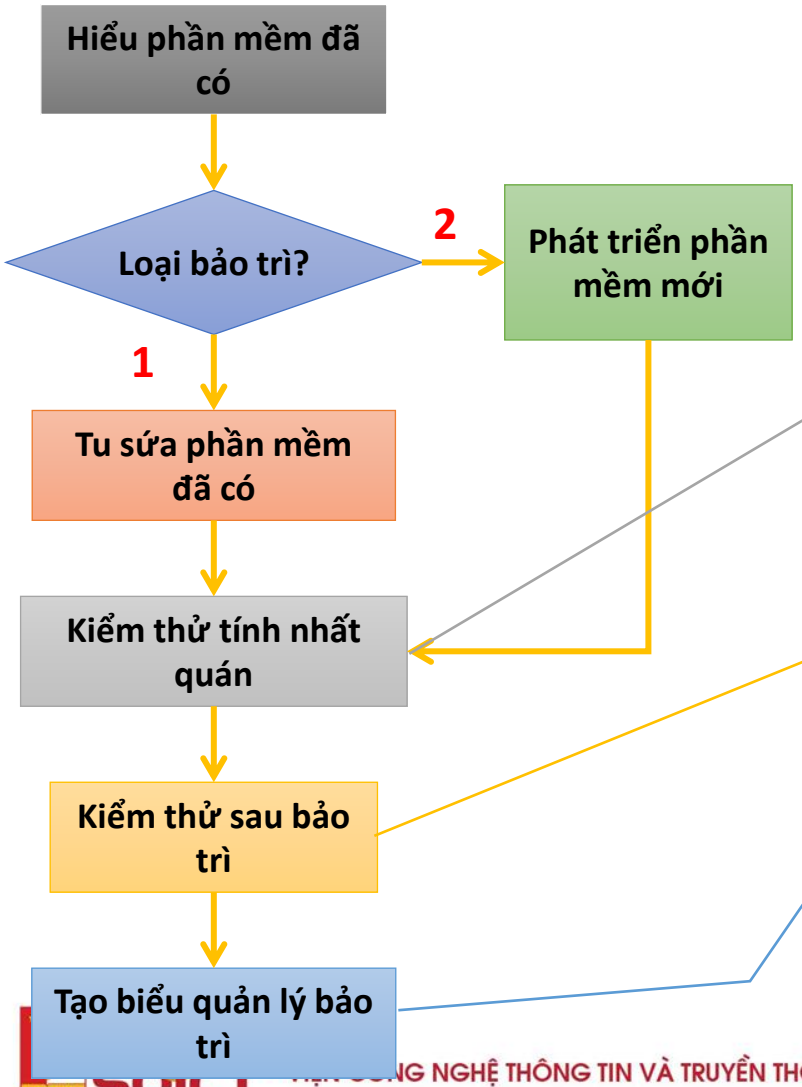


Thực thi "trên bàn":
-Nắm vững các chức năng của hệ thống theo tài liệu
-Nắm vững đặc tả chi tiết, điều kiện kiểm thử, . . . theo tài liệu
-Dò đọc chương trình nguồn, hiểu trình tự xử lý chi tiết của hệ thống

-Khi thêm chức năng mới phải phát triển chương trình cho phù hợp với yêu cầu
-Cần tiến hành từ thiết kế, lập trình, gỡ lỗi và kiểm thử unit
-Phản ảnh vào giao diện của phần mềm (thông báo, phiên bản, . . .)

-Bảo trì chương trình nguồn, tạo các module mới và dịch lại.
-Thực hiện kiểm thử unit và tu chỉnh những mục liên quan có trong tư liệu đặc tả.
-Chú ý theo sát tác động của module được sửa đến các thành phần khác trong hệ thống.

Sơ đồ bảo trì



Bằng kiểm thử tích hợp
-Đưa đơn vị (unit) đã dược kiểm thử vào hoạt động trong hệ thống
-Điều chỉnh sự tương tích giữa các module
-Dùng các dữ liệu trước đây khi kiểm thử để kiểm thử lại tính nhất quán
! Chú ý hiệu ứng làn sóng trong chỉnh sửa

Khi hoàn thành bảo trì:
-Kiểm tra nội dung mô tả có trong tư liệu đặc tả
-Cách ghi tư liệu có phù hợp với mô tả môi trường phần mềm mới hay không ?

Để quản lý tình trạng bảo trì, lập biểu:
-Ngày tháng, giờ
-Nguyên nhân
-Tóm tắt cách khắc phục
-Chi tiết khắc phục, hiệu ứng làn sóng
-Người làm bảo trì
-Số công

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

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 bảo trì
- Dự án nên cử một người chủ chốt của mình làm công việc bảo trì sau khi dự án kết thưc giai đoạn phát triển.

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.

Kiểm thử và bảo trì

- Kiểm thử
  - Phát hiện lỗi, đánh giá phần mềm
  - Thực hiện trong và sau quá trình phát triển phần mềm
- Bảo trì
  - Thay đổi, cải tiến phần mềm đã được phát triển
  - Thực hiện sau khi phần mềm được đưa vào sử dụng


Q&A

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions !

soict.hust.edu.vn/ fb.com/groups/soict