TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

NHẬP MÔN CNPM

Nội dung / Chương 11-2: Kiểm thử hộp trắng và hộp đen

Thông tin GV

Nội dung chương

Phần 1. ABC

Phần 2. ABC

Phần 3. ABC

Phần 4. ABC

Phần 5. ABC

Phần 6. ABC

Testability

Testability

- Operability— thực hiện hiệu quả, đúng đắn.
- Observability— mọi kết quả của các test case đều có thể quan sát được.
- Controllability— mỗi trường hợp testing có thể thực hiện và tối ưu được.
- Decomposability— testing có thể hướng mục đích.
- Simplicity— giảm thiểu cấu trúc và logic phức tạp để test đơn giản hơn.
- Stability— có thể sửa đổi mức độ giới hạn trong quá trình test.
- Understandability— thuộc về thiết kế.

Test thế nào gọi là tốt?

- Test có xác suất tìm ra lỗi lớn.
- Không thừa thãi.
- Best of bread: giữa nhiều cách test nên chọn cách hiệu quả nhất.
- Không được quá đơn giản hay quá phức tạp.

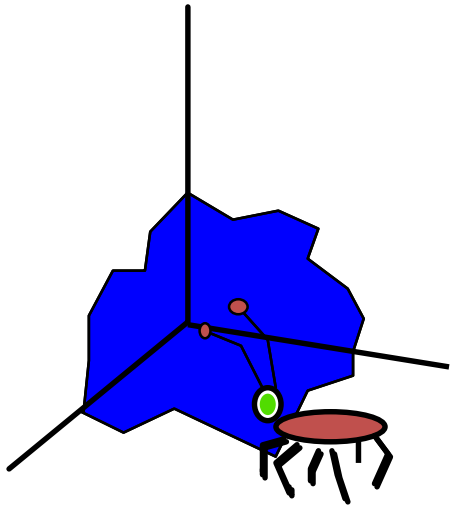
Góc nhìn trong và ngoài

- Mọi sản phẩm kĩ thuật nên được test bằng 1 trong 2 cách:
  - Nếu biết được hàm bất kì được thiết kế để làm gì, test có thể vừa chứng minh được hàm thực hiện đầy đủ công việc, vừa tìm được lỗi của mỗi hàm (công việc).
  - Nếu biết được chi tiết bên trong hoạt động của sản phẩm, tests có thể dẫn tới đảm bảo rằng "mọi thứ đã vào guồng".

Thiết kế Test Case

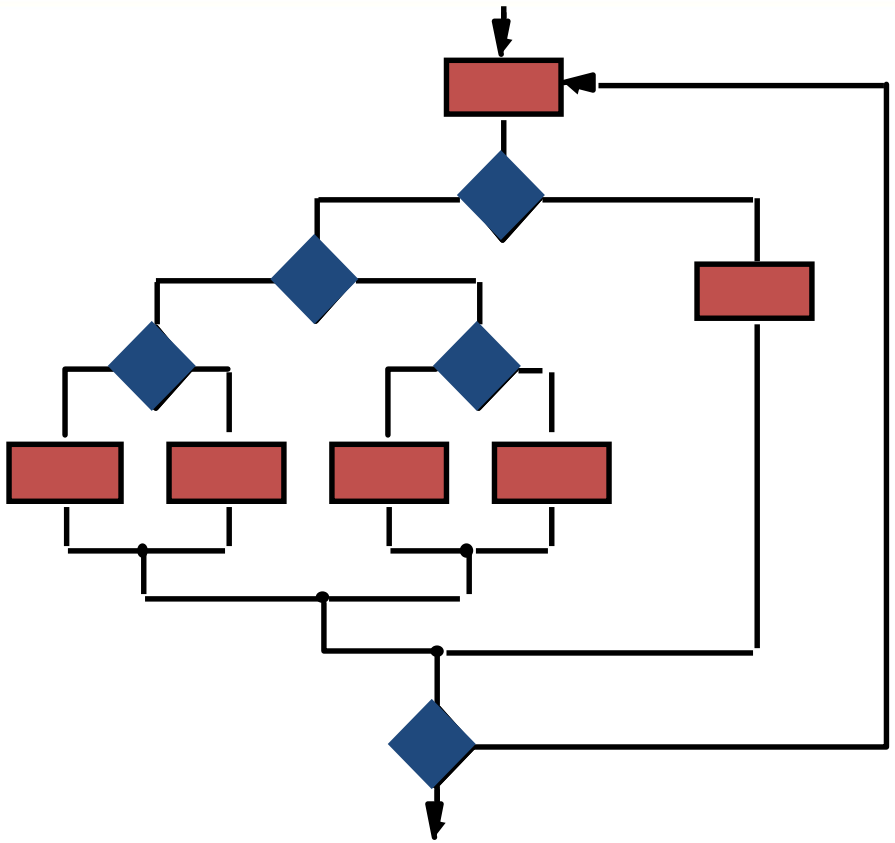
"Bugs (lỗi) ẩn náu tại góc
và hội tụ tại
các biên..."

Boris Beizer



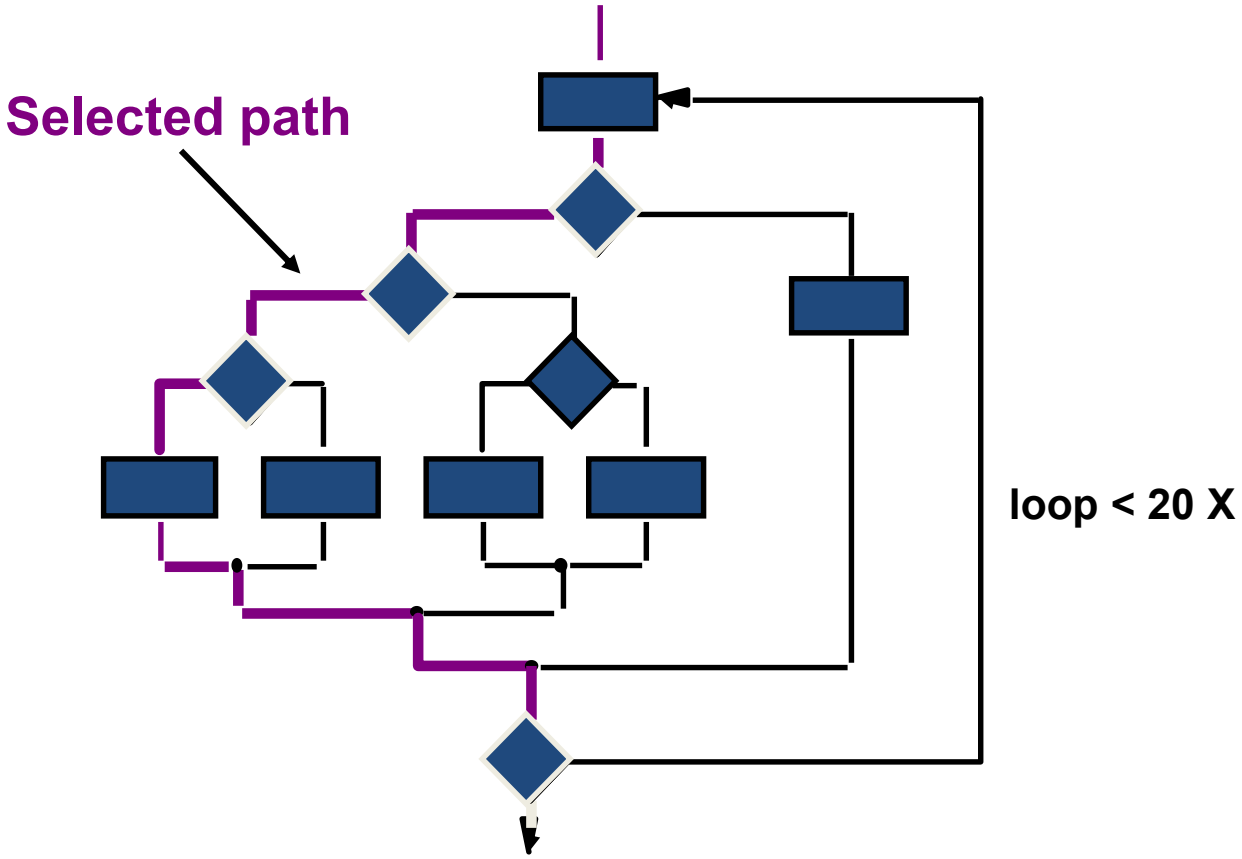
OBJECTIVE	**Để phát hiện lỗi**
CRITERIA	**với cách thức hoàn thiện**
CONSTRAINT	**với công sức và thời gian ít nhất**

Kiểm thử vét cạn

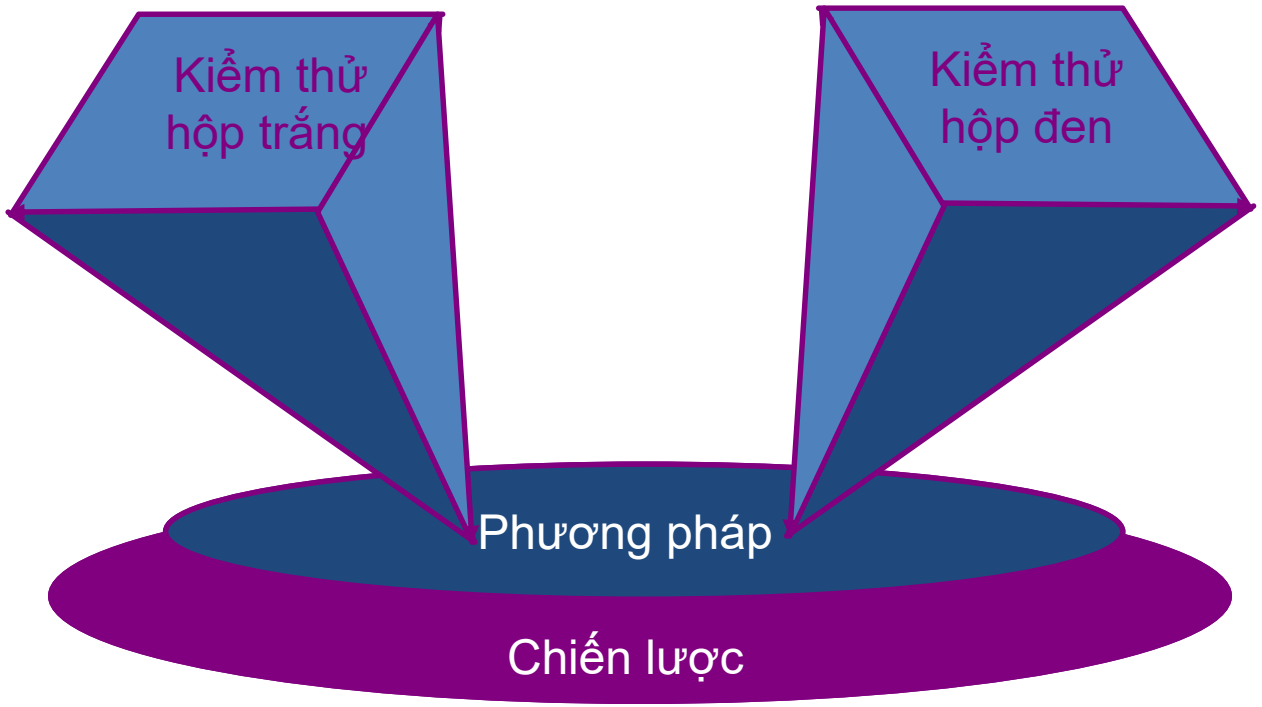


Có tất cả 10^{14} luồng thực thi! Nếu ta có thể thực hiện 1 test mỗi giây, nó có thể tốn 3170 năm để test chương trình

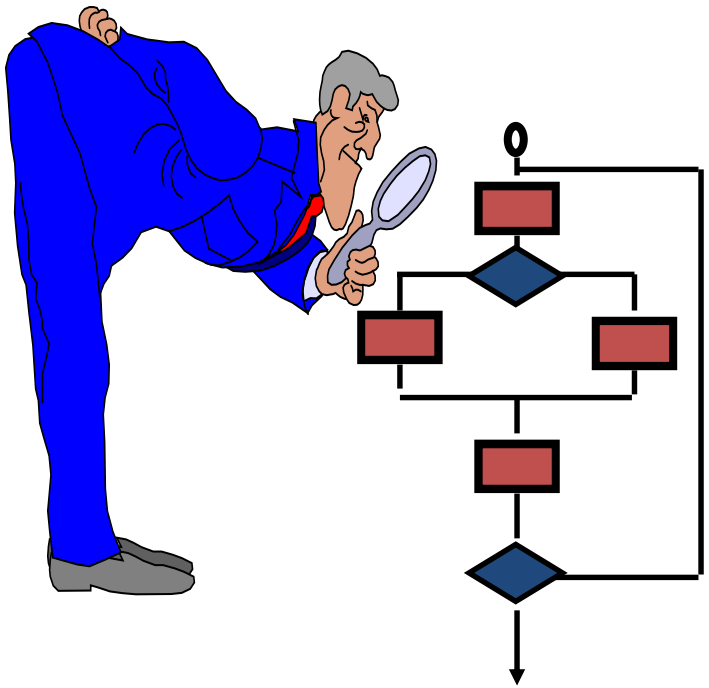
Kiểm thử chọn lựa



Kiểm thử phần mềm



Kiểm thử hộp trắng



... Mục đích để chắc chắn toàn bộ

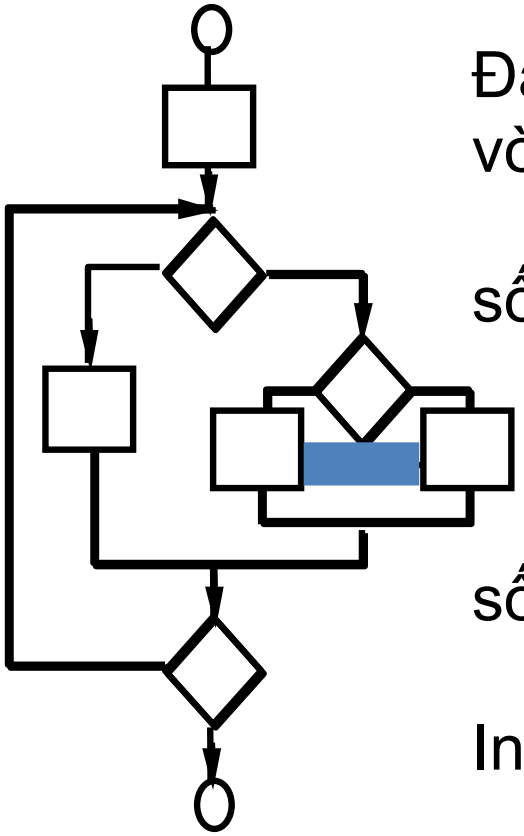
Câu lệnh và điều kiện đều được thực

Hiện ít nhất 1 lần...

Tại sao?

- Lỗi logic và các giả định sai là tỉ lệ nghịch với xác suất chương trình thực thi đi qua 1 đường chỉ định.
- Ta thường tin rằng 1 đường nào đó có thể không được thực hiện; nhưng thực tế đó là trực giác.
- Lỗi là ngẫu nhiên; nên có thể đường nào đó không được test có thể vẫn chứa lỗi.

Basis Path Testing



Đầu tiên, ta tính độ phức tạp vòng lặp:

số lượng các quyết định + 1

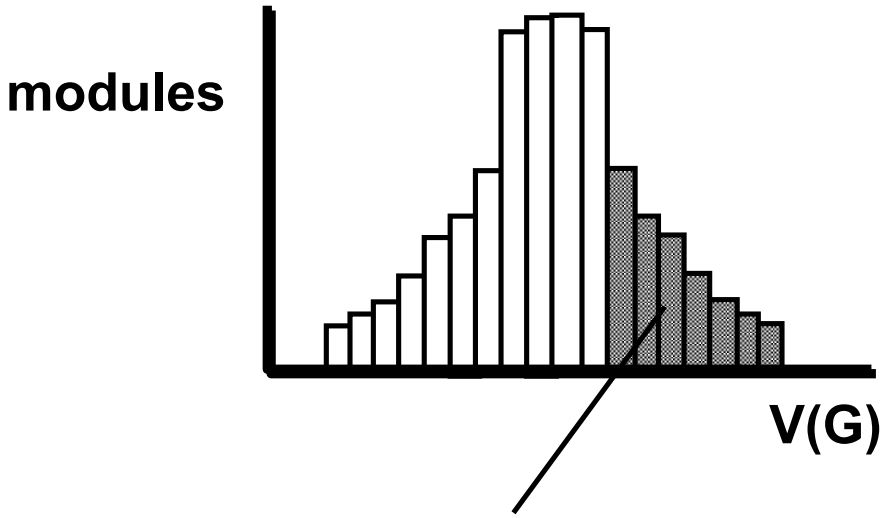
hoặc

số lượng các vị trí kết thúc + 1

In this case, V(G) = 4

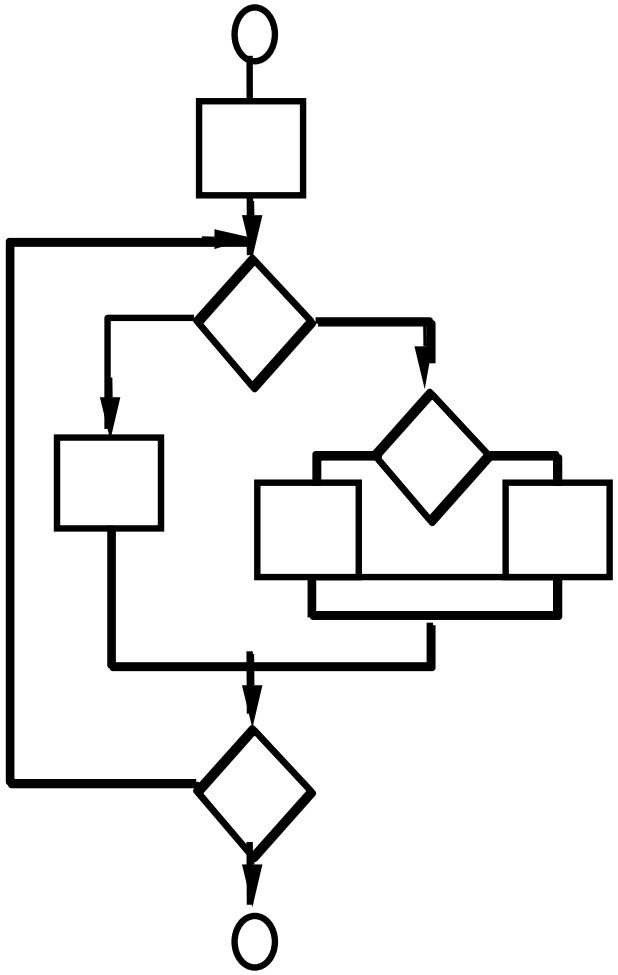
Cyclomatic Complexity

V(G) càng cao, khả năng lỗi càng lớn.



Các module trong vùng này
Có khả năng lỗi cao

Basis Path Testing



Tiếp theo, ta xét các independent paths:

V(G) = 4,
Nên có 4 đường:

Path 1: 1,2,3,6,7,8

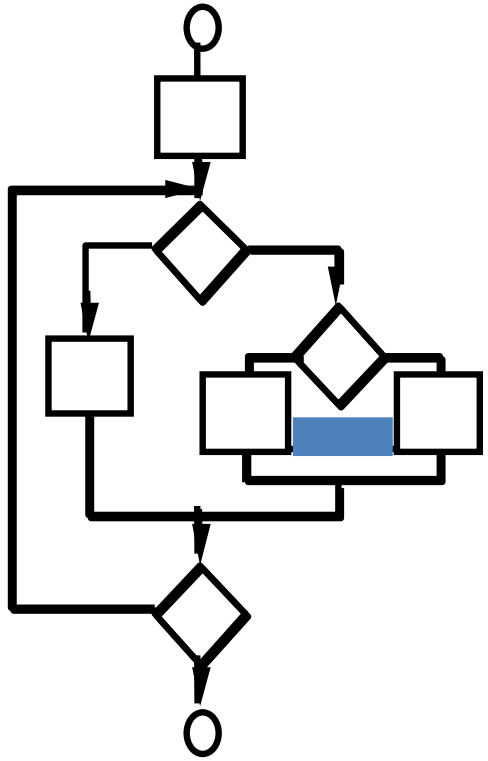
Path 2: 1,2,3,5,7,8

Path 3: 1,2,4,7,8

Path 4: 1,2,4,7,2,4,...7,8

Cuối cùng, ta tạo các test cases để kiểm tra các đường.

Basis Path Testing Notes



- Bạn không cần dùng biểu đồ, nhưng có hình vẽ sẽ giúp bạn tìm được các đường thực thi .
- Đếm mỗi test logic đơn giản, Ghép các tests count thành 2 hoặc nhiều hơn.
- basis path testing nên được áp dụng cho các module quan trọng.

Deriving Test Cases

- Tổng kết:
  - Sử dụng thiết kế hoặc mã nguồn để làm cơ sở vẽ đồ thị các đường thực thi.
  - Xác định cyclomatic complexity của đồ thị các luồng thực thi.
  - Xác định các đường độc lập cơ bản (independent paths).
  - Chuẩn bị các test case sao cho thực thi mỗi đường ít nhất 1 lần.

Ma trận đồ thị

- Ma trận đồ thị là ma trận vuông có kích thước bằng số các nút của đồ thị các đường thực thi.
- Mỗi dòng và cột biểu thị nút của đồ thị, mỗi ô của ma trận biểu thị cạnh giữa 2 nút trên đồ thị.
- Thêm trọng số vào mỗi ô trên ma trận thì ma trận đồ thị sẽ trở thành 1 công cụ đắc lực để quản lý cấu trúc chương trình trong quá trình kiểm thử.

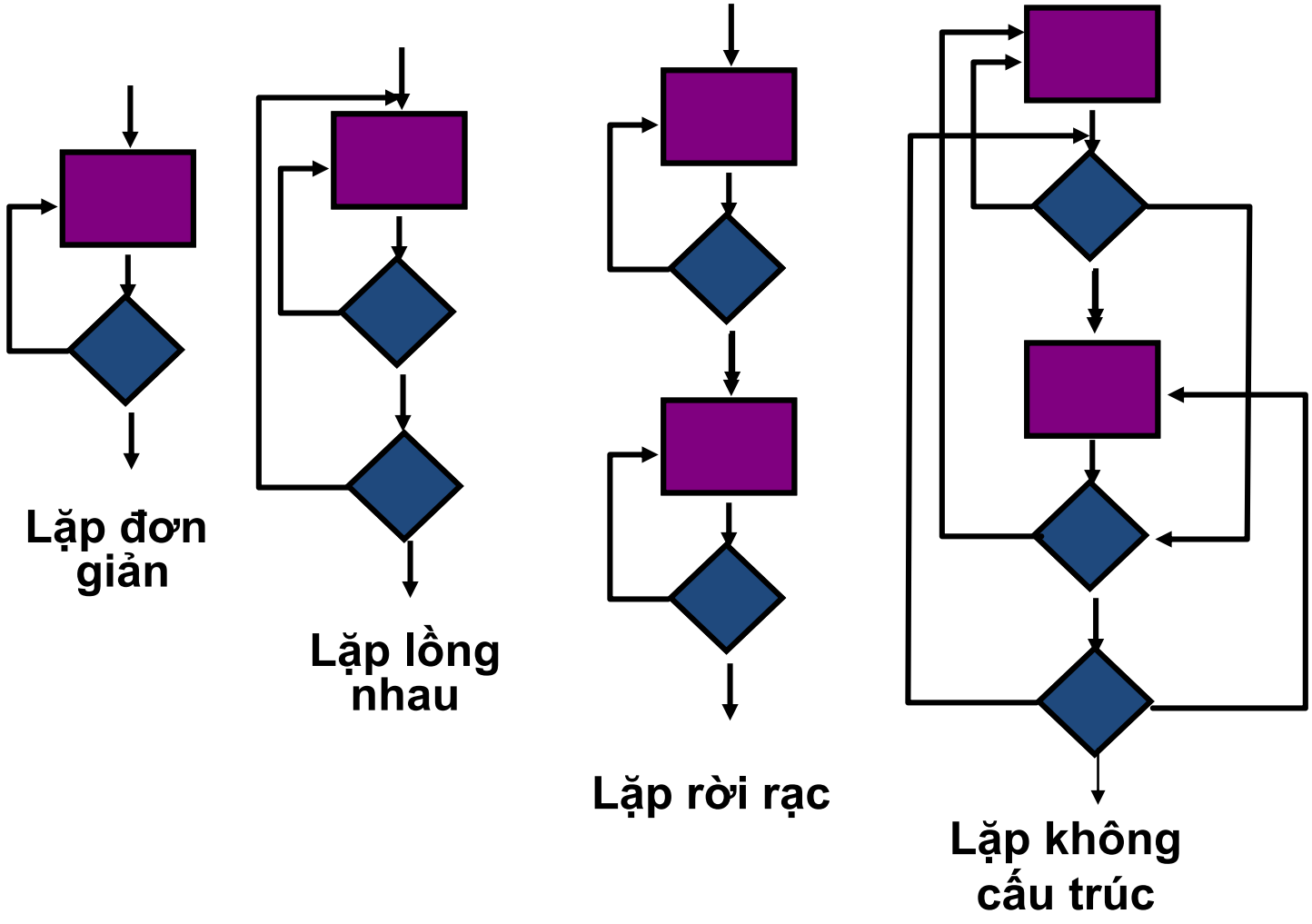
Control Structure Testing

- Kiểm thử có điều kiện— một test case thiết kế 1 phương thức kiểm tra các biểu thức điều kiện logic trong chương.
- Kiểm thử luồng dữ liệu— chọn đường (test paths) của chương trình dựa vào vị trí khai báo và sử dụng biến của chương trình.

Kiểm thử luồng dữ liệu

- Kiểm thử luồng dữ liệu chọn đường (test paths) của chương trình dựa vào vị trí khai báo và sử dụng biến của chương trình.
  - Giả sử mỗi câu lệnh cảu chương trình được đánh số thứ tự duy nhất và mỗi hàm không sửa đổi tham số của nó và biến toàn cục. Với mỗi câu lệnh với S là số thứ tự câu lệnh
    - DEF(S) = {X | câu lệnh S chứa khai báo của X}
    - USE(S) = {X | câu lệnh S có sử dụng X}
- Một definition-use (DU) chain của biến X có dạng [X, S, S'], với S và S' là các số thứ tự câu lệnh, X thuộc DEF(S) và USE(S'), và khai báo vủa X trong câu lệnh S được dùng ở câu lệnh S'.

Kiểm thử lặp



Kiểm thử lặp: Lặp đơn giản

Số điều kiện ít nhất—Lặp đơn

1. bỏ qua toàn bộ vòng lặp
2. chỉ 1 test case qua vòng lặp
3. 2 test case qua vòng lặp
4. m test case qua vòng lặp m<n
5. (n-1), n, và (n+1) qua vòng lặp

Với n là số test lớn nhất có thể qua vòng lặp.

Kiểm thử lặp: lặp lồng nhau

Lặp lồng nhau

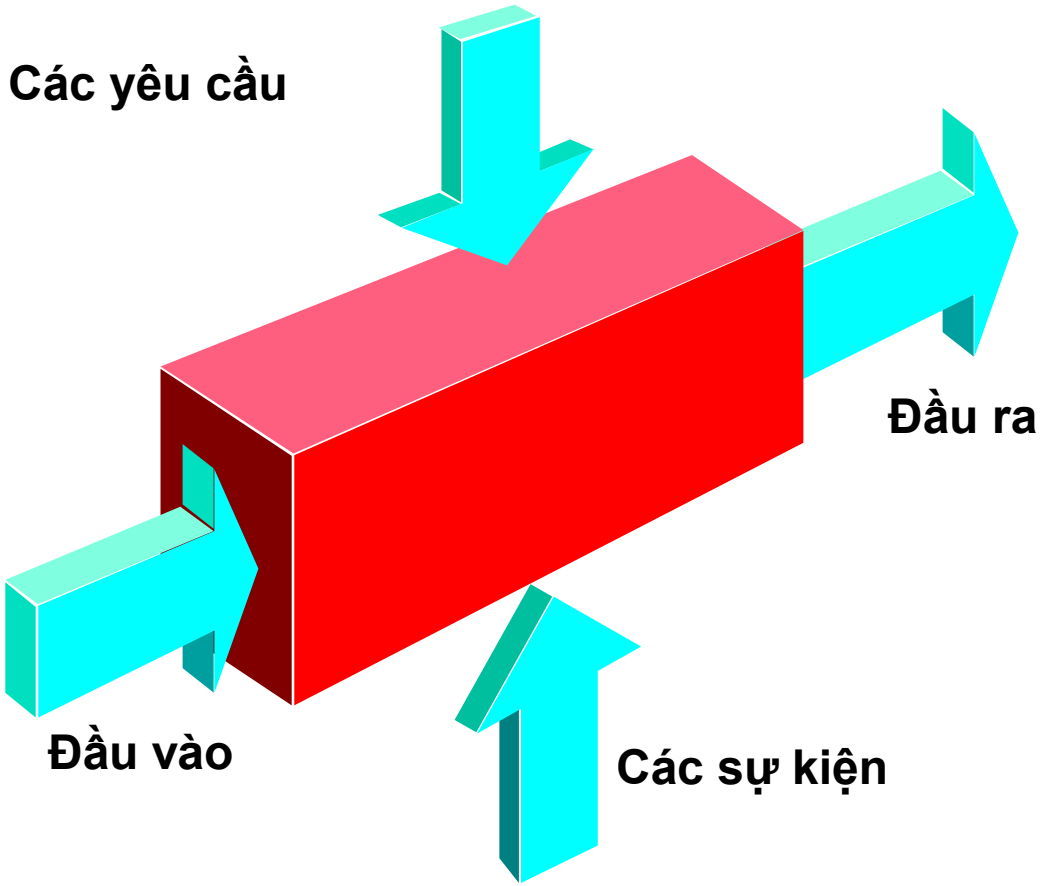
- Bắt đầu từ vòng lặp trong cùng. Đặt các tham số lặp của các vòng lặp ngoài là ít nhất.
- Kiểm thử min+1,…, max-1 và max cho vòng lặp trong cùng trong khi giữ các vòng lặp ngoài các tham số vòng lặp có giá trị nhỏ nhất.
- Tiếp đến vòng lặp trong ngoài tiếp theo, lặp lại bước 2 tương ứng với các vòng lặp ngoài của nó. Tiếp tục như vậy cho đến khi tất cả các vòng lặp đều được kiểm thử.

Lặp rời rạc

If 1 vài vòng lặp là độc lập với các vòng lặp khác
then coi chúng là các vòng lặp đơn
else coi chúng là các vòng lặp lồng nhau
endif*

- *Ví dụ , lần lặp cuối cùng của vòng lặp 1 là dùng để khởi tạo vòng lặp 2.*

Kiểm thử hộp đen



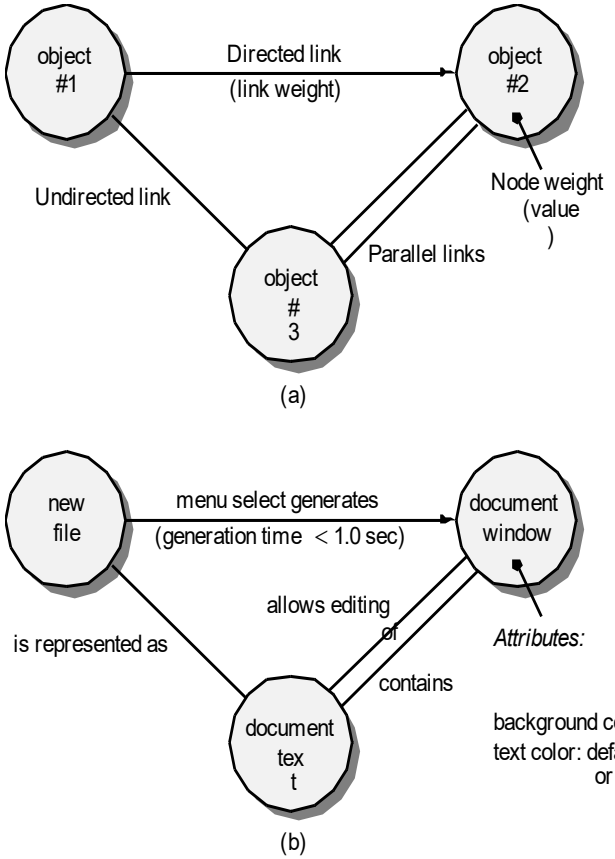
Kiểm thử hộp đen

- Làm cách nào kiểm thử hàm là hợp lệ?
- Kiểm thử hành vi của hệ thống?
- Những loại đầu vào nào là các test case tốt?
- Có phải hệ thống là "nhạy cảm" với một số dữ liệu đầu vào?
- Miền biên của dữ liệu sai khác như thế nào?
- Mức độ chịu lỗi (lưu lượng và khối lượng dữ liệu) của hệ thống?
- Tác nhân nào sẽ làm dữ liệu hội tụ trong khi hệ thống thực thi?

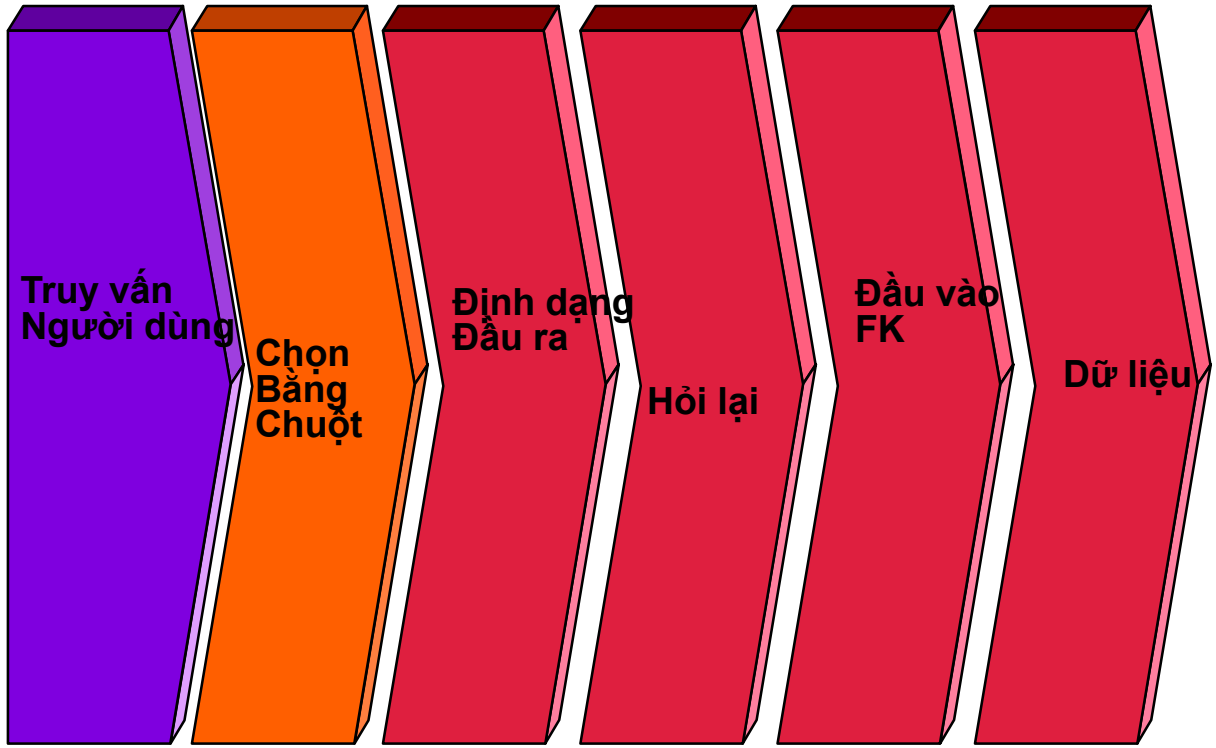
Phương pháp dựa vào đồ thị

Để hiểu các đối tượng định nghĩa trong chương trình và mối quan hệ của chúng.

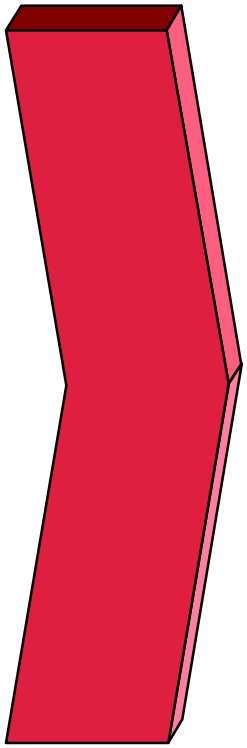
Ta để ý thuật ngữ "đối tượng" dùng trong rất nhiều ngữ cảnh. Nó chứa dữ liệu, các hàm, thành phần truyền thống (module), và các thành phần hướng đối tượng trong phần mềm máy tính.



Cách phân chia



Lấy mẫu các lớp phù hợp



Dữ liệu hợp lệ

Người dùng đưa ra các lệnh

Trả lợi các câu hỏi hệ thống (prompt)

tên tập tin

Dữ liệu vi tính

- **tham số vật lý**
- **giá trị biên**
- **giá trị khởi tạo**

Định dạng dữ liệu đầu ra

Xử lý các báo lỗi

Dữ liệ đồ họa(e.g., chọn bằng chuột)

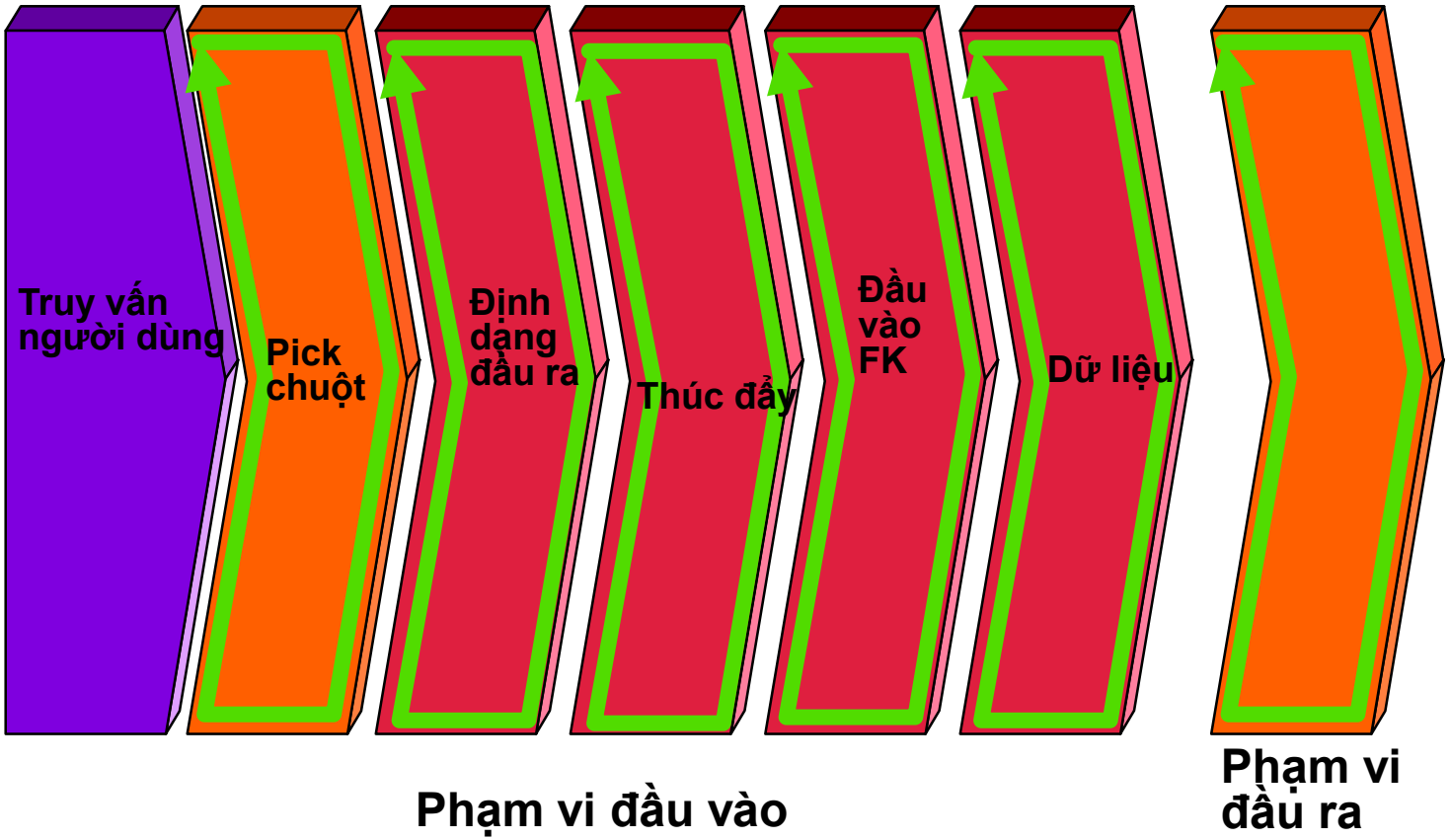
Dữ liệu không hợp lệ

Dữ liệu ngoài biên miền dữ liệu của chương trình.

Dữ liệu vật lý không thể có

Giá trị đúng được đưa ra bởi nguồn không hợp lệ.

Ranh giới phân tích giá trị

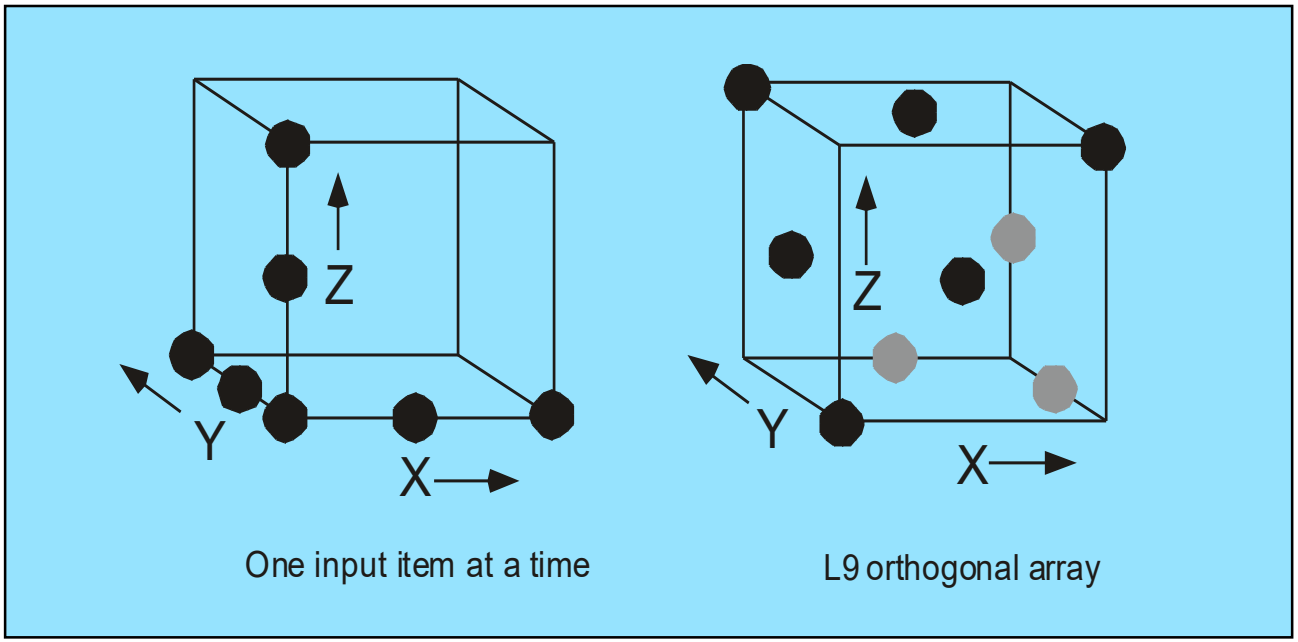


Kiểm thử so sánh

- Chỉ được sử dụng trong các tình huống mà trong đó độ tin cậy của phần mềm là vô cùng quan trọng (ví dụ, hệ thống đánh giá con người)
  - Các đội ngũ kỹ sư phần mềm riêng biệt phát triển các phiên bản độc lập của ứng dụng bằng cách sử dụng các đặc điểm kỹ thuật giống nhau
  - Mỗi phiên bản có thể được thử nghiệm với các dữ liệu thử nghiệm tương đồng để đảm bảo rằng tất cả các đầu ra giống hệt nhau
  - Sau đó, tất cả các phiên bản đều được thực hiện song song với thời gian thực và so sánh kết quả để đảm bảo tính nhất quán

Kiểm thử mảng trực giao

- Được sử dụng khi số lượng các tham số đầu vào là nhỏ và các giá trị mà mỗi tham số có thể nhận bị chặn



Kiểm thử dựa trên mô hình

- Phân tích một mô hình hành vi hiện hành cho các phần mềm hoặc tạo ra mô hình này
  - Nhớ rằng một mô hình hành vi chỉ ra cách mà phần mềm sẽ đáp ứng lại với các sự kiện bên ngoài hoặc các kích thích.
- Duyệt qua các mô hình hành vi và xác định các yếu tố đầu vào mà sẽ buộc các phần mềm chuyển đổi giữa các trạng thái
  - Các yếu tố đầu vào sẽ kích hoạt sự kiện cái sẽ gây ra quá trình chuyển đổi giữa các trạng thái xảy ra.
- Xem lại các mô hình hành vi và lưu ý các đầu ra mong đợi khi các phần mềm chuyển đổi giữa các trạng thái
  - Thực hiện các trường hợp kiểm thử.
- So sánh kết quả thực tế và dự kiến và đưa ra hành động khắc phục theo yêu cầu.

Kiểm thử phần mềm mẫu

- Kiểm thử mẫu được thực hiện theo cách tương tự như các mẫu thiết kế (Chương 12).
- Ví dụ:
  - Tên mẫu: ScenarioTesting
  - Abstract: Sau khi việc kiểm tra đơn vị và hợp nhất được thực hiện, cần xác định xem phần mềm sẽ thực hiện có thỏa mãn người dùng. Các mô hình ScenarioTesting mô tả một kỹ thuật để thực hiện các phần mềm dựa trên quan điểm của người sử dụng. Thất bại ở bước này chỉ ra rằng phần mềm đã thất bại trong việc đáp ứng yêu cầu của người sử dụng.

Tài liệu tham khảo

- Slide đi kèm với Software Engineering: A Practitioner’s Approach, 7/e by Roger S. Pressman
- Slides copyright © 1996, 2001, 2005, 2009 by Roger S. Pressman
- Chỉ dùng cho mục đích giáo dục phi lợi nhuận.
- Có thể sửa đổi slide chỉ nhằm mục đích phục vụ sinh viên đại học trong những môn học liên quan tới sách Software Engineering: A Practitioner's Approach, 7/e. Nghiêm cấm mọi hoạt động sửa đổi khác hoặc sử dụng không được sự cho phép của tác giả.
- Mọi thông tin bản quyền phải được đi kèm nếu những slide này được đăng lên mạng để phục vụ sinh viên.