

Kiểm thử phần mềm

Khoa CNTT

ĐH Công nghiệp Hà Nội

Nội dung

- Bài 1. Tổng quan về kiểm thử phần mềm
- Bài 2. Quy trình kiểm thử phần mềm
- Bài 3. Các cấp độ kiểm thử
- Bài 4. Các loại hình kiểm thử
- Bài 5. Các kỹ thuật kiểm thử
- Bài 6. Kiểm thử tự động

Bài 1. Tổng quan về kiểm thử phần mềm

- 1.1. Phần mềm và chất lượng phần mềm, SQA
- 1.2. Các yếu tố ảnh hưởng đến chất lượng pm
- 1.3. Khái niệm kiểm thử
- 1.4. Mục tiêu của kiểm thử
- 1.5. Tầm quan trọng của kiểm thử
- 1.6. Các nguyên tắc trong kiểm thử
- 1.7. Một số khái niệm liên quan
- 1.8. Các đối tượng thực hiện kiểm thử
- 1.9. Các điểm cần lưu ý khi kiểm thử
- 1.10. Các hạn chế của kiểm thử

1.1 Phần mềm và chất lượng phần mềm

- Phần mềm và các đặc trưng
- Các khái niệm về lỗi, sai sót, hỏng hóc
- Nguyên nhân gây ra lỗi phần mềm
- Chất lượng phần mềm
- Đảm bảo chất lượng phần mềm

1.1.1 Phần mềm

- **Theo định nghĩa của IEEE:** Bao gồm các chương trình máy tính, các thủ tục, các tài liệu có thể liên quan và các dữ liệu liên quan đến hoạt động của hệ thống máy tính
- **Theo định nghĩa của ISO:** 4 thành phần cơ bản của phần mềm:
 - Chương trình máy tính (code)
 - Các thủ tục
 - Tài liệu
 - Dữ liệu cần thiết để vận hành phần mềm

1.1.1 Phần mềm

- **Đặc trưng của phần mềm:**

- Phần mềm được kỹ nghệ, không được chế tạo theo nghĩa cổ điển:
 - Phần mềm được thiết kế, chế tạo như các loại sản phẩm công nghiệp khác, nhưng **không được định hình trước**
 - Quá trình phát triển phần mềm quyết định giá thành và chất lượng của nó
 - Các phần mềm chỉ thực sự được tìm ra lỗi trong pha phát triển.

1.1.1 Phần mềm

- **Đặc trưng của phần mềm:**

- *Có tính phức tạp cao và luôn thay đổi.*
- Phần mềm là một hệ thống logic với nhiều khái niệm và các mối liên hệ logic khác nhau => mỗi một vòng lặp với một giá trị khác nhau là cơ hội để tìm ra lỗi của phần mềm
- Thay đổi theo nhu cầu của người dùng
- Thay đổi để đáp ứng môi trường vận hành
- Phần mềm không nhìn thấy được
 - Phần mềm không nhìn thấy được mà chỉ có thể nhận biết qua sự mô tả từ những khía cạnh khác nhau(sơ đồ điều khiển, mô hình luồng dữ liệu, mô hình tương tác...)
 - Do đặc trưng này nên khả năng tìm ra lỗi một cách nhanh chóng là không thể

1.1.2 Khái niệm lỗi, sai sót, hỏng

- **Lỗi phần mềm (software error)**

- Là lỗi do con người gây ra (thường là các lập trình viên)
- Lỗi phần mềm có thể là lỗi cú pháp hoặc lỗi logic

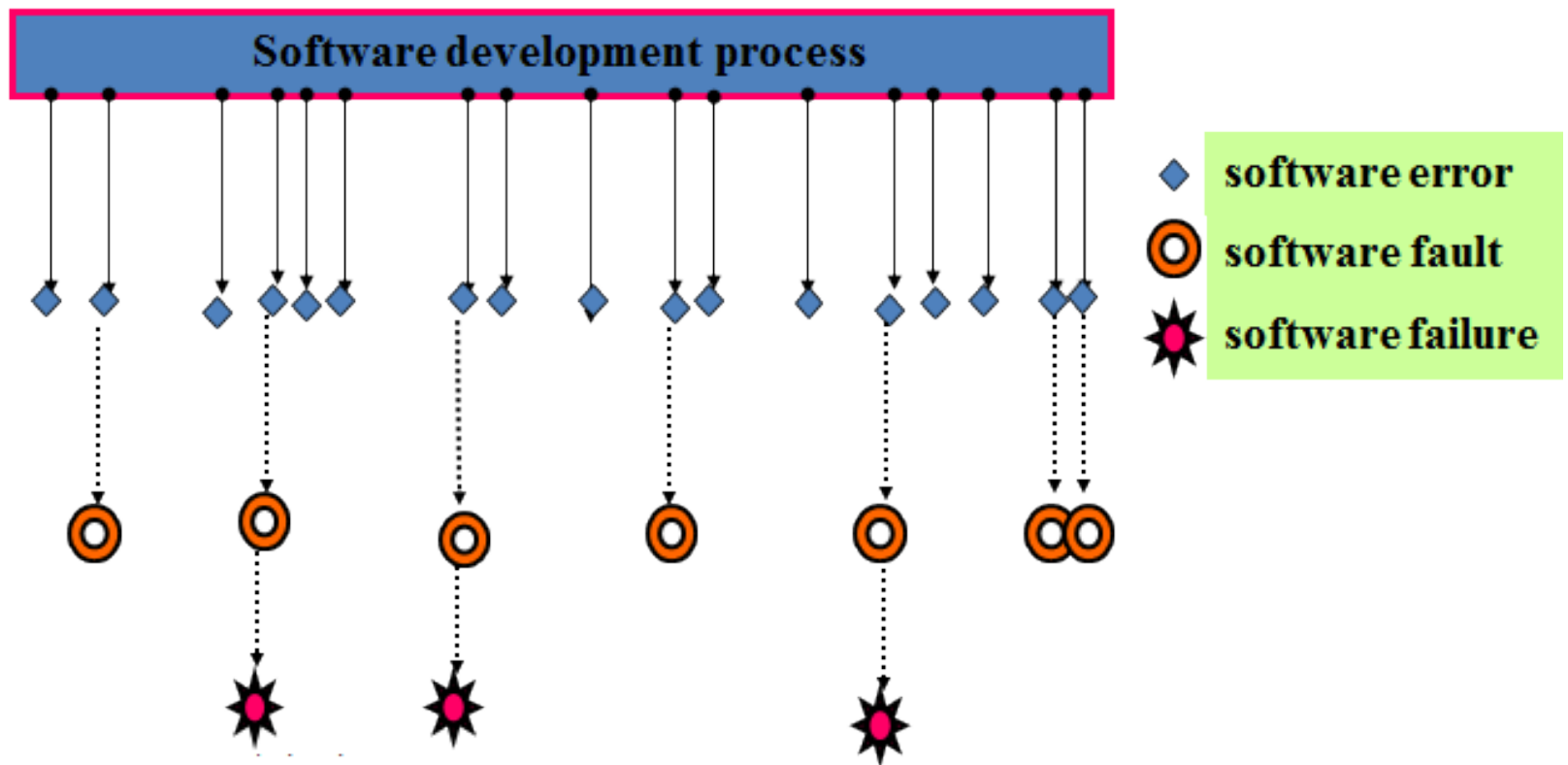
- **Sai sót của phần mềm (software fault)**

- Sai sót của phần mềm không phải lúc nào cũng do lỗi phần mềm
- Có thể có sai sót do dư thừa hoặc bỏ sót yêu cầu phần mềm (từ khâu khảo sát, phân tích, đưa ra yêu cầu phần mềm bị thừa hoặc bị sót so với yêu cầu của khách hàng)

- **Hỏng hóc của phần mềm (software failure)**

- Một sai sót của phần mềm dẫn đến hỏng hóc khi nó sai sót đó bị phát hiện
- Một sai sót của phần mềm nếu không bị phát hiện hoặc ko gây ảnh hưởng tới phần mềm thì sẽ không được coi là hỏng hóc của pm

1.1.2 Khái niệm lỗi, sai sót, hỏng



1.1.3 Các nguyên nhân gây ra lỗi phần mềm

1. Định nghĩa sai yêu cầu của khách hàng

- Đây được coi là gốc rễ của việc gây ra lỗi phần mềm
- Hiểu sai yêu cầu của khách hàng
- Yêu cầu của khách hàng không được làm rõ
- Triển khai phần mềm thiếu yêu cầu của khách hàng
- Khách hàng đưa ra quá nhiều yêu cầu không cần thiết và không liên quan



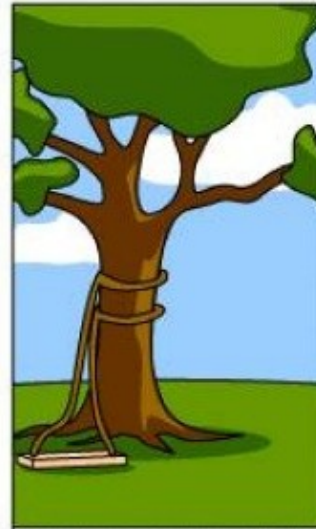
How the customer explained it



How the Project Leader understood it



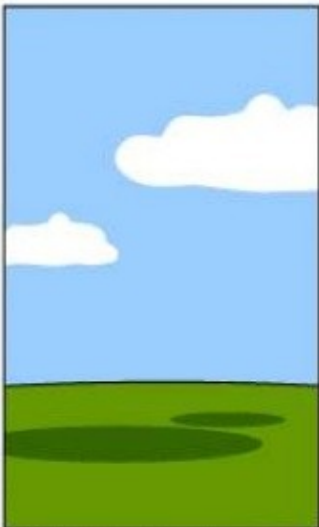
How the Analyst designed it



How the Programmer wrote it



How the Business Consultant described it



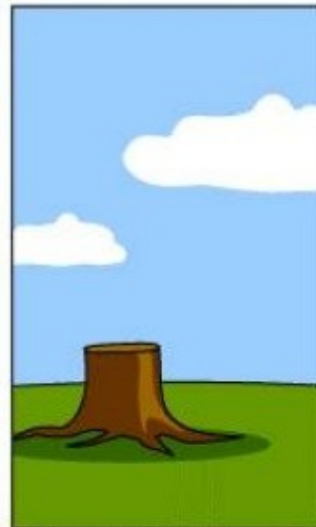
How the project was documented



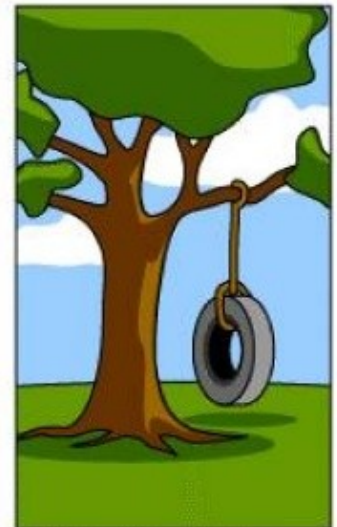
What operations installed



How the customer was billed



How it was supported



What the customer really needed

1.1.3 Các nguyên nhân gây ra lỗi phần mềm

2. Thất bại trong việc giao tiếp giữa người phát triển và khách hàng

- Có sự không hiểu cấu trúc của tài liệu yêu cầu phần mềm
- Không nắm bắt được những thay đổi được viết trong tài liệu yêu cầu
- Những thay đổi được yêu cầu từ khách hàng nhưng không được lưu dưới dạng văn bản
- Thiếu sự chú ý tới:
 - Thông điệp của khách hàng đề cập tới việc thay đổi yêu cầu
 - Trả lời của khách hàng tới những câu hỏi mà developer đặt ra

1.1.3 Các nguyên nhân gây ra lỗi phần mềm

3. Tạo ra độ lệch cố ý trong yêu cầu phần mềm

- Lập trình viên sử dụng những module phần mềm có sẵn từ những dự án trước mà không thay đổi cho phù hợp với yêu cầu của dự án mới nhằm tiết kiệm thời gian
- Bỏ qua một vài yêu cầu của phần mềm do thời gian quá gấp hoặc chi phí không đủ đáp ứng.

1.1.3 Các nguyên nhân gây ra lỗi phần mềm

4. Lỗi logic trong thiết kế phần mềm

- Thiết kế sai thuật toán
- Ghi nhận sai sự tuần tự
- Ghi nhận sai các điều kiện biên
- Ghi nhận sai các trạng thái của hệ thống
- Ghi nhận sai các phản ứng khi gặp các hoạt động không đúng yêu cầu của hệ thống

1.1.3 Các nguyên nhân gây ra lỗi phần mềm

5. Lỗi mã hóa

- Lỗi logic
- Lỗi cú pháp
- Lỗi thời gian chạy

6. Không tuân theo các tài liệu và cấu trúc code

- Không tuân theo các chuẩn tài liệu (templates...)
- Không tuân theo các cấu trúc mã hóa

7. Rút ngắn quá trình kiểm thử phần mềm

- Do áp lực về thời gian, tiến độ hoàn thành dự án
- Lập kế hoạch kiểm thử không đầy đủ
- Không báo cáo đầy đủ các lỗi
- Báo cáo không chính xác lỗi

1.1.3 Các nguyên nhân gây ra lỗi phần mềm

8. Lỗi thủ tục

Chỉ dẫn cho người dùng những hoạt động cần thiết ở một quá trình. Nó quan trọng trong các hệ thống pm phức tạp khi quá trình xử lý được thực hiện qua nhiều bước. Mỗi bước có nhiều dạng dữ liệu và cho phép kiểm tra kết quả trung gian

9. Lỗi tài liệu

- Sai sót trong hồ sơ thiết kế
- Sai sót trong việc lập tài liệu hướng dẫn sử dụng
- Các danh sách chức năng không có trong phần mềm nhưng lại có trong tài liệu

1.1.4 Chất lượng phần mềm – quan điểm

- **Theo quan điểm của người dùng:** sản phẩm phù hợp với mục đích sử dụng của người dùng
- **Theo quan điểm của nhà cung cấp sản phẩm:** sản phẩm đạt được các tiêu chí đánh giá do nhà cung cấp đề ra
- **Theo quan điểm của nhà sản xuất phần mềm:** sản phẩm đáp ứng đầy đủ các tiêu chí đề ra trong bản đặc tả.

1.1.4 Chất lượng phần mềm – Khái niệm

- **Định nghĩa của IEEE:**
- Chất lượng phần mềm là:
 - Mức độ mà một hệ thống, thành phần hoặc quá trình đáp ứng yêu cầu quy định
 - Mức độ và một hệ thống, thành phần hoặc quá trình đáp ứng nhu cầu của người sử dụng hoặc mong đợi của khách hàng.
- **Theo cách tiếp cận của ISO:**
- Chất lượng toàn diện của phần mềm cần phải được quan tâm từ:
 - Chất lượng quy trình
 - Chất lượng phần mềm nội bộ (chất lượng trong)
 - Chất lượng phần mềm đối chiếu với yêu cầu người dùng(chất lượng ngoài)
 - Chất lượng phần mềm trong sử dụng (chất lượng sử dụng)

1.1.5 Đảm bảo chất lượng phần mềm

- Đảm bảo chất lượng phần mềm:
- Thiết lập một **tập hợp** các hoạt động có chủ đích và có hệ thống nhằm mang lại sự tin tưởng sẽ đạt được chất lượng đúng theo yêu cầu.
 - Đảm bảo dự án phần mềm sẽ hoàn thành đúng đặc tả, theo chuẩn mực định trước và các chức năng đòi hỏi, không có hỏng hóc và các vấn đề tiềm ẩn.
 - Điều khiển và cải tiến tiến trình phát triển phần mềm ngay từ khi dự án bắt đầu. Nó có tác dụng “phòng ngừa” cái xấu, cái kém chất lượng.
 - Mục tiêu: thỏa mãn khách hàng (Thời gian+Ngân sách+Chất lượng)

Tester vs QA

- KS kiểm định (Tester) có nhiệm vụ khảo sát, chạy thử để bảo đảm PM thỏa mãn các yêu cầu về chức năng và khả năng vận hành mà nó phải có, báo cáo các lỗi nếu có để các bộ phận liên quan chỉnh sửa. Công việc của KS kiểm định liên quan đến sản phẩm (product).
- KS chất lượng (QA) có nhiệm vụ giám sát để bảo đảm các tiêu chuẩn và quy trình sản xuất PM được định nghĩa và tuân thủ nghiêm túc, hướng đến mục tiêu các sản phẩm (SP) trung gian cũng như SP sau cùng của dự án thỏa mãn các tiêu chuẩn và yêu cầu đã định trước đó. Công việc của KS chất lượng liên quan đến quy trình (process).
- Ví dụ: Kiểm tra để bảo đảm các giải thuật khi viết code phải được chú thích rõ ràng, các Yêu cầu khách hàng được xem xét cẩn thận và mọi người hiểu giống nhau, các tài liệu đi kèm SP được kiểm tra trước khi gửi cho khách hàng

1.2. Các yếu tố ảnh hưởng đến chất lượng phần mềm

- Có ba yếu tố ảnh hưởng tới chất lượng phần mềm (tam giác chất lượng)
 - ◆ Con người
 - ◆ Quy trình
 - ◆ Công cụ

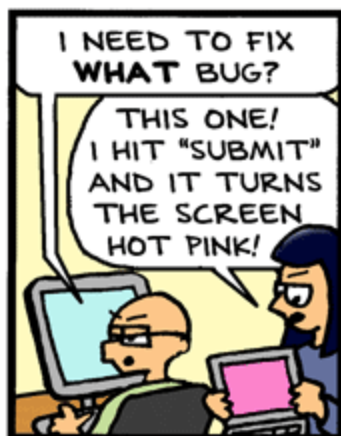


1.2. (tiếp)



1.2. (tiếp)

- Khoảng cách giữa yêu cầu người dùng và bản đặc tả yêu cầu hệ thống:
 - Không hiểu rõ yêu cầu của người dùng
 - Bỏ qua yêu cầu
 - Thiếu yêu cầu
 - Không đồng bộ về các phiên bản của tài liệu yêu cầu người dùng và tài liệu đặc tả
 - Bản đặc tả có thêm những yêu cầu không xuất phát từ người dùng



Bug Bash by Hans Bjordahl



<http://www.bugbash.net/>

1. 2. (tiếp)

- Khoảng cách giữa bản đặc tả và sản phẩm:
 - Hiểu sai yêu cầu đặc tả do trong bản đặc tả có những chỗ diễn đạt chưa rõ ràng cụ thể.
 - Có các yêu cầu được đưa thêm vào trong quá trình phát triển nhưng không được thêm vào bản đặc tả.
 - Có sự thay đổi yêu cầu trong quá trình phát triển nhưng không được cập nhật vào bản đặc tả
 - Các tính năng mới được thêm vào bởi mục đích riêng của người phát triển
 - Các yêu cầu có trong bản đặc tả nhưng bị bỏ qua do quá khó để thực hiện

1.2. (tiếp)

- Khoảng cách giữa yêu cầu người dùng và sản phẩm:
 - Khoảng cách này xuất hiện do sản phẩm làm ra không thỏa mãn yêu cầu người dùng
 - Độ lệch này phụ thuộc vào hai cạnh còn lại của tam giác chất lượng
 - Đây là độ lệch gây tổn kém nhất để sửa chữa

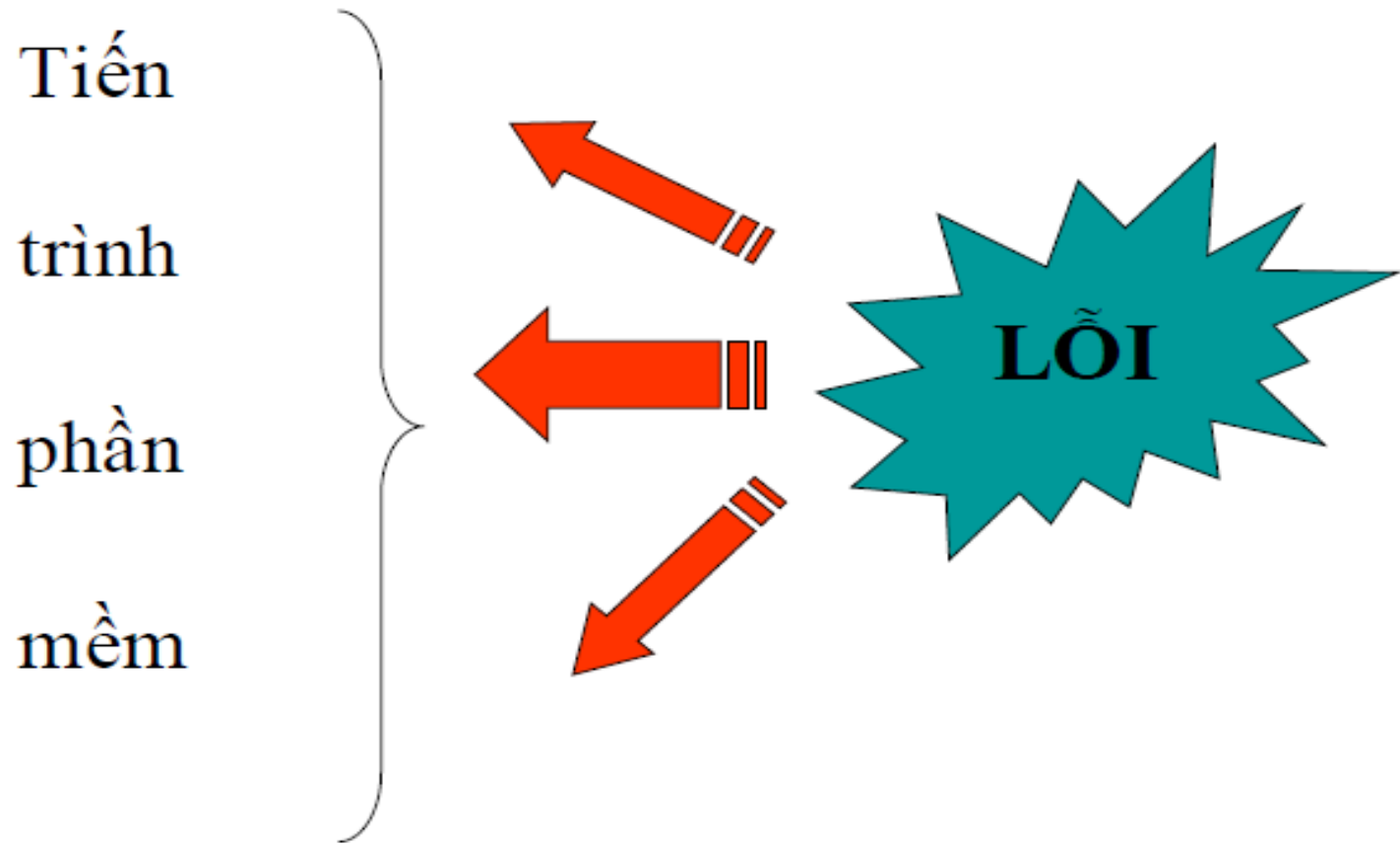
1.3. Khái niệm kiểm thử

- Theo Glenford Myers:
 - Kiểm thử là quá trình vận hành chương trình để tìm ra lỗi
- Theo IEEE: Kiểm thử là
 - (1) Là quá trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó.
 - (2) Là quá trình phân tích phần mềm để tìm ra sự khác biệt giữa điều kiện thực tế và điều kiện yêu cầu và dựa vào điểm khác biệt đó để đánh giá tính năng phần mềm

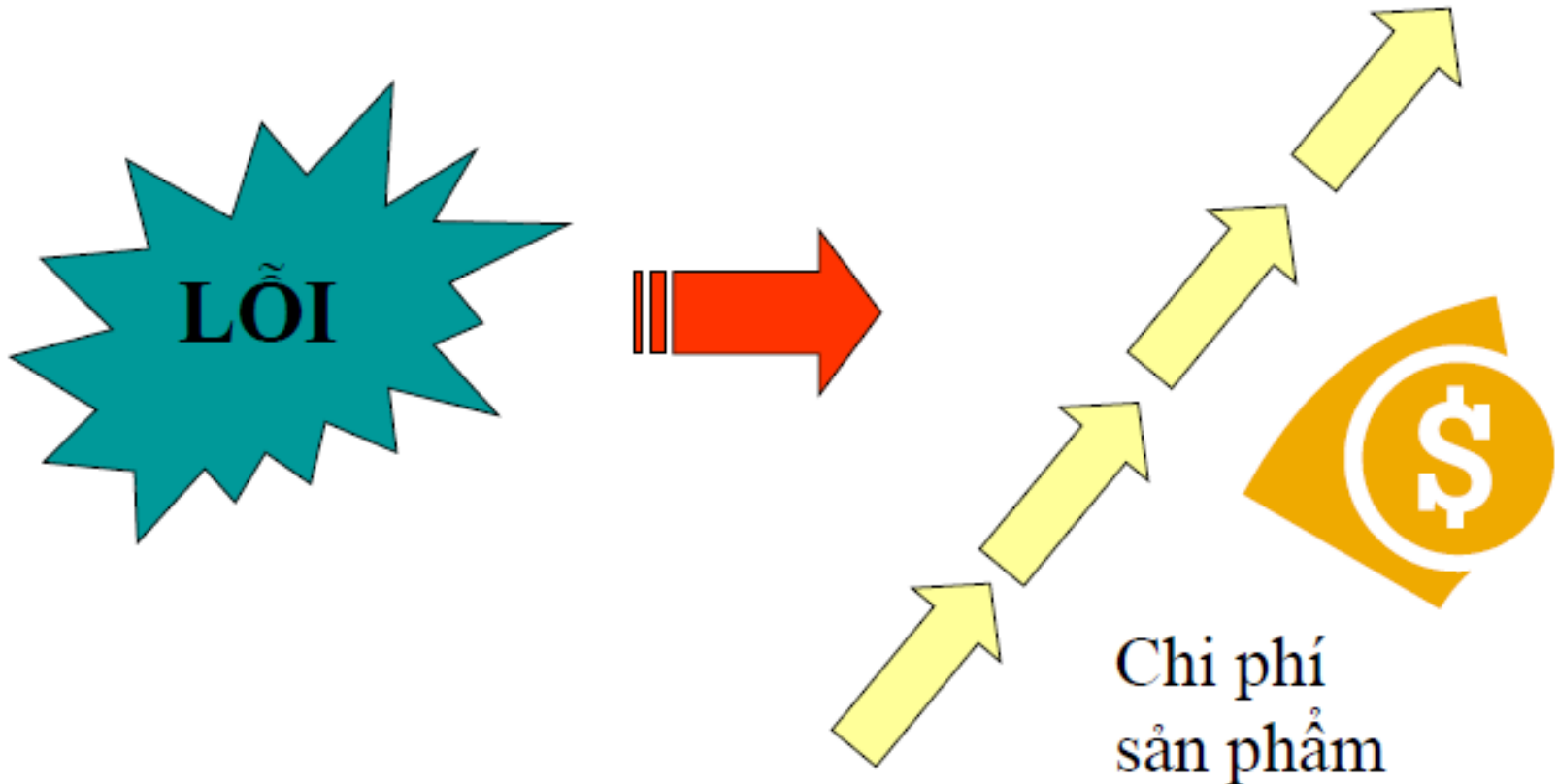
1.4. Mục tiêu của kiểm thử

- Tìm ra được càng nhiều lỗi càng tốt trong điều kiện về thời gian đã định và nguồn lực sẵn có
- Chứng minh rằng sản phẩm phần mềm phù hợp với các đặc tả của nó.
- Xác thực chất lượng kiểm thử phần mềm đã dùng chi phí và nỗ lực tối thiểu
- Thiết kế tài liệu kiểm thử một cách có hệ thống và thực hiện nó sao cho có hiệu quả, tiết kiệm được thời gian công sức.

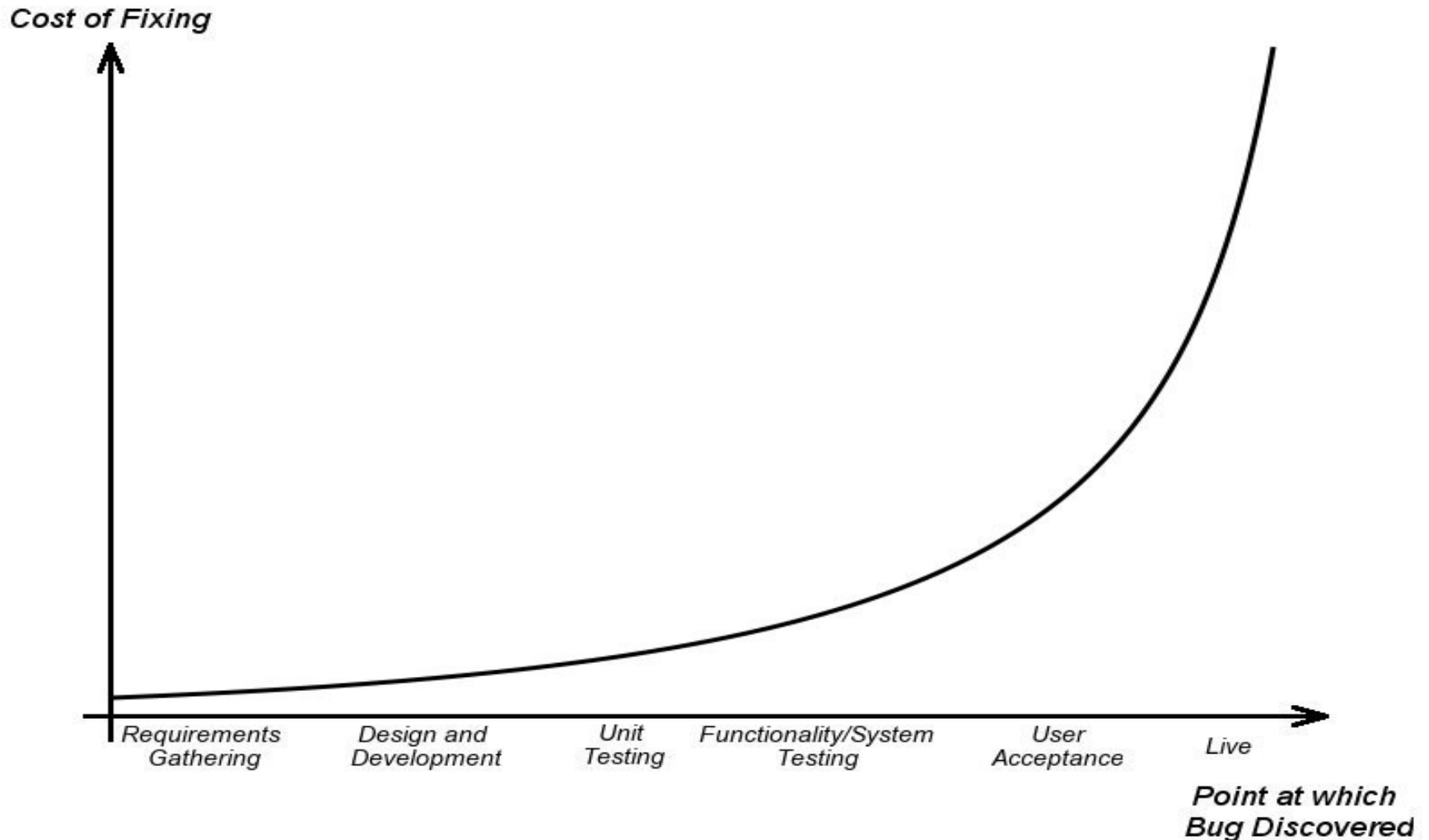
1.5. Tầm quan trọng của kiểm thử



1.5. Tầm quan trọng của kiểm thử



1.5. Tầm quan trọng của kiểm thử



1.5. Tầm quan trọng của kiểm thử

- Những người phát triển phần mềm cho rằng:
 - Kiểm thử chỉ để chứng minh chương trình không có lỗi
 - Mục đích của kiểm thử là chỉ ra rằng chương trình đã thực hiện đúng các chức năng đã đưa ra.
 - Kiểm thử là quy trình thực hiện để chứng tỏ chương trình đã làm được các chức năng cần có.
- Những ý kiến trên về kiểm thử đã đầy đủ?
 - Kiểm thử còn để tìm ra lỗi và sửa chữa các lỗi đó nhằm tăng độ tin cậy cho phần mềm.

1.5. Tầm quan trọng của kiểm thử

- Tại sao cần thực hiện kiểm thử?
 - Để xem xét chất lượng sản phẩm
 - Để phát hiện ra lỗi
- Ví dụ:
- Khách hàng có thể rút tiền ở máy ATM với số tiền tối đa là 250\$/1 giao dịch
- Người kiểm thử 1:
 - Thử 3 lần với 3 yêu cầu: 50\$, 150\$, 250\$ thấy máy đều nhả ra số tiền chính xác, kết luận chức năng rút tiền hoạt động đúng yêu cầu của khách hàng là yêu cầu rút ra bao nhiêu đều trả về đúng bấy nhiêu tiền.
- Người kiểm thử 2:
 - Yêu cầu số tiền là 300\$, máy vẫn nhả ra đúng 300\$ mà ko đưa ra thông báo số tiền rút bị quá hạn, như vậy là có lỗi mà người kiểm thử 1 ko tìm ra được.

1.6. Các nguyên tắc trong kiểm thử

- Trong kiểm thử có 7 nguyên tắc cơ bản:
 1. Kiểm thử chỉ ra sự hiện diện của lỗi trong phần mềm
 2. Kiểm thử tất cả các trường hợp là điều không thể
 3. Nên thực hiện kiểm thử càng sớm càng tốt
 4. Sự phân cụm của các lỗi
 5. Nghịch lý thuốc trừ sâu
 6. Kiểm thử theo các ngữ cảnh độc lập
 7. Sự sai lầm về việc không có lỗi

1.7. Phân loại kiểm thử

- Phân loại kiểm thử dựa trên các yếu tố:
 - Mục đích kiểm thử
 - Chiến lược kiểm thử
 - Phương pháp kiểm thử
 - Kỹ thuật kiểm thử

1.7.1. Dựa vào mục đích kiểm thử

- Kiểm thử đơn vị, module
- Kiểm thử cấu hình
- Kiểm thử sơ lược (smoke testing)
- Kiểm thử chức năng
- Kiểm thử tích hợp
- Kiểm thử hồi quy
- Kiểm thử hệ thống
- Kiểm thử tải dữ liệu (load testing)
- Kiểm thử tải trọng (stress testing)
- Kiểm thử hiệu suất (performance testing)
- Kiểm thử chấp nhận (UAT)
- Kiểm thử bảo mật (security testing)

1.7.2. Dựa vào chiến lược kiểm thử

- **Kiểm thử thủ công:**

- Thực hiện kiểm thử mọi thứ bằng tay, từ viết test case đến thực hiện test.

- **Kiểm thử tự động:**

- Thực hiện một cách tự động các bước trong kịch bản kiểm thử bằng cách dùng một công cụ trợ giúp
- Kiểm thử tự động nhằm tiết kiệm thời gian kiểm thử

1.7.3. Dựa vào pp tiến hành kiểm thử

- **Kiểm thử tĩnh:**

- Một hình thức của kiểm thử mà phần mềm không được sử dụng thực sự.
- Thường không kiểm thử chi tiết mà chủ yếu kiểm tra tính đúng đắn của code, thuật toán hoặc tài liệu
- Các hoạt động: Đi xuyên suốt (walk through), thanh tra (inspection)

- **Kiểm thử động:**

- Một hình thức kiểm thử phần mềm chạy mã lập trình thực tế trong các tình huống, diễn ra khi bản thân chương trình đó đang được sử dụng
- Kiểm thử động có thể bắt đầu trước khi chương trình đã hoàn tất.

1.7.4. Dựa vào kỹ thuật kiểm thử

- **Kiểm thử hộp trắng**

- Kiểm thử theo góc nhìn thực hiện
- Cần có kiến thức về chi tiết thiết kế và thực hiện bên trong
- Kiểm thử dựa vào phủ các lệnh, các nhánh, phủ các điều kiện con

- **Kiểm thử hộp đen**

- Kiểm thử theo góc nhìn sử dụng
- Kiểm thử dựa trên các yêu cầu và đặc tả sử dụng thành phần phần mềm
- Không đòi hỏi kiến thức về chi tiết thiết kế và thực hiện ở bên trong chương trình

1.8. Một số khái niệm liên quan

- **Xác minh (Verification)**

- **Xác minh** là quy trình xác định xem sản phẩm của một công đoạn trong quy trình phát triển phần mềm có thỏa mãn các yêu cầu đặt ra trong công đoạn trước hay không?(Ta có đang xây dựng đúng sản phẩm mà được đặc tả không?)
- Xác minh quan tâm tới việc ngăn chặn lỗi giữa các công đoạn
- Xác minh thường là hoạt động kỹ thuật vì nó có sử dụng các kiến thức về các yêu cầu, các đặc tả rời rạc của phần mềm
- Các hoạt động của xác minh bao gồm: Kiểm thử (Testing) và Rà soát loại (Review)

1.8. Một số khái niệm liên quan

- **Thẩm định (Validation)**

- Là tiến trình nhằm chỉ ra toàn bộ hệ thống đã phát triển xong phù hợp với tài liệu mô tả yêu cầu. Thẩm định là quá trình kiểm chứng chúng ta xây dựng phần mềm có đúng theo yêu cầu khách hàng không?
- Thẩm định chỉ quan tâm đến sản phẩm cuối cùng không còn lỗi

1.4. Một số khái niệm liên quan

Verification

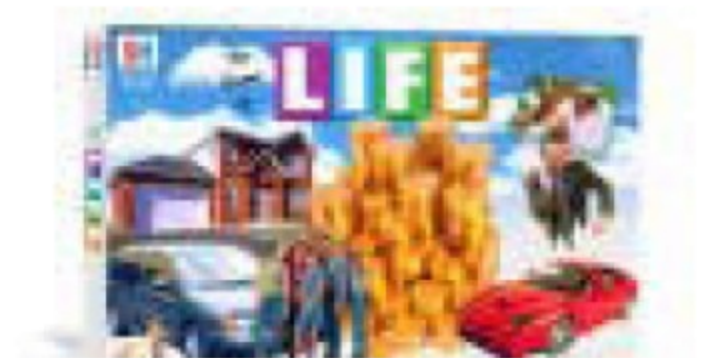
Are we building the product **right**?



“I landed on “Go” but didn’t get my \$200!”

Validation

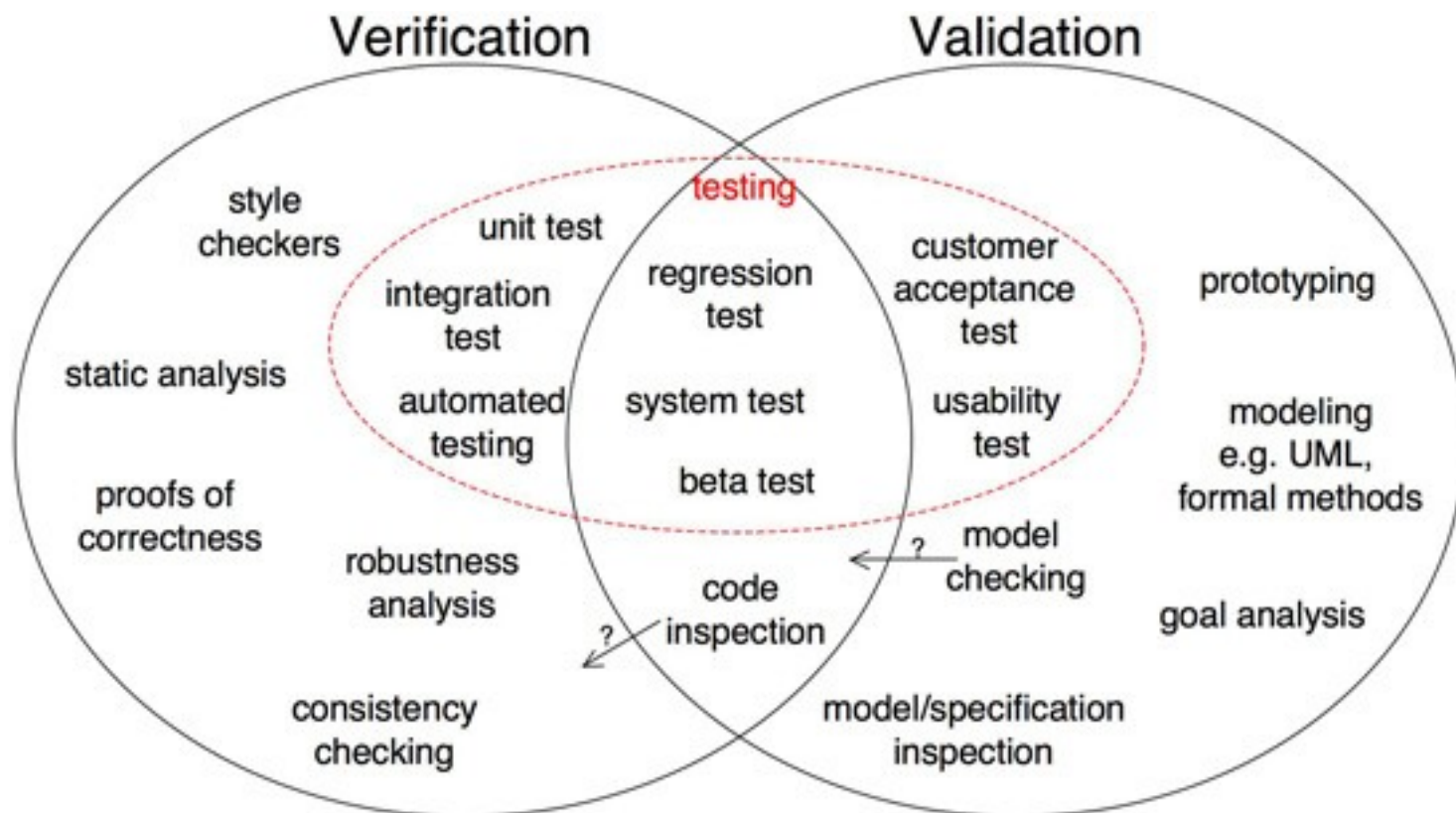
Are we building the **right** product?



“I know this game has money and players and “Go” – but this is not the game I wanted.”

1.7. Một số khái niệm liên quan

- Xác minh và thẩm định (Verification & Validation)



1.7. Một số khái niệm liên quan

- **Dữ liệu kiểm thử** (test data): Dữ liệu cần cung cấp để phần mềm có thể thực thi để kiểm thử
- **Kịch bản kiểm thử** (test scenario): Các bước thực hiện khi kiểm thử
- **Kỹ sư kiểm thử** (tester): người thực hiện kiểm thử

1.8. Một số khái niệm liên quan

- **Ca kiểm thử** (test case): chứa các thông tin cần thiết để kiểm thử thành phần phần mềm theo 1 mục tiêu xác định.
- Test case gồm bộ 3 thông tin { tập dữ liệu đầu vào, thứ tự thực hiện, tập kết quả kỳ vọng}
 - Tập dữ liệu đầu vào (input): gồm các giá trị dữ liệu cần thiết để thành phần phần mềm dùng và xử lý
 - Tập kết quả kỳ vọng (output): kết quả mong muốn sau khi thành phần phần mềm xử lý dữ liệu nhập
 - Thứ tự thực hiện:

1.8. Một số khái niệm liên quan

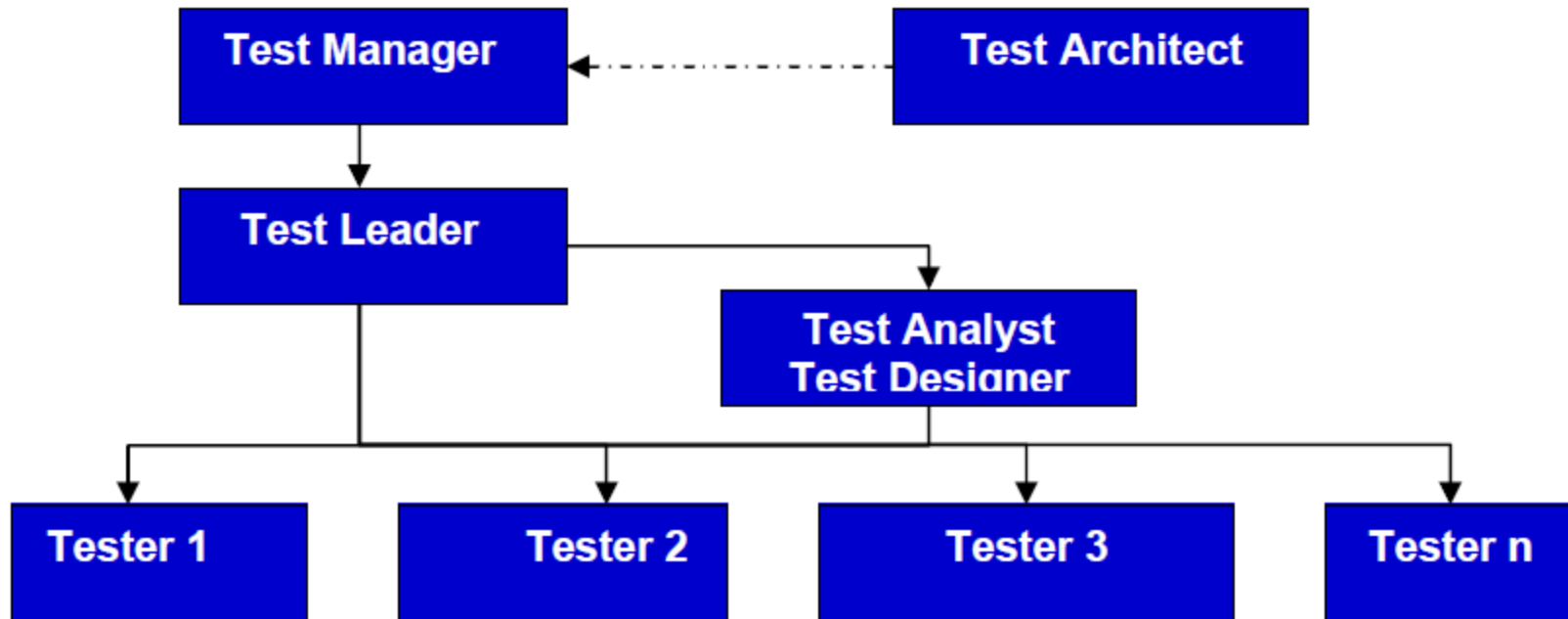
- **Ca kiểm thử** (test case): chứa các thông tin cần thiết để kiểm thử thành phần phần mềm theo 1 mục tiêu xác định.
- Test case gồm bộ 3 thông tin { tập dữ liệu đầu vào, thứ tự thực hiện, tập kết quả kỳ vọng}
 - Tập dữ liệu đầu vào (input): gồm các giá trị dữ liệu cần thiết để thành phần phần mềm dùng và xử lý
 - Tập kết quả kỳ vọng (output): kết quả mong muốn sau khi thành phần phần mềm xử lý dữ liệu nhập
 - Thứ tự thực hiện: các bước để hoàn thành ca kiểm thử từ lúc nhập dữ liệu đầu vào tới lúc nhận được kết quả đã qua xử lý của phần mềm

1.8. Một số khái niệm liên quan

- Thiết kế các ca kiểm thử dựa trên thứ tự thực hiện các ca kiểm thử:
- **Kiểm thử nối tầng**
 - Một ca kiểm thử này có thể được xây dựng dựa trên một ca kiểm thử khác.
 - Ưu điểm của phong cách này là mỗi ca kiểm thử sẽ trở nên nhỏ hơn và đơn giản hơn.
 - Nhược điểm là nếu một ca kiểm thử sai, sẽ dẫn tới ca kiểm thử xây dựng dựa trên ca kiểm thử đó sẽ sai theo
- **Kiểm thử độc lập**
 - Mỗi ca kiểm thử được xây dựng độc lập, không dựa vào các ca kiểm thử khác, và không đòi hỏi các ca kiểm thử khác phải thực hiện thành công.
 - Ưu điểm của phong cách này là một ca kiểm thử có thể thực hiện bất cứ lúc nào, không phụ thuộc vào thứ tự thực hiện các ca kiểm thử.
 - Nhược điểm chính là mỗi ca kiểm thử sẽ trở nên cồng kềnh và phức tạp hơn, và cũng làm cho quá trình thiết kế, thực hiện và bảo trì trở nên khó khăn hơn.

1.9. Đối tượng thực hiện kiểm thử

- Sơ đồ tổ chức của đội kiểm thử



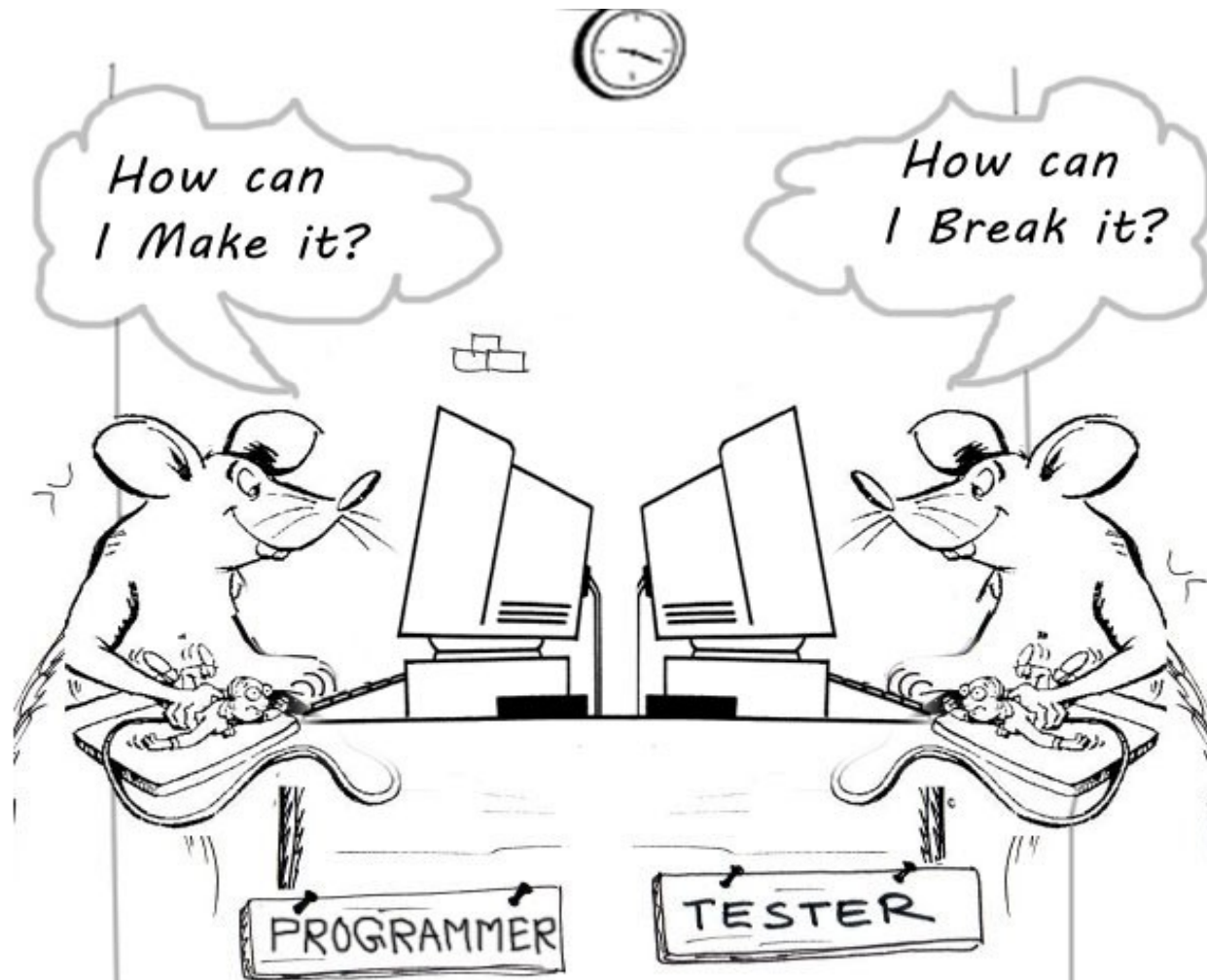
1.9. Đối tượng thực hiện kiểm thử



Lập trình viên



Kiểm thử độc lập



*They are not so much different,
however they have different path to acheive the same goal of
providing quality to the stakeholders!*

1.10. Các điểm cần lưu ý khi kiểm thử

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

1.11. Các hạn chế của việc kiểm thử

- Không thể chắc chắn đặc tả phần mềm đúng hoàn toàn
- Không thể chắc chắn hệ thống hay tool kiểm thử là đúng
- Không có tool kiểm thử nào thích hợp cho mọi phần mềm
- Kỹ sư kiểm thử không chắc chắn họ hiểu đầy đủ về sản phẩm
- Không có tài nguyên để thực hiện tất cả các kiểm thử
- Không thể tìm ra được tất cả các lỗi

Bài 2. Quy trình kiểm thử phần mềm

2.1. Các vấn đề liên quan tới quy trình kiểm thử

2.2. Quy trình kiểm thử

2.3. Cấu trúc của bản kế hoạch kiểm thử

2.1. Các vấn đề liên quan tới quy trình kiểm thử

- 2.1.1. Khái niệm quy trình kiểm thử phần mềm
- 2.1.2. Tầm quan trọng của kiểm thử theo quy trình
- 2.1.3. Vị trí của kiểm thử trong vòng đời phần mềm

2.1.1. Khái niệm Quy trình kiểm thử PM

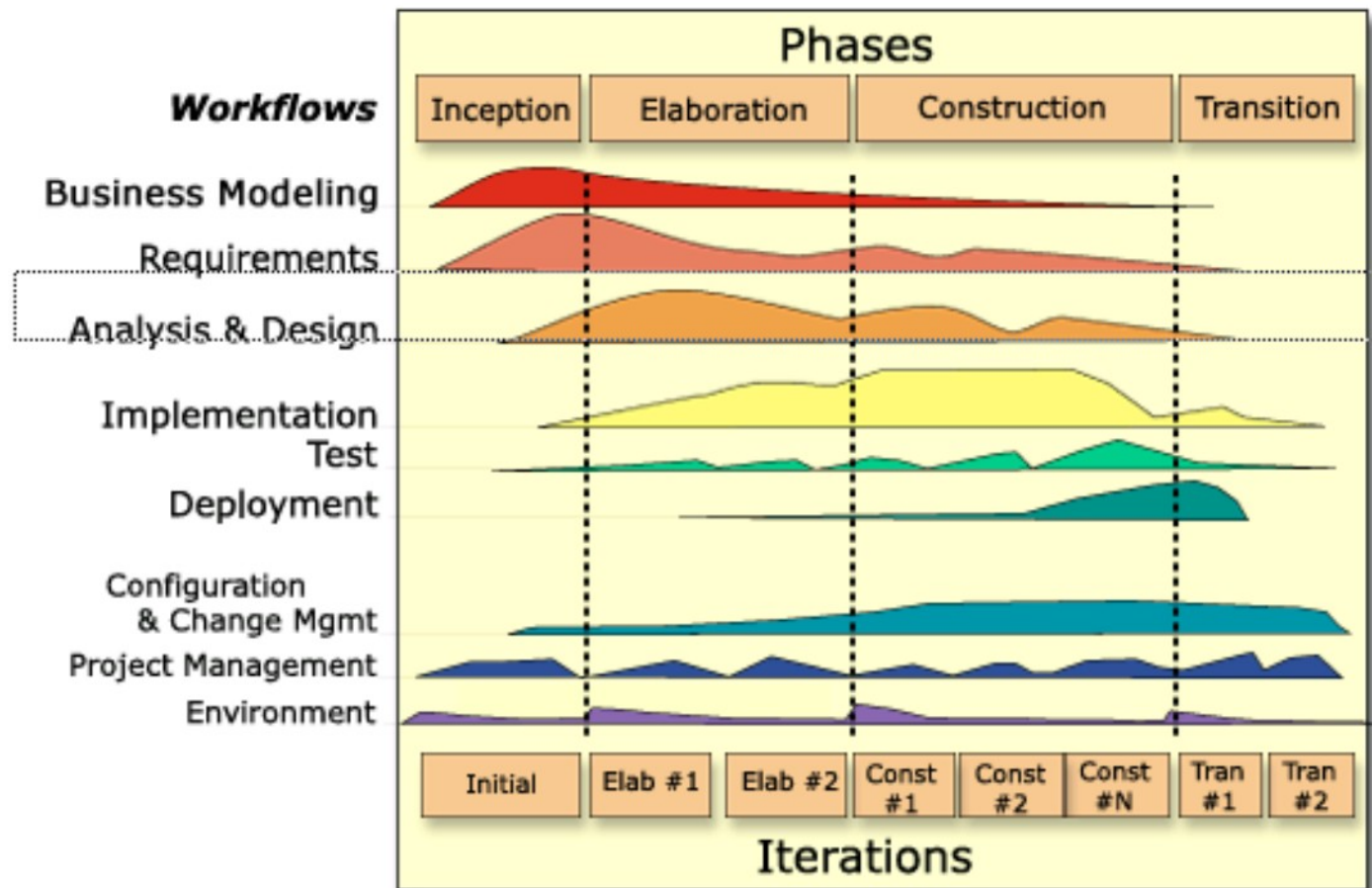
- Khái niệm Quy trình (theo IEEE): là một tập hợp các bước có thứ tự được thực hiện cho một mục đích cụ thể
- Quy trình kiểm thử phần mềm **một tập các hoạt động, các phương thức mà con người phải làm để thực hiện việc kiểm thử cho một phần mềm hay một hệ thống phần mềm**

2.1.2. Tầm quan trọng của kiểm thử theo quy trình

- Cần làm rõ vai trò và trách nhiệm của việc kiểm thử phần mềm
- Cần làm rõ các công đoạn, các bước kiểm thử
- Cần hiểu và phân biệt các tính chất kiểm thử (tại sao phải kiểm thử), các bước kiểm thử (khi nào thực hiện), và các kỹ thuật kiểm thử (kiểm thử bằng cách nào?)

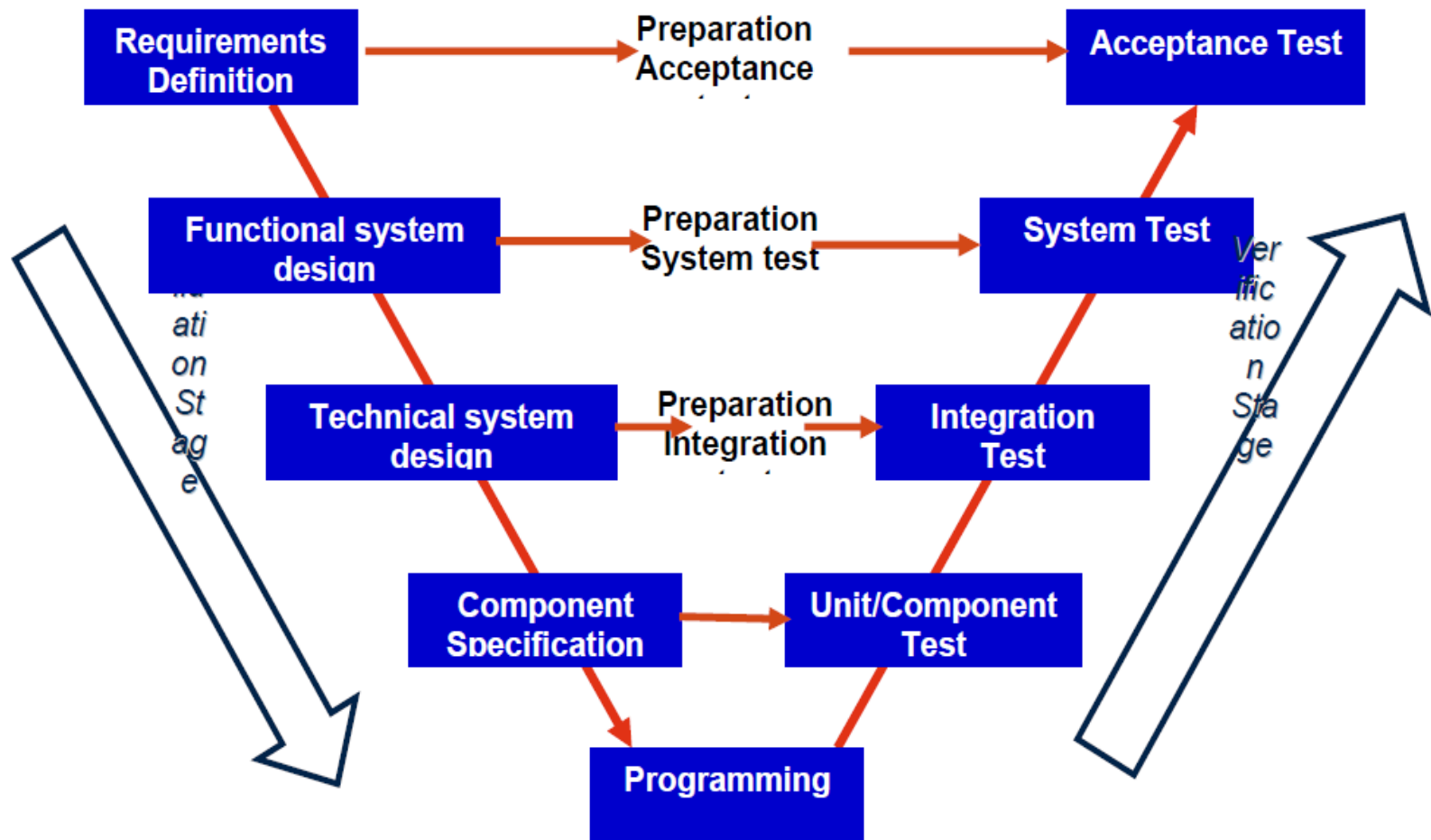
2.1.3. Vị trí của kiểm thử trong vòng đời phần mềm

- Kiểm thử được thực hiện sau mỗi bước lặp với quy trình RUP



2.1.3. Vị trí của kiểm thử trong vòng đời phần mềm

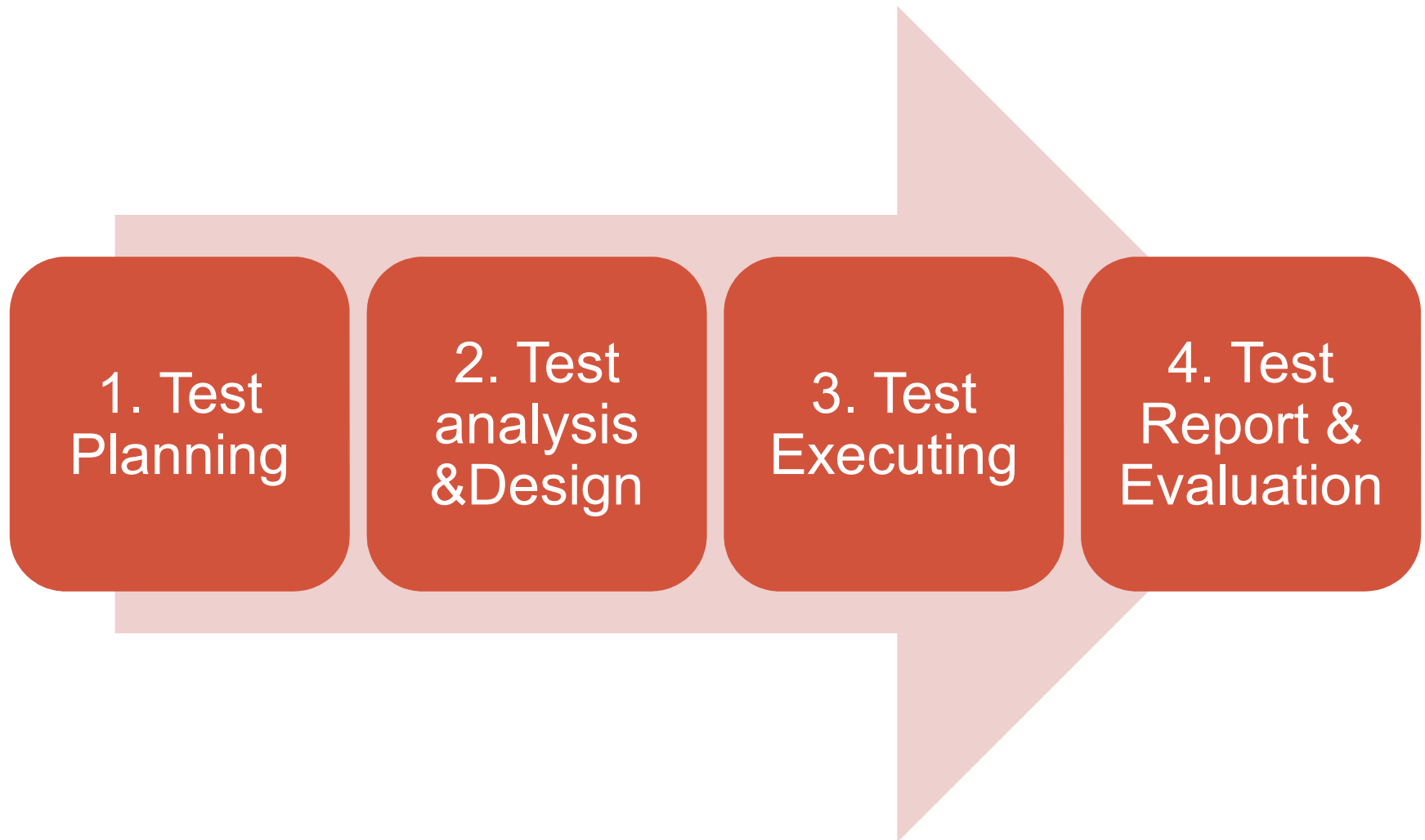
- Mô hình chữ V



2.1.3. Vị trí của kiểm thử trong vòng đời phần mềm

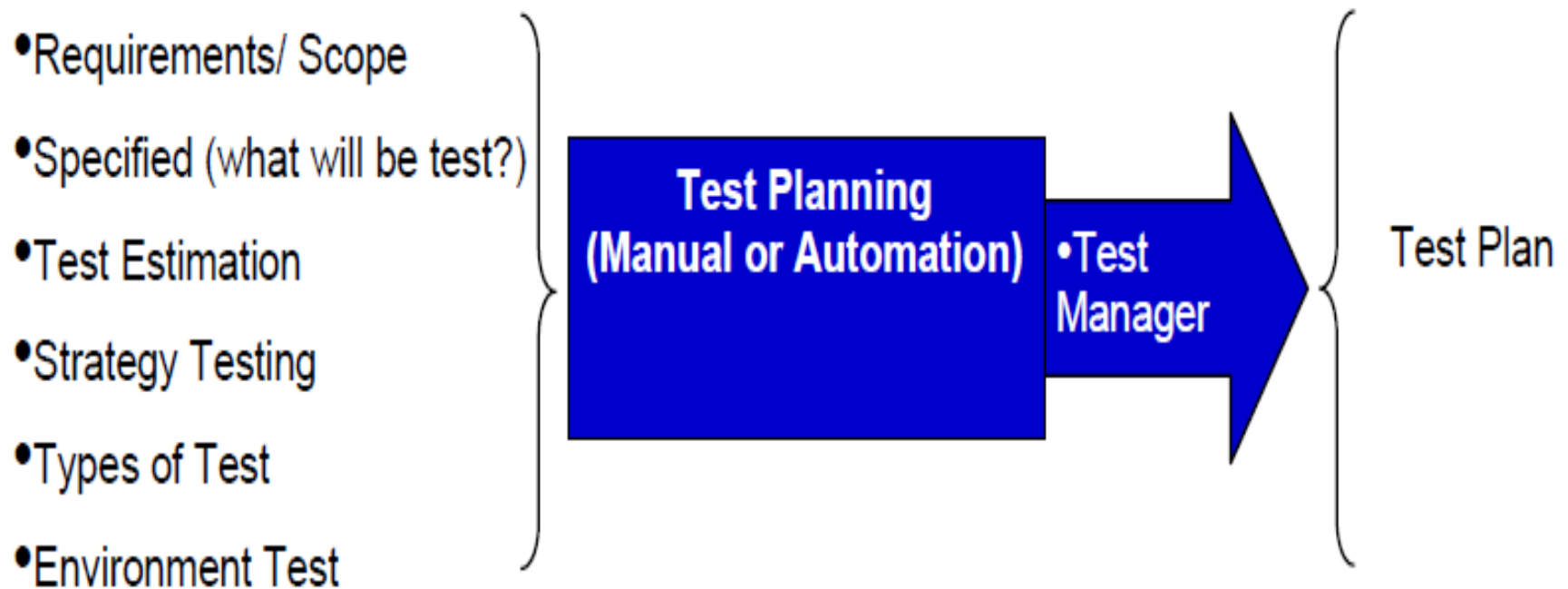
- Các tính chất cần ghi nhận của mô hình chữ V
- Các hoạt động thực hiện và các hoạt động kiểm thử được tách biệt nhưng độ quan trọng là như nhau
- Mô hình này minh họa cho mọi hoạt động của quá trình thẩm định và xác minh

2.2. Quy trình kiểm thử tổng quát



2.2.1. Lập kế hoạch kiểm thử

- Lập kế hoạch kiểm thử là quá trình tạo ra bản kế hoạch kiểm thử
- Bản kế hoạch kiểm thử là tài liệu mô tả về phạm vi, cách tiến cần các nguồn lực và kế hoạch thực hiện kiểm thử



2.2.1. Lập kế hoạch kiểm thử

- Mục tiêu của lập kế hoạch kiểm thử:
 - Thiết lập được mục tiêu dài hạn và ngắn hạn của việc kiểm thử.
 - Nhận biết được các rủi ro có thể xảy ra
 - Xác định được cách tiếp cận và kế hoạch cho việc kiểm thử.

2.2.1. Lập kế hoạch kiểm thử

- **6 nhiệm vụ chính trong hoạt động lập kế hoạch:**

1. Xác định được phạm vi kiểm thử, các rủi ro có thể xảy ra, xác định được mục tiêu kiểm thử để đảm bảo đo đạc được chất lượng phần mềm sản xuất ra
2. Xác định cách tiếp cận việc kiểm thử: nên sử dụng kỹ thuật kiểm thử nào, độ bao phủ kiểm thử cho phép...
3. Thực thi theo chính sách và chiến lược kiểm thử
4. Xác định nguồn lực kiểm thử cần thiết.
5. Lên kế hoạch cho hoạt động phân tích, thiết kế, thực thi...
6. Xác định tiêu chí kết thúc kiểm thử

2.2.1. Lập kế hoạch kiểm thử

- Test Manager hoặc Test Leader sẽ xây dựng kế hoạch ban đầu về kiểm thử:
 - Định nghĩa phạm vi kiểm thử
 - Nhận dạng các yếu tố để kiểm thử
 - Ước lượng kiểm thử
 - Chiến lược kiểm thử
 - Xây dựng môi trường kiểm thử
 - ...
- Kế hoạch kiểm thử cần được:
 - Xem lại bởi QA, Business Analysis, PM, Customer
 - Chấp thuận bởi PM và Customer
 - Hiệu chỉnh trong suốt chu kỳ kiểm thử để phản ánh các thay đổi nếu cần thiết

2.2.1. Lập kế hoạch kiểm thử

- Hoạt động kiểm soát kiểm thử: là hoạt động nhằm quản lý việc kiểm thử được thực hiện theo đúng kế hoạch
- Kế hoạch kiểm thử cần phải được xây dựng sớm như có thể có trong mỗi chu kỳ phát triển phần mềm để :
 - ☐Tập hợp và tổ chức các thông tin kiểm thử cần thiết.
 - ☐Cung cấp thông tin về qui trình kiểm thử sẽ xảy ra trong tổ chức kiểm thử.
 - ☐Cho mỗi thành viên trong đội kiểm thử có hướng đi đúng.
 - ☐Gán các trách nhiệm rõ ràng cụ thể cho mỗi thành viên đội kiểm thử.
 - ☐Có lịch biểu làm việc rõ ràng và các thành viên có thể làm việc với nhau tốt.
- Giám sát được tình trạng hiện tại của việc kiểm thử
- Đưa ra các hành động nhằm điều chỉnh kịp thời các hoạt động kiểm thử để đảm bảo kế hoạch

2.2.1. Lập kế hoạch kiểm thử

- 5 nhiệm vụ của hoạt động kiểm soát kiểm thử
 1. Đo đạc, phân tích các kết quả từ việc kiểm thử.
 2. Giám sát và ghi lại tiến độ, độ bao phủ và tiêu chí kết thúc.
 3. Cung cấp thông tin thường xuyên cho các bên liên quan.
 4. Đề xuất các hành động hiệu chỉnh.
 5. Đưa ra các quyết định để tiếp tục hay dừng việc kiểm thử

2.2.2. Phân tích và thiết kế kiểm thử

Là hoạt động chuyển các mục tiêu của kiểm thử thành các trường hợp kiểm thử cụ thể



2.2.2. Phân tích và thiết kế kiểm thử

- Mục tiêu của phân tích và thiết kế kiểm thử:
 - Xây dựng được bộ khung các tình huống cần kiểm thử (high level test case). Trong đó:
 - Các test case cần bao phủ tất cả khía cạnh kiểm thử cho từng yêu cầu phần mềm
 - Các test case cần bao phủ tất cả yêu cầu trong các chiến lược kiểm thử
 - Nếu cần kiểm thử tự động, test designer sẽ xây dựng các kịch bản kiểm thử tự động dựa trên các test case/ test procedure

2.2.2. Phân tích và thiết kế kiểm thử

- **Các hoạt động chính của quá trình phân tích và thiết kế kiểm thử:**

1. Kiểm tra lại tất cả các loại tài liệu của dự án: bản đặc tả yêu cầu hệ thống, thiết kế kiến trúc, thiết kế chi tiết, nguyên mẫu giao diện của hệ thống
2. Phân tích và đánh giá khả năng kiểm thử được của hệ thống dựa trên yêu cầu của khách hàng.
3. Xác định và đặt thứ tự ưu tiên cho các điều kiện kiểm thử dựa trên kết quả phân tích các chức năng cần kiểm thử, bản mô tả các chức năng đó
4. Thiết kế và đặt ưu tiên cho các tình huống kiểm thử mức cao
5. Xác định dữ liệu kiểm thử cần thiết cho các điều kiện và trường hợp kiểm thử
6. Thiết kế cho việc thiết lập môi trường kiểm thử, xác định yêu cầu về cơ sở hạ tầng và các công cụ cần thiết
7. Tạo mối liên hệ giữa yêu cầu khách hàng và các trường hợp kiểm thử để kiểm soát được hoạt động kiểm thử và sự thay đổi yêu cầu của khách hàng nếu có

2.2.2. Phân tích và thiết kế kiểm thử

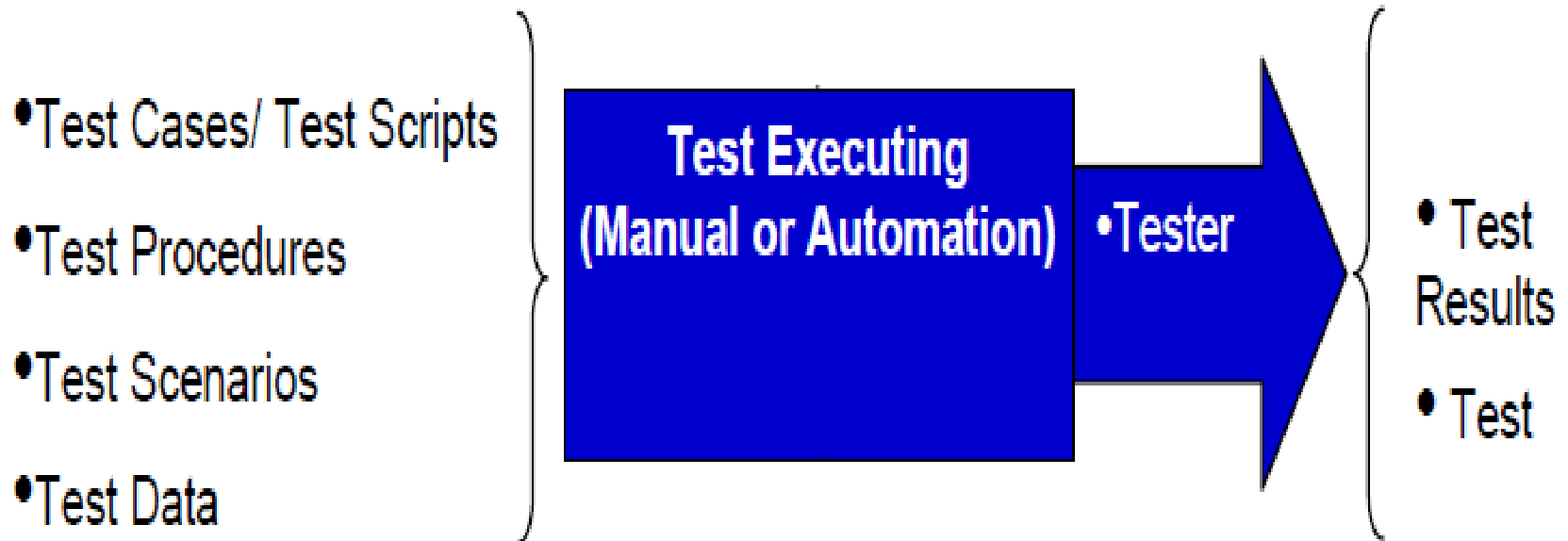
- Các test case cần được:
 - Xem xét lại bởi Project Leader, Developer có liên quan, các Testers khác, Test Leader, Business Analysis và Customer.
 - ☐ Chấp thuận bởi Test Leader hoặc Customer
 - Hiệu chỉnh/cập nhật nếu Tester đã tìm được những lỗi mà không nằm trong các testcase hiện có.

2.2.3. Thực hiện kiểm thử

- Thực hiện kiểm thử là quá trình bao gồm:
 - Phát triển và đặt thứ tự ưu tiên cho các thủ tục kiểm thử (test case/suite), tạo dữ liệu kiểm thử (test data), chuẩn bị các dụng cụ kiểm thử nếu có (test harness), viết kịch bản kiểm thử tự động (test scripts), chuẩn bị môi trường kiểm thử (test environment)
 - Chạy thử một thành phần chức năng hay cả một hệ thống dựa trên các sản phẩm, tài liệu đã chuẩn bị ở bước trên nhằm đưa ra kết quả thực tế

2.2.3. Thực hiện kiểm thử

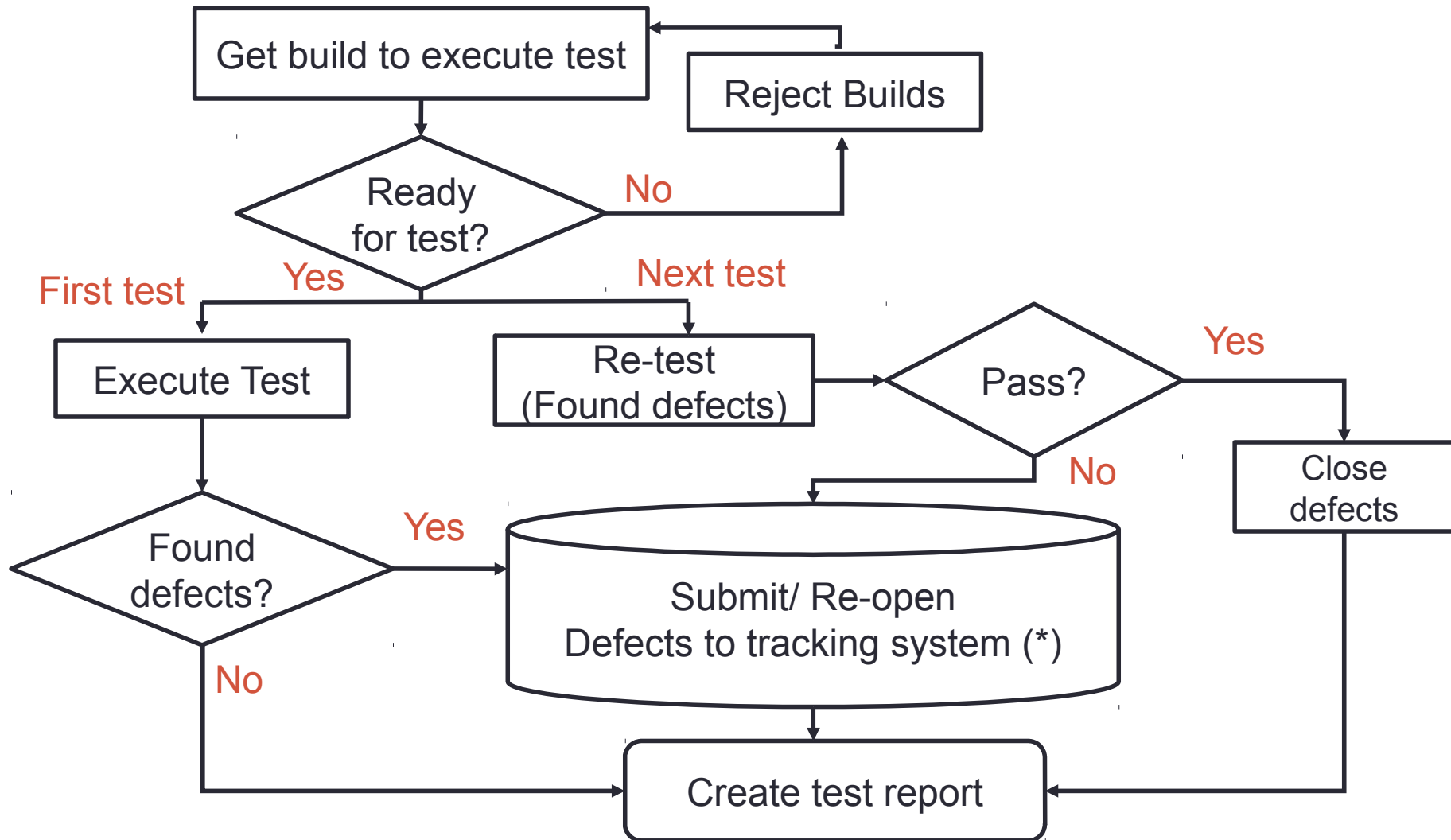
- Mục tiêu của chuẩn bị và thực hiện kiểm thử
 - Xây dựng được các thủ tục kiểm thử
 - Cài đặt môi trường và dữ liệu kiểm thử



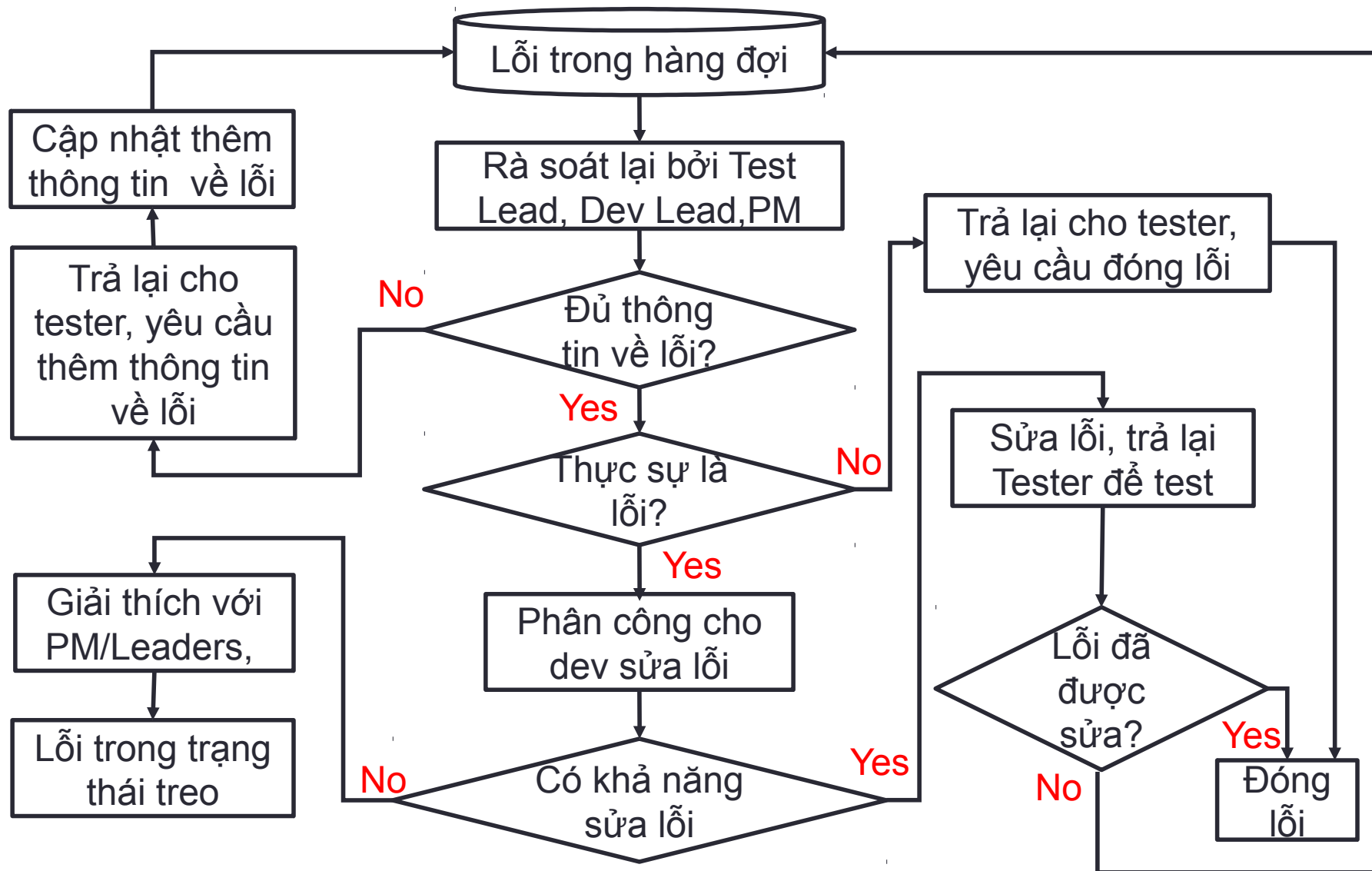
2.2.3. Thực hiện kiểm thử

- Nhiệm vụ chính của thực hiện kiểm thử:
 1. Phát triển và xét thử tự ưu tiên cho các tình huống kiểm thử, tạo dữ liệu và chuẩn bị các công cụ kiểm thử, viết các tình huống kiểm thử tự động
 2. Xây dựng các bộ kiểm thử từ các tình huống kiểm thử để kiểm thử đạt hiệu quả nhất
 3. Cài đặt và kiểm tra môi trường kiểm thử
 4. Thực hiện kiểm thử cho một bộ, hoặc theo các
 5. Ghi lại kết quả của kiểm thử
 6. So sánh kết quả kiểm thử thực tế với kết quả mong đợi
 7. Báo cáo sự cố và phân tích nguyên nhân
 8. Lặp lại hoạt động kiểm thử cho mỗi sự cố đã được xác định hoặc kiểm tra lại để xác nhận lỗi xảy ra không ảnh hưởng đến các phần khác của phần mềm

2.2.3. Thực hiện kiểm thử



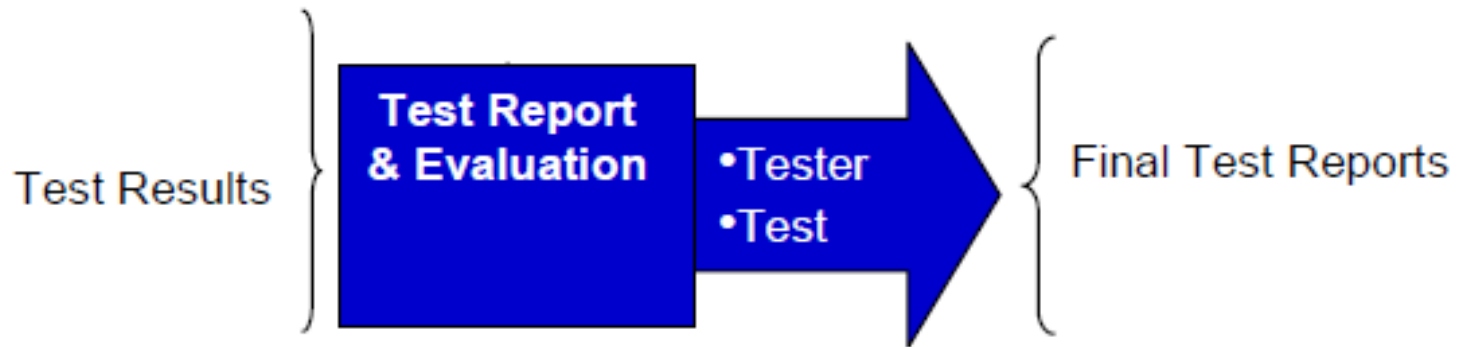
Quy trình xử lý lỗi



2.2.4. Báo cáo và đánh giá kiểm thử

- Báo cáo và đánh giá kiểm thử là hoạt động đánh giá kết quả kiểm thử so với mục tiêu đã đề ra
- Mục tiêu chính: Để đánh giá việc kiểm thử như thế nào là đủ cho mỗi giai đoạn kiểm thử
- 3 nhiệm vụ chính:
 - Kiểm tra các kết quả kiểm thử so với các tiêu chí dừng kiểm thử.
 - Đánh giá tình trạng hiện tại để tiến hành kiểm thử thêm nếu cần.
 - Viết bản báo cáo tổng kết các hoạt động kiểm thử cho các bên liên quan

2.2.4. Báo cáo và đánh giá kiểm thử



- Test Manager hoặc Test Leader sẽ phân tích các lỗi trong hệ thống theo dõi các lỗi.
 - Tạo các báo cáo lỗi.
 - Đánh giá các kết quả kiểm thử, thống kê các yêu cầu thay đổi.
 - ☐ Tính và phân phối các thông tin đo lường hoạt động kiểm thử.
 - ☐ Tạo bảng tổng kết đánh giá hoạt động kiểm lỗi.
 - ☐ Xác định xem đã đạt tiêu chí thành công và hoàn thành kiểm thử chưa.

2.2.4. Báo cáo và đánh giá kiểm thử

- **Hoạt động kết thúc kiểm thử**

- Là hoạt động thu thập dữ liệu từ các hoạt động kiểm thử, tổng hợp các kinh nghiệm dựa trên việc kiểm tra và hoàn thiện bộ sản phẩm kiểm thử, dữ liệu sau khi thu thập sẽ là căn cứ, cơ sở để phân tích số liệu, đưa ra các bài học để áp dụng cho tương lai

- **Mục tiêu chính** của hoạt động kết thúc kiểm thử là thu thập dữ liệu nhằm:

- Cung cấp dữ liệu cho việc bàn giao sản phẩm phần mềm
- Lưu trữ dữ liệu
- Phân tích cho các hoạt động cải tiến sau này

2.2.4. Báo cáo và đánh giá kiểm thử

- 7 hoạt động chính của hoạt động kết thúc kiểm thử:
 1. Kiểm tra các sản phẩm thực tế được bàn giao so với kế hoạch. Đảm bảo các lỗi được giải quyết hay có kế hoạch giải quyết cho các lần bàn giao sau
 2. Đóng các báo cáo về sự cố hoặc ghi chép các thay đổi cho bất cứ vấn đề nào còn đang để mở
 3. Viết biên bản chấp nhận phần mềm
 4. Lưu trữ các sản phẩm kiểm thử, môi trường kiểm thử và cơ sở hạ tầng cho các lần sử dụng sau
 5. Bàn giao các sản phẩm kiểm thử cho các bộ quản lý dữ liệu và bảo trì sản phẩm
 6. Phân tích các bài học để xác định những điểm cần thay đổi cho dự án sau
 7. Sử dụng các thông tin thu thập được để cải tiến công việc kiểm thử một cách định kỳ.

2.3. Bản kế hoạch kiểm thử

- 2.3.1. Định nghĩa của bản kế hoạch kiểm thử
- 2.3.2. Mục tiêu của bản kế hoạch kiểm thử
- 2.3.3. Quy trình xây dựng bản kế hoạch kiểm thử
- 2.3.4. Cấu trúc của bản kế hoạch kiểm thử

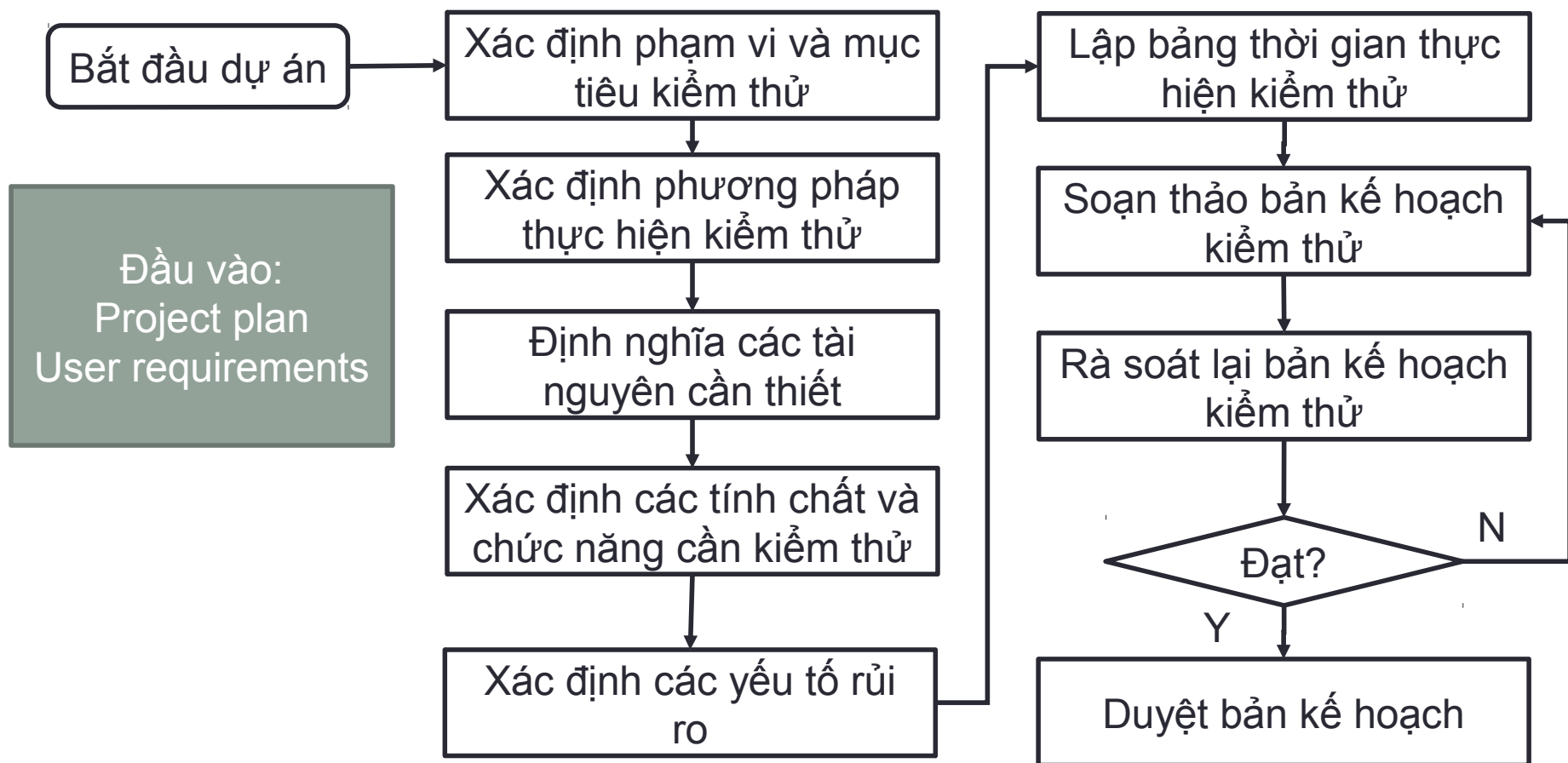
2.3.1. Định nghĩa bản kế hoạch kiểm thử

- Bản kế hoạch kiểm thử là một tài liệu và chứa các kết quả của các hoạt động sau :
 - ☐ Nhận dạng các chiến lược được dùng để kiểm tra và đảm bảo rằng sản phẩm thỏa mãn đặc tả thiết kế phần mềm và các yêu cầu khác về phần mềm.
 - ☐ Định nghĩa các mục tiêu và phạm vi của nỗ lực kiểm thử.
 - ☐ Nhận dạng phương pháp mà đội kiểm thử sẽ dùng để thực hiện công việc kiểm thử.
 - Nhận dạng phần cứng, phần mềm và các tiện ích cần cho kiểm thử.
 - ☐ Nhận dạng các tính chất và chức năng sẽ được kiểm thử.
 - ☐ Xác định các hệ số rủi ro gây nguy hại cho việc kiểm thử.
 - ☐ Lập lịch kiểm thử và phân phối công việc cho mỗi thành viên tham gia.
 - ☐ ...
- Test Manager hoặc Test Leader sẽ xây dựng kế hoạch kiểm thử.

2.3.2. Mục tiêu của bản kế hoạch kiểm thử

1. Xác định phạm vi kiểm thử
2. Nhận diện các rủi ro có thể xảy ra khi thực hiện kiểm thử
3. Xác định các tiêu chí hoàn thành kiểm thử (acceptance criteria)
4. Xác định chiến lược kiểm thử dựa trên phạm vi kiểm thử
5. Xác định các nguồn lực cho kiểm thử
6. Xác định các chỉ số để đánh giá kiểm thử
7. Đưa ra quyết định sớm về việc sử dụng kiểm thử tự động
8. Xác định các lịch trình, thời gian biểu cụ thể cần đưa ra các sản phẩm kiểm thử
9. Giúp các thành viên trong tổ dự án làm việc hiệu quả hơn

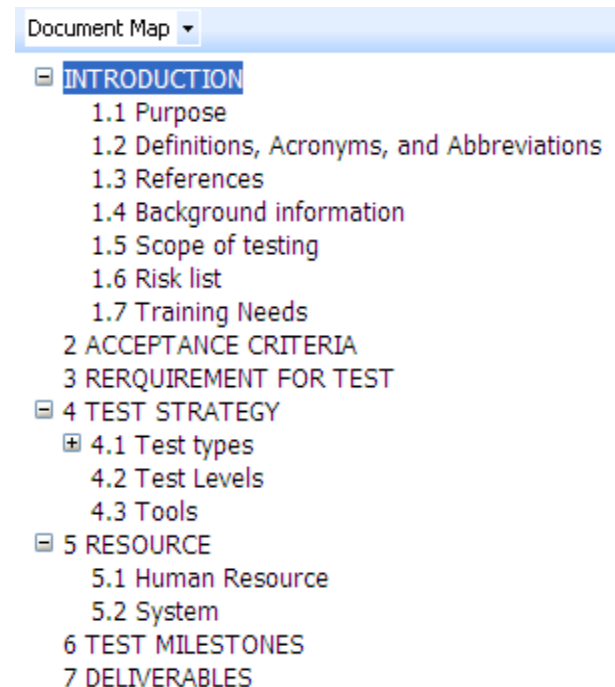
2.3.3. Quy trình xây dựng kế hoạch kiểm thử



2.3.4. Cấu trúc bản kế hoạch kiểm thử

- Bản kế hoạch kiểm thử cơ bản bao gồm 7 thành phần:

1. Introduction
2. Acceptance criteria
3. Requirements for test
4. Test strategy
5. Resources for testing
 - ✓ Human and responsibilities
 - ✓ System: hardware & software
6. Test milestones
7. Deliverables of test: Test Plan, Test Case, Test Reports



Document Map ▼

- [-] INTRODUCTION
 - 1.1 Purpose
 - 1.2 Definitions, Acronyms, and Abbreviations
 - 1.3 References
 - 1.4 Background information
 - 1.5 Scope of testing
 - 1.6 Risk list
 - 1.7 Training Needs
- 2 ACCEPTANCE CRITERIA
- 3 RERQUIREMENT FOR TEST
- [-] 4 TEST STRATEGY
 - [+] 4.1 Test types
 - 4.2 Test Levels
 - 4.3 Tools
- [-] 5 RESOURCE
 - 5.1 Human Resource
 - 5.2 System
- 6 TEST MILESTONES
- 7 DELIVERABLES

2.3.4.1. Giới thiệu chung- Introduction

- Mục đích: Trình bày ngắn gọn về mục đích và tổ chức của tài liệu
- Các định nghĩa, từ viết tắt, thuật ngữ: cung cấp các định nghĩa của các thuật ngữ, từ viết tắt cần thiết để giải thích đúng các kế hoạch kiểm thử
- Tài liệu tham khảo: Liệt kê tất cả những tài liệu dùng để tạo ra bản kế hoạch
- Thông tin cơ bản: Mô tả ngắn gọn về dự án
- Phạm vi kiểm thử:
 - Danh mục các giai đoạn kiểm thử
 - Danh sách các loại hình kiểm thử
 - Danh sách các giả định
 - Các khiếm khuyết theo dự kiến
- Danh mục các rủi ro: Liệt kê các rủi ro có thể ảnh hưởng đến việc thiết kế hoặc thực thi các kiểm thử
- Nhu cầu đào tạo: Liệt kê các nhu cầu đào tạo của các thành viên trong nhóm để thực thi việc kiểm thử

2.3.4.2. Các tiêu chí chấp nhận sản phẩm

Acceptance criteria

- Danh sách các tiêu chí nhằm xác định mức độ chất lượng kiểm thử đủ để bàn giao cho khách hàng hoặc đủ để sang pha tiếp theo
- Các tiêu chí có thể là:
 - Tỷ lệ bao phủ của kiểm thử
 - Tỷ lệ bao phủ thành công
 - Số lượng ca kiểm thử
 - Tỷ lệ lỗi tìm được
 - Tỷ lệ bỏ qua lỗi
 -

2.3.4.3. Các yêu cầu cần kiểm thử- Requirements for test

- Liệt kê các yêu cầu kiểm thử
 - ✓ Yêu cầu chức năng
 - ✓ Yêu cầu phi chức năng
- Liệt kê các đặc tính và chức năng không cần kiểm thử

2.3.4.4. Chiến lược kiểm thử-Test strategy

- Là một tài liệu mô tả việc kiểm thử sẽ được thực hiện như thế nào
- Trong mục chiến lược kiểm thử sẽ xác định các nội dung:
 - Các loại kiểm thử: mỗi loại kiểm thử cụ thể cho từng loại yêu cầu của phần mềm
 - Các cấp độ kiểm thử: cấp độ kiểm thử nào sẽ được thực hiện và loại kiểm thử nào sẽ được thực hiện ở mỗi cấp độ
 - Công cụ kiểm thử: liệt kê đầy đủ các công cụ hỗ trợ kiểm thử

2.3.4.4. Chiến lược kiểm thử-Test strategy

- Có 2 chiến lược kiểm thử cơ bản:
- **Kiểm thử Big bang**(kiểm thử vụ nổ lớn): là chiến lược kiểm thử tích hợp hệ thống một lần duy nhất để được module chức năng (hay hệ thống hoàn chỉnh)
- **Kiểm thử gia tăng**: chiến lược kiểm thử từ thấp tới cao, bao gồm 4 mức:
 - Kiểm thử đơn vị
 - Kiểm thử tích hợp
 - Kiểm thử hệ thống
 - Kiểm thử chấp nhận

2.3.4.5. Nguồn lực dành cho kiểm thử- Resources for testing

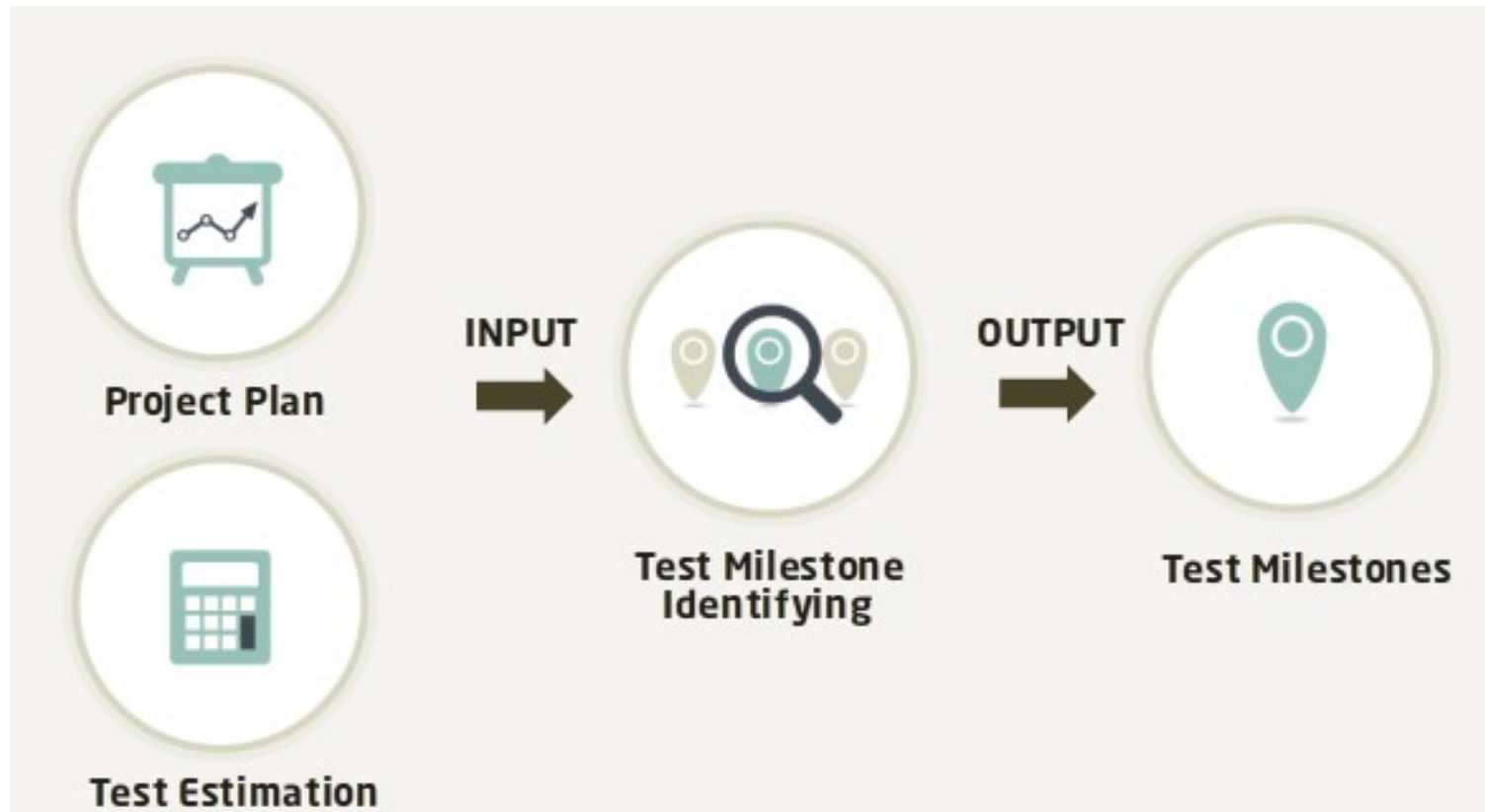
- Nguồn lực hỗ trợ cho hoạt động kiểm thử gồm 2 loại:
 - **Nhân lực:** khi liệt kê phải chỉ rõ người nào làm công việc gì
 - **Nguồn lực hệ thống:** liệt kê các phần mềm, phần cứng để đáp ứng cho việc kiểm thử

2.3.4.6. Các mốc kiểm thử- Test milestones

- **Cột mốc kiểm thử:** một sự kiện, một thành tích kiểm thử quan trọng đạt được hay cần đạt được của dự án.
- Mỗi cột mốc kiểm thử phải bao gồm ít nhất một hoạt động kiểm thử và đạt được một hoặc nhiều sản phẩm kiểm thử.

2.3.4.6. Các mốc kiểm thử- Test milestones

➤ Quy trình xác định mốc kiểm thử:



2.3.4.6. Các mốc kiểm thử- Test milestones

- Cách biểu diễn các mốc kiểm thử

Milestone Name	Effort (pd)	Start Date	End Date
		<dd-MM-yy>	<dd-MM-yy>



2.3.4.7. Sản phẩm cần bàn giao- Deliverables of test

- Liệt kê tên sản phẩm cần bàn giao
- Ngày bàn giao
- Người bàn giao
- Bàn giao cho ai

Đề tài thảo luận

- Tìm hiểu mô hình kiểm tra phần mềm TMM (Testing maturity model)
- Yêu cầu:
- Tìm trong tài liệu của Thạc Bình Cường

Bài 3. Các cấp độ kiểm thử

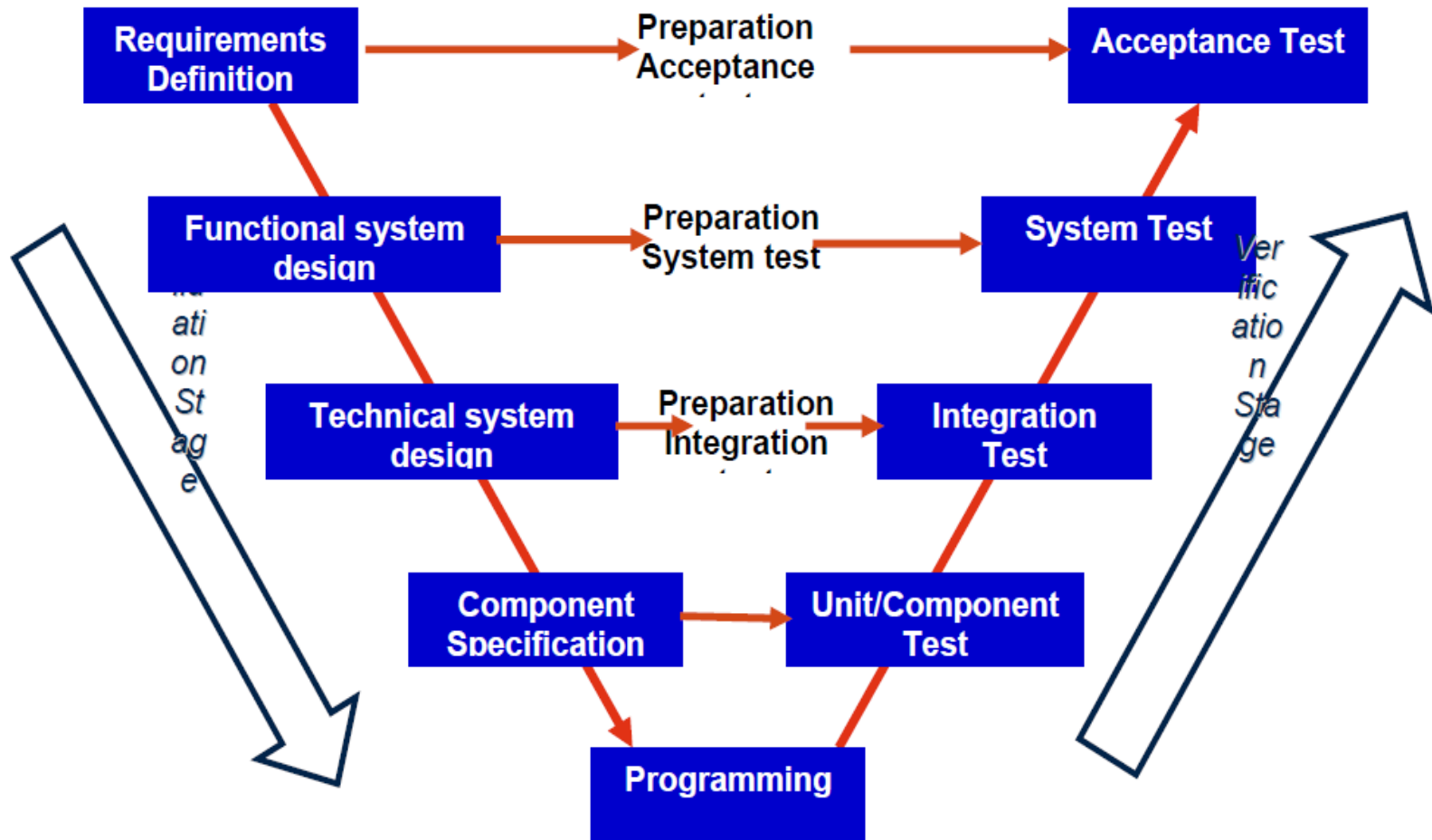
3.1. Kiểm thử đơn vị- Unit testing

3.2. Kiểm thử tích hợp- Integration testing

3.3. Kiểm thử hệ thống- System testing

3.4. Kiểm thử chấp nhận- Acceptance testing

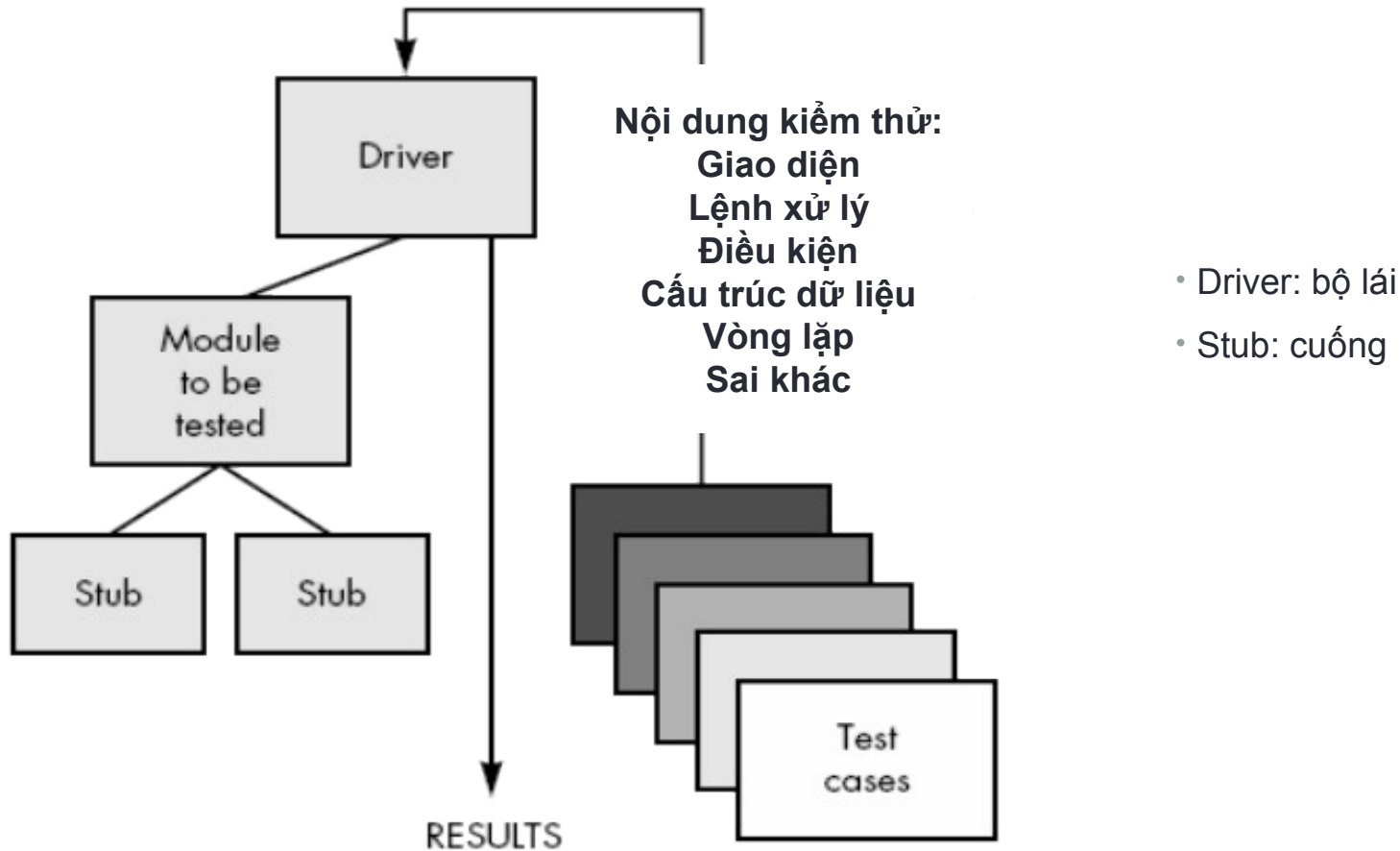
3. Các cấp độ kiểm thử



3.1. Kiểm thử đơn vị

- Kiểm thử đơn vị nhằm kiểm tra đơn vị thiết kế nhỏ nhất- một module phần mềm. Một module hoạt động thường có trao đổi thông tin với module mức dưới và mức trên nó, do đó phạm vi phát hiện lỗi liên quan chặt chẽ tới module này
- **Người tiến hành kiểm thử đơn vị:** lập trình viên cùng nhóm của mình.
- **Kỹ thuật kiểm thử đơn vị:** chủ yếu là hộp trắng, trong các trường hợp cần thiết có thể sử dụng thêm kỹ thuật kiểm thử hộp đen

3.1.1. Mô hình kiểm thử đơn vị



3.1. 2. Nội dung kiểm thử đơn vị:

- a) Kiểm thử giao diện (các tham số vào/ra qua giao diện)
- b) Kiểm thử vào/ra (các file, bộ đệm và các lệnh đóng mở)
- c) Kiểm thử cấu trúc dữ liệu cục bộ (khai báo và sử dụng biến)
- d) Kiểm thử xử lý (các phép toán và tính đúng đắn của kết quả)
- e) Kiểm thử điều kiện logic
- f) Kiểm thử sai tiềm ẩn (về ngoại lệ, mô tả)
- g) Kiểm thử các giá trị biên

a. Kiểm thử dữ liệu qua giao diện

- Kiểm thử dòng dữ liệu qua giao diện của module liên quan đến định lượng và định dạng của các biến và các module sử dụng trên giao diện
- Đặc trưng cụ thể:
 - Số lượng?
 - Định dạng?

a. Kiểm thử dữ liệu qua giao diện

- Các đặc trưng qua giao diện là:
 - Số tham số= số đối số?
 - Tính chất của tham số= tính chất của đối số
 - Đơn vị của tham số= đơn vị của đối số
 - Số đối số được truyền gọi module= số các tham số đầu vào của module?
 - Thứ tự truyền tham số ko chính xác
 - ...
- Ví dụ:

```
String calc_day(date d) {...}
```

```
Void calc_day_test()
```

```
{ date d;
```

```
    string s;
```

```
    ...
```

```
    d= calc_day(s); // truyền tham số ko chính xác
```

```
    ...
```

```
}
```

b. Kiểm thử vào/ra

- Kiểm thử các file, bộ đệm, các lệnh đóng, mở
- Khi thực hiện kiểm thử vào/ ra cần xem xét:
 - Tính chất của các file có đúng đắn ko?
 - Các câu lệnh OPEN/CLOSE có đúng đắn ko?
 - Đặc tả hình thức có đúng đắn ko?
 - Các file có mở trước khi sử dụng ko?
 - Các điều kiện end of file có được xử lý không?
 - Có sai văn bản nào trong thông tin ra?

c. Kiểm thử cấu trúc dữ liệu cục bộ

- Kiểm thử khai báo và sử dụng biến
- Cấu trúc dữ liệu cục bộ cho module có thể sai. Vì thế thiết kế các kiểm thử cần làm lộ ra các loại lỗi sau:
 - Đánh máy ko đúng hoặc ko nhất quán?
 - Giá trị ngầm định hoặc giá trị khởi tạo sai
 - Tên các biến ko đúng (sai chữ hoặc mất chữ)
 - Kiểu dữ liệu không nhất quán
- Vd:

```
int i;  
  
d= i*10; // lỗi chưa khai báo biến d
```


d. Kiểm thử về các xử lý

- Kiểm thử các phép toán và tính đúng đắn của kết quả
- Cần lưu ý các sai về trình tự, độ chính xác:
 - Thứ tự ưu tiên các phép tính số học
 - Sự nhất quán của các phép toán trộn module
 - Khởi tạo/kết thúc không đúng
 - Độ chính xác của kết quả trả về
- Ví dụ:
 - Thực hiện phép toán trên toán hạng ko phải là số:
 - String s1, s2;
 - Int ketqua=s1/s2;

e. Kiểm thử các điều kiện logic

- Các sai kiểu, toán tử, ngữ nghĩa:
 - So sánh các kiểu dữ liệu khác nhau
 - Ưu tiên hoặc toán tử logic không đúng đắn
 - Dự đoán một biểu thức so sánh, trong khi sai số làm cho đẳng thức không chắc có thực
 - Các giá trị so sánh không đúng đắn

- Ví dụ:

```
int ival;
```

```
char sval[20];
```

```
If(ival==sval) {...} //So sánh 2 dữ liệu ko tương thích
```

f. Kiểm thử sai tiềm ẩn

- Các sai tiềm ẩn cần được xem xét là:
 - Mô tả sai(khó hiểu)
 - Dữ liệu ghi không tương ứng với sai đã gặp
 - Điều kiện sai có trước khi xử lý sai
 - Xử lý điều kiện ngoại lệ là không đúng đắn
 - Mô tả sai không cung cấp đủ thông tin để trợ giúp định vị nguyên nhân của sai
 - Ví dụ:
 - If ($a > 0$) then {...}
 - If ($a = 0$) then {...} // thiếu trường hợp xét $a < 0$

g. Kiểm thử các giá trị biên

- Các sai biến, số vòng lặp:
 - Vòng lặp không kết thúc hoặc kết thúc không chính xác
 - Lặp vô hạn
 - Biến lặp bị thay đổi không chính xác
- Sai ở các biên:
 - Kiểm thử ở biên là nhiệm vụ cuối cùng của kiểm thử đơn vị. Các giá trị ở biên thường hay gây ra lỗi.
- VD:

Int i,j;

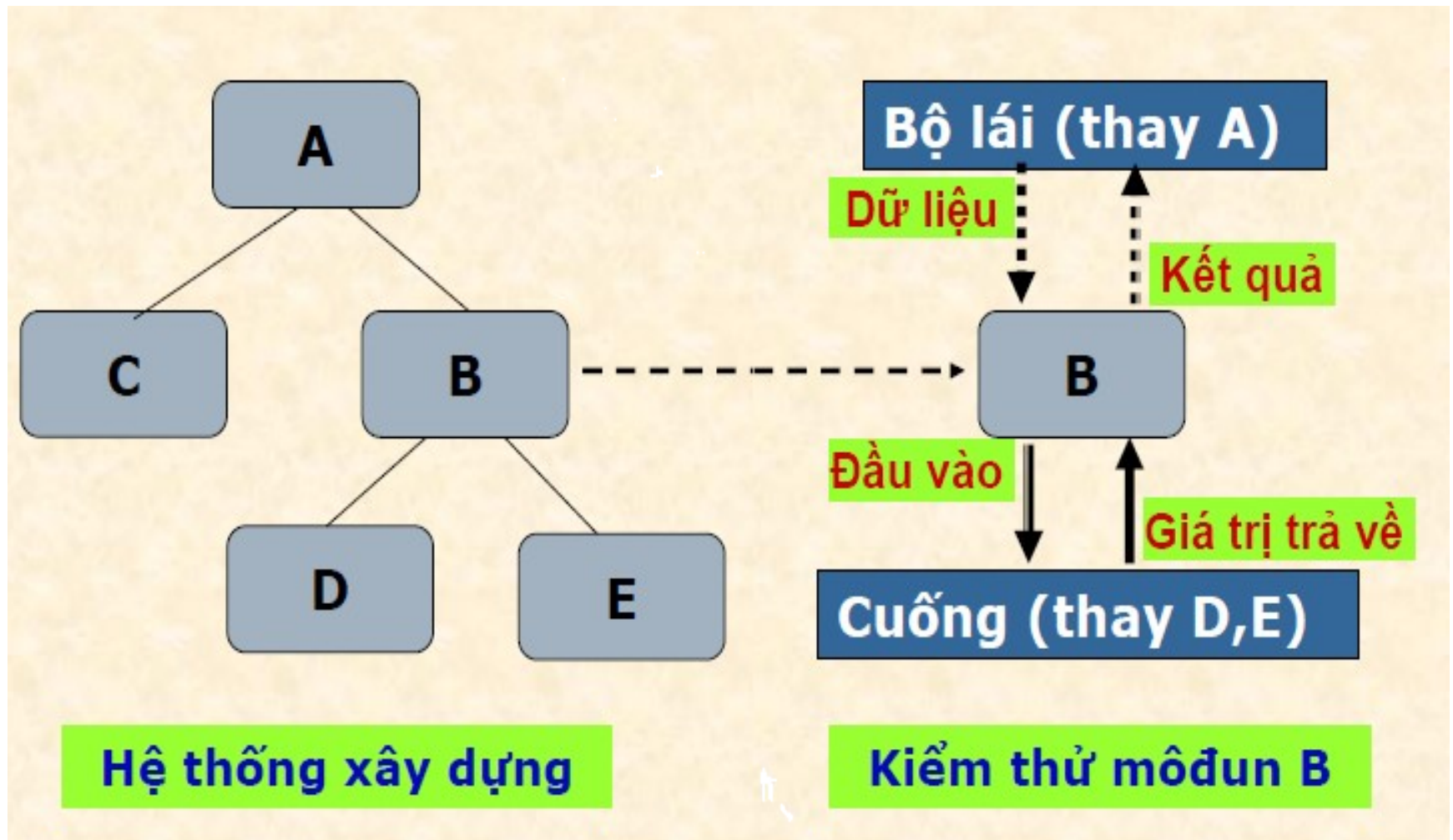
For (i=1;i<=10,i++)

For (j=i+1, j<=10, j++) {...}

3.1.3. Kỹ thuật kiểm thử đơn vị

- Module không phải là một chương trình độc lập, nên cần phát triển thêm các Driver và Stub để tiến hành kiểm thử đơn vị.
- **Bộ lái (driver):** là một hàm main điều khiển việc đưa dữ liệu vào và nhận kết quả của module đang cần kiểm thử
- **Cuồng (stub):** là một chương trình máy tính dùng để thay thế cho một module phần mềm sẽ được xác định sau (IEEE)
- Stub (dummy program): Là một đoạn mã dùng để mô phỏng hoạt động của thành phần còn thiếu.

3.1.3. Kỹ thuật kiểm thử đơn vị



Code example -Stub

```
void functionWeTest(params...) {
```

```
....
```

```
int p= price(param1);
```

```
....
```

```
} // chương trình chính
```

```
void price(int param) {
```

```
    return 10; //không cần quan tâm giá là gì, được tính thế nào chỉ cần  
    giá trị trả về để test module functionWeTest
```

```
} // đây là stub
```

Code example- Driver

```
void function ThatCallPrice(params...) {// đây là driver
    int p= price(param1);
    printf("Price is:%d",p);
}

void price(int param){
    ....
    // chương trình chính để tính được giá trị thật
    ...
}
```


3.2. Kiểm thử tích hợp

- Kiểm thử tích hợp nhằm nhận được một bộ phận chức năng hay một hệ con tốt
- Là một kỹ thuật có tính hệ thống để xây dựng cấu trúc của chương trình
- Từ các module đã qua kiểm thử đơn vị, xây dựng cấu trúc chương trình đảm bảo tuân theo thiết kế
- Có hai cách tích hợp:
- **Tích hợp từng bước.** Theo cách này có 3 chiến lược:
 - Tích hợp từ dưới lên(bottom-up testing)
 - Tích hợp từ trên xuống (top-down testing)
 - Kết hợp 2 chiến lược trên (sandwich testing)
- **Tích hợp đồng thời:** kiểm thử vụ nổ lớn (big bang testing)

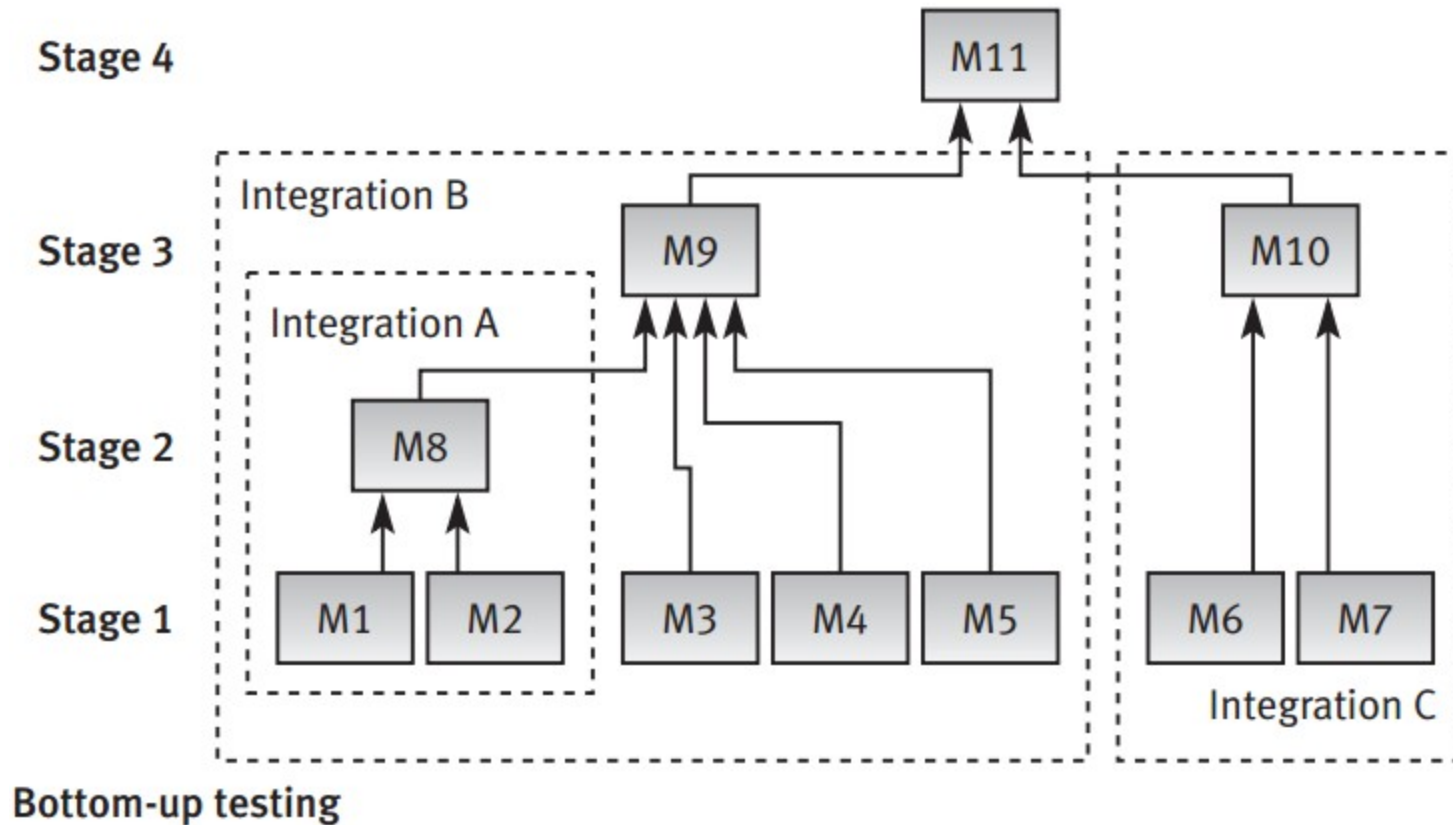
3.2.1. Các lỗi thường gặp khi tích hợp

- Dữ liệu bị mất khi đi qua một giao diện
- Hiệu ứng 1 module vô tình gây ra ảnh hưởng tới các module khác
- Sự kết hợp các chức năng phụ có thể không tạo ra được chức năng chính mong muốn
- Các sai sót nhỏ có thể trở thành thảm họa
- Có thể gặp vấn đề với các cấu trúc dữ liệu toàn cục

3.2.2. Kiểm thử từ dưới lên

- Là quá trình tích hợp và kiểm thử bắt đầu với các module mức thấp.
- Để kiểm thử các module cấp dưới, lúc đầu ta dùng các bộ lái, sau đó thay thế dần các bộ lái bằng các module thượng cấp đã được phát triển.

3.2.2. Kiểm thử từ dưới lên



3.2.2. Kiểm thử từ dưới lên

- **Ưu điểm:**

- Tránh phải tạo các cuồng phức tạp hay tạo các kết quả nhân tạo: do tích hợp từ dưới lên nên chỉ cần tạo ra các bộ lái, các module mức dưới đã được kiểm thử
- Thuận tiện cho phát triển các module cấp dưới: nhờ phát triển từ dưới lên, người thiết kế có thể thiết kế các module dịch vụ dùng chung cho nhiều chức năng của hệ thống

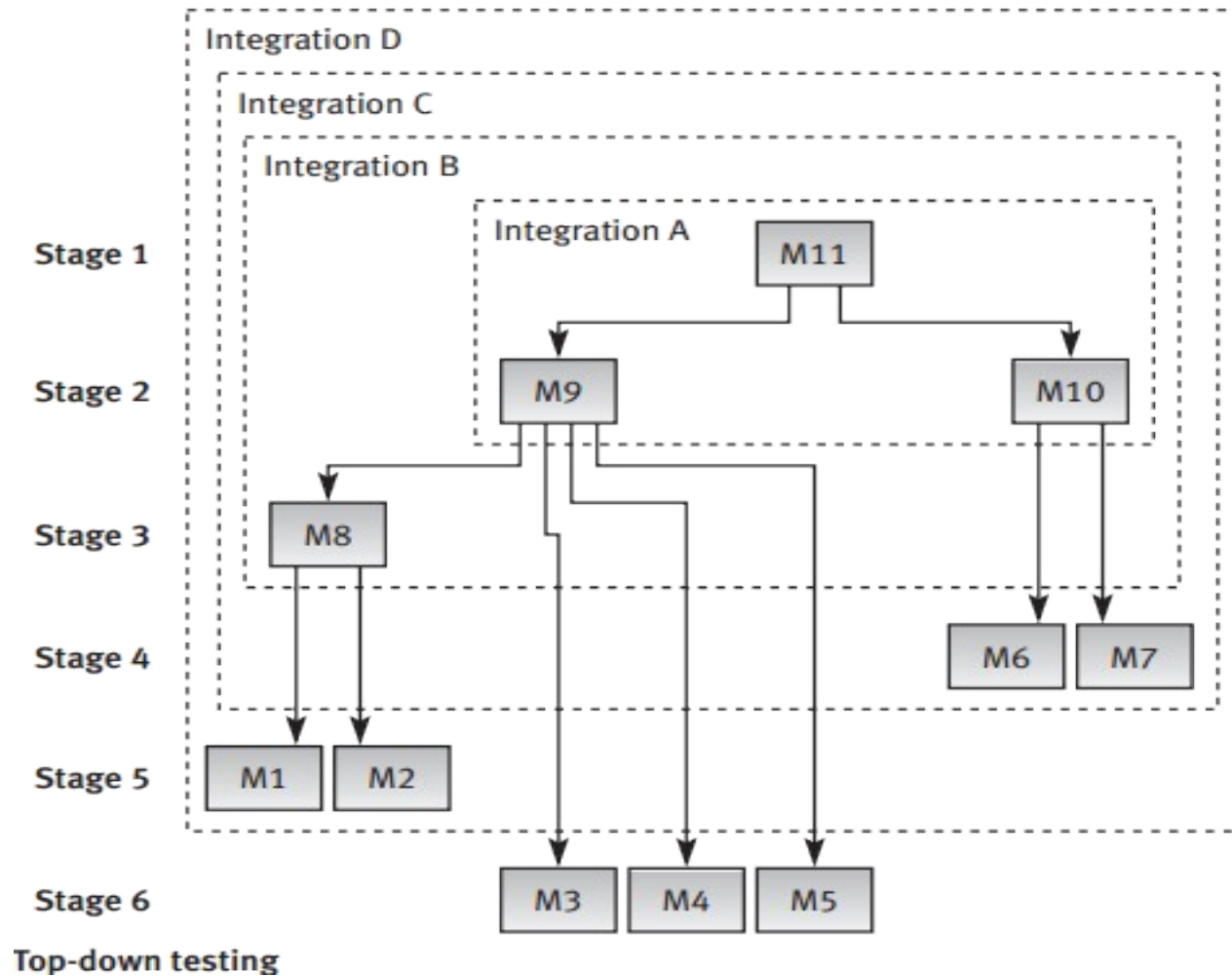
3.2.2. Kiểm thử từ dưới lên

- **Nhược điểm:**
- Chậm phát hiện các lỗi thiết kế: các lỗi tổng thể như phát triển sai chức năng hay hệ thống kém hiệu quả thường bị phát hiện muộn. Do phát hiện lỗi muộn nên chi phí và thời gian sửa lỗi tăng cao
- Chậm có phiên bản của hệ thống làm việc được.

3.2.3. Kiểm thử từ trên xuống

- Kiểm thử từ trên xuống tiến hành kiểm thử các module bắt đầu từ mức cao, các module mức thấp tạm sử dụng các chức năng hạn chế, có giao diện như đặc tả.
- Module mức thấp có thể chỉ đơn giản là các cuống trả lại kết quả với một vài đầu vào được xác định trước. Sau đó các cuống được thay thế dần bằng các module thực đã được phát triển.
- Kiểm thử từ trên xuống có thể thực hiện theo chiều sâu hoặc theo chiều rộng

3.2.3. Kiểm thử từ trên xuống



3.2.3. Kiểm thử từ trên xuống

- **Ưu điểm:**
 - Phát hiện sớm các lỗi thiết kế: dễ dàng phát hiện các lỗi như phát triển nhầm, thiếu chức năng so với đặc tả, do đó làm giảm chi phí cho việc thiết kế và cài đặt lại
 - Có phiên bản hoạt động sớm: kiểm thử từ trên xuống luôn đảm bảo có phiên bản hoạt động sớm, do đó có thể thẩm định tính dùng được của sản phẩm và dùng nó để huấn luyện người dùng
- **Nhược điểm:**
 - Tạo ra các cuống rất phức tạp, có thể dẫn tới việc không kiểm thử được đầy đủ chức năng các module

3.3. Kiểm thử hệ thống

- Kiểm thử hệ thống:
 - Tìm kiếm các lỗi, nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn hệ thống
 - Mức kiểm thử này đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với phần mềm hoặc phần cứng bên ngoài, chẳng hạn các lỗi "tắc nghẽn" (deadlock) hoặc chiếm dụng bộ nhớ
 - Đòi hỏi nhiều thời gian, công sức, thiết bị...
- Mục đích: kiểm thử thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không tìm.
- Phương pháp: kiểm thử hộp đen
- Người thực hiện: kiểm thử viên

3.3. Kiểm thử hệ thống

Khi nào có thể thực hiện kiểm thử hệ thống:

- Hệ thống cần kiểm thử đã hoàn thiện
- Kiểm thử tích hợp và đơn vị đã hoàn thành
- Sản phẩm được tích hợp đúng thiết kế
- Các tài liệu đặc tả đã là bản cuối cùng
- Các tài liệu hỗ trợ kiểm thử như test plan, test case đã hoàn thành.

3.4. Kiểm thử chấp nhận

- Kiểm thử chấp nhận (acceptance testing) : vận hành hệ thống trong môi trường của người sử dụng
- Kiểm thử alpha (alpha testing)
 - Người dùng thực hiện với số liệu giả lập
 - Trong môi trường phát triển
- Kiểm thử beta (beta testing)
 - Người dùng thực hiện với số liệu thực
 - Trong môi trường ứng dụng thực

Bài 4. Các loại hình kiểm thử

- 4.1. Kiểm thử chức năng
- 4.2. Kiểm thử phi chức năng
- 4.3. Kiểm thử liên quan đến sự thay đổi

4.1. Kiểm thử chức năng

- Qui trình cố gắng tìm ra các khác biệt giữa đặc tả bên ngoài của phần mềm và thực tế mà phần mềm cung cấp.
- Đặc tả bên ngoài của phần mềm là đặc tả chính xác về hành vi của phần mềm theo góc nhìn của người dùng thấy.
- Các loại kiểm thử chức năng:
 - Kiểm thử chức năng của hệ thống
 - Kiểm thử tích hợp dữ liệu và cơ sở dữ liệu
 - Kiểm thử vòng lặp công việc
 - Kiểm thử kiểm soát truy cập
 - Kiểm thử giao diện

4.1.1. Kiểm thử chức năng hệ thống

- Mục tiêu của loại kiểm thử này là đảm bảo đúng mục tiêu của kiểm thử chức năng: nhập dữ liệu- xử lý- lấy và kiểm tra kết quả trả về
- Kiểm tra sản phẩm phần mềm và các hoạt động của các chức năng bên trong sản phẩm đó bằng cách tương tác thông qua giao diện người dùng của sản phẩm
- Phân tích kết quả trả về

4.1.2. Kiểm thử giao diện

- Mục tiêu: kiểm tra giao diện của các chức năng trong một sản phẩm hệ thống hoạt động so với thiết kế
- Kiểm thử giao diện cần kiểm thử:
 - Liên kết hay chuyển tiếp
 - Cách thức truy cập(sử dụng phím tab, chuột...)
 - Kiểm tra các đối tượng trên màn hình:
 - Màu sắc
 - Vị trí
 - Kích thước chữ
 - Kiểu đối tượng
 -

4.1.3. Kiểm thử tích hợp dữ liệu và CSDL

- Kiểm tra các chức năng của một sản phẩm hay hệ thống phần mềm hoạt động đúng không sau khi sản phẩm pm đó đã có sự tích hợp hay đưa dữ liệu cũ, dữ liệu đã có sẵn từ bên ngoài vào sản phẩm.
- Đảm bảo các chức năng của hệ thống mới sử dụng được các dữ liệu cũ

4.1.4. Kiểm thử vòng lặp công việc

- Đảm bảo hoạt động của các công việc được chạy tự động theo lịch đã đặt trước không do người dùng tác động

4.1.5. Kiểm thử kiểm soát truy cập

- Đảm bảo các tác nhân, người sử dụng chỉ có thể truy cập vào đúng chức năng họ được phép truy cập
- Đảm bảo chỉ những người dùng được phân quyền truy cập hệ thống mới có thể truy cập vào hệ thống và thông qua các gateway thích hợp



	Navigate Pages	Draw/Markup	Import Content Sync Rights	Delete Pages & other users content	Permission Controls	ViewBook Settings
★ Owner	✓	✓	✓	✓	✓	✓
★ Moderator	✓	✓	✓	✓	✓	
✎ Collaborator	✓	✓	✓			
🔍 Reviewer	✓	✓				
👁 Viewer	✓*					

* A viewer can only navigate pages if Sync Mode is OFF.

4.2. Kiểm thử phi chức năng

- Tập trung vào kiểm thử sản phẩm, hệ thống phần mềm cần kiểm thử có những đặc tính tốt như thế nào (how well)
- Kiểm thử phi chức năng có thể được sử dụng ở mọi cấp độ kiểm thử nhưng thường được sử dụng hiệu quả nhất trong cấp độ kiểm thử hệ thống và kiểm thử chấp nhận sản phẩm.
- Các loại kiểm thử phi chức năng (4 loại thường dùng)
 - Kiểm thử hiệu năng (performance testing)
 - Kiểm thử tải trọng (load testing)
 - Kiểm thử tập trung (stress testing)
 - Kiểm thử với lượng dữ liệu lớn (volume testing)

4.2.1. Kiểm thử hiệu năng

- Bảo đảm tối ưu việc phân bổ tài nguyên hệ thống (ví dụ bộ nhớ) nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng câu truy vấn...
- Thí dụ:
 - ☐ trình chiếu phim full HD không chiếu kịp 20 frame/sec.
 - ☐ trình nén dữ liệu không nén dữ liệu kịp với tốc độ đề ra.
 - ☐ trình soạn thảo văn bản không nhận và xử lý kịp các ký tự được nhập bởi người dùng.
 - ☐ trình ghi DVD không tạo dữ liệu ghi kịp tốc độ mà ổ DVD yêu cầu...

4.2.2. Kiểm thử tải trọng

- Kiểm thử tải trọng (kiểm thử đồng thời): tập trung vào xác định đặc tính hiệu suất của hệ thống hay sản phẩm phần mềm trong điều kiện tải hay upload cụ thể
- Bảo đảm hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người truy xuất cùng lúc).
- Ví dụ: Hệ thống hỗ trợ cho việc truy cập, giải quyết 3000 yêu cầu trong một ngày, đáp ứng việc sử dụng của 350 người sử dụng đồng thời từ 9h30 đến 11h am

4.2.3. Kiểm thử tập trung

- Stress Test tập trung vào các trạng thái tới hạn, các "điểm chết", các tình huống bất thường như đang giao dịch thì ngắt kết nối (xuất hiện nhiều trong kiểm tra thiết bị như POS, ATM...)...

4.2.3. Kiểm thử với lượng dữ liệu lớn

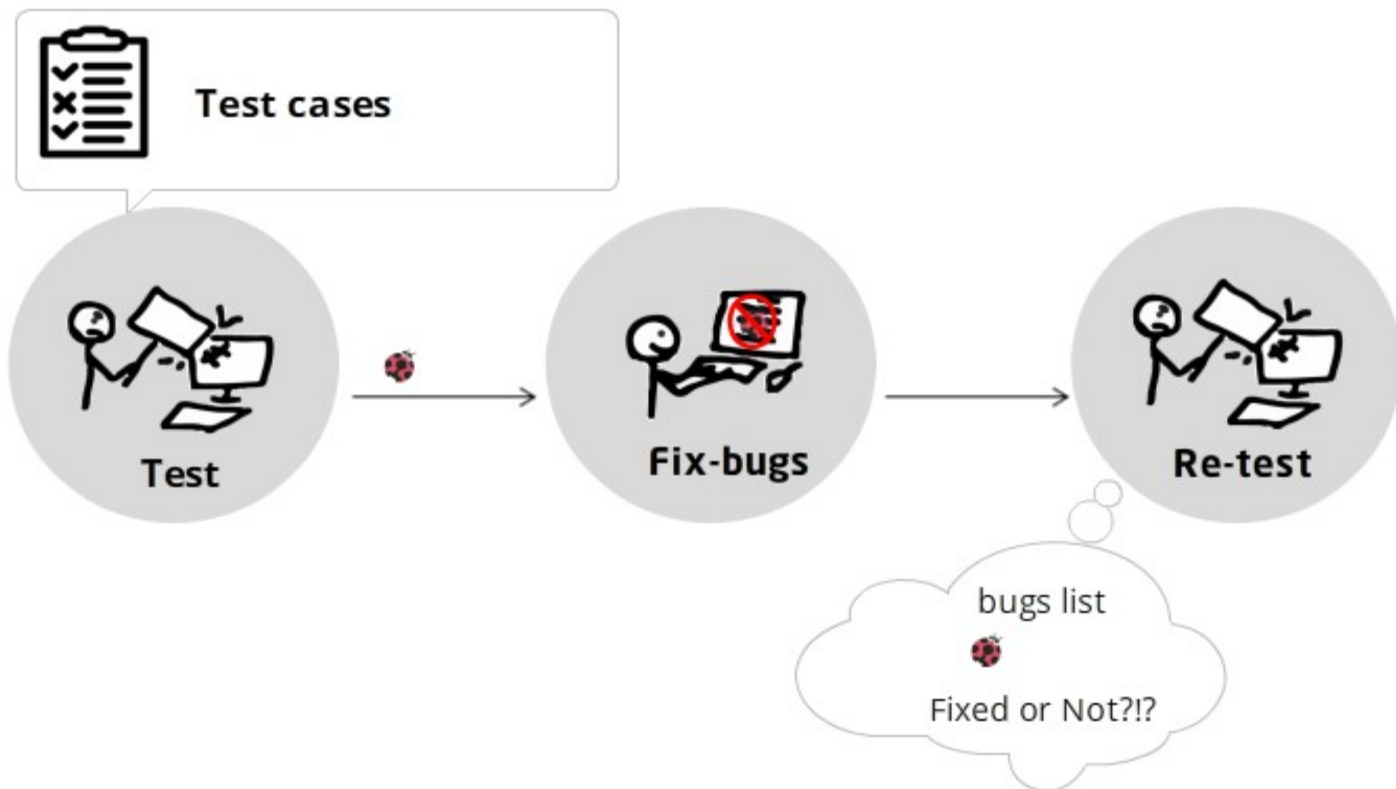
- Kiểm thử tập trung vào việc xác định hoặc xác nhận đặc tính hiệu suất của hệ thống hoặc ứng dụng được kiểm thử trong điều kiện hệ thống có lượng dữ liệu rất lớn
- Dữ liệu lớn có thể là cơ sở dữ liệu lớn hoặc dữ liệu trong file upload lên hệ thống có dung lượng lớn

4.4. Kiểm thử liên quan đến sự thay đổi

- Thực hiện hoạt động kiểm thử khi có sự thay đổi trên hoặc trong sản phẩm phần mềm.
- Sự thay đổi của sản phẩm phần mềm có thể là:
 - Sửa chữa các lỗi tìm được
 - Sản phẩm được nâng cấp, được thay đổi về chức năng
- Các loại kiểm thử liên quan đến sự thay đổi:
 - Kiểm thử lại
 - Kiểm thử hồi quy

4.4.1. Kiểm thử lại (kiểm thử xác nhận)

- Chỉ kiểm thử những test case chưa pass



4.4.1. Kiểm thử lại (kiểm thử xác nhận)

- Khi thực hiện kiểm thử xác nhận cần chú ý:
- Thực hiện kiểm thử đúng các bước như trong mô tả trường hợp kiểm thử:
 - Đúng các tập đầu vào
 - Đúng các dữ liệu
 - Đúng môi trường kiểm thử

4.4.2. Kiểm thử hồi quy

- Kiểm thử hồi quy để đảm bảo rằng những thay đổi mới không làm ảnh hưởng đến những phần đã hoàn thiện trước đó
- Kiểm thử hồi quy thường được thực hiện tự động

Have requirement changes



Regression Testing



- After the Software has changed
- When the Environment has changed

Validation of change

4.4.2. Kiểm thử hồi quy

- Các phép thử hồi quy được chia làm 3 loại:
 - **Các phép thử đại diện:** thực hiện tất cả chức năng của pm
 - **Các phép thử bổ sung:** tập trung vào chức năng dễ bị ảnh hưởng nhất khi có thay đổi
 - **Các phép thử tập trung:** tập trung vào thành phần pm bị thay đổi



Bài 5. Các kỹ thuật kiểm thử

- 5.1. Kiểm thử hộp trắng
- 5.2. Kiểm thử hộp đen

5.1. Kiểm thử hộp trắng

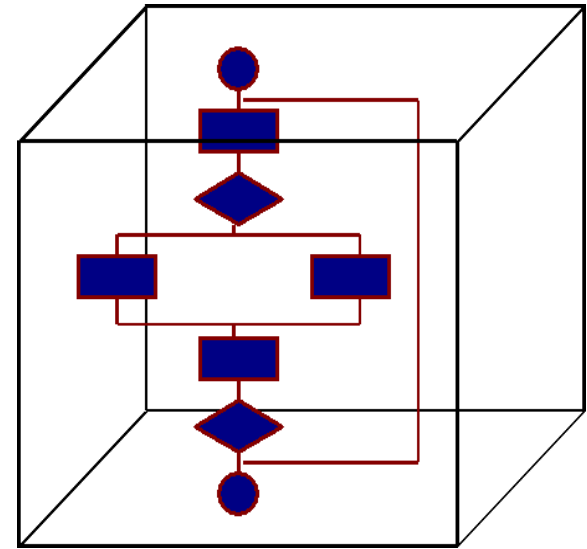
5.1.1. Khái niệm kiểm thử hộp trắng

5.1.2. Kiểm thử luồng điều khiển

5.1.1. Khái niệm

Kiểm thử hộp trắng:

- Kiểm tra các đoạn mã chương trình xem nó có vận hành đúng như thiết kế hay không.
- Kiểm thử hộp trắng dựa trên việc xem xét cấu trúc bên trong của chương trình theo cấu trúc điều khiển và hoạt động của chúng



5.1.1. Khái niệm (t)

- Các tên gọi khác: kiểm thử cấu trúc (structural testing), kiểm thử hộp kính (glass box), kiểm thử rõ ràng (clear box testing).
- Đối tượng chính của kiểm thử hộp trắng là tập trung vào cấu trúc bên trong chương trình và tìm ra tất cả những lỗi bên trong chương trình.
- Việc kiểm tra tập trung chủ yếu vào:
 - Cấu trúc chương trình: Những câu lệnh và các nhánh, các loại đường dẫn chương trình.
 - Logic bên trong chương trình và cấu trúc dữ liệu.
 - Những hành động và trạng thái bên trong chương trình.

Ưu, nhược điểm của kiểm thử hộp trắng

- **Ưu điểm:**

- Khi sử dụng kiểm thử hộp trắng, kiểm thử viên có thể chắc chắn rằng mọi đường xuyên qua phần mềm cần kiểm thử đã được xác định và kiểm thử.

Ưu, nhược điểm của kiểm thử hộp trắng

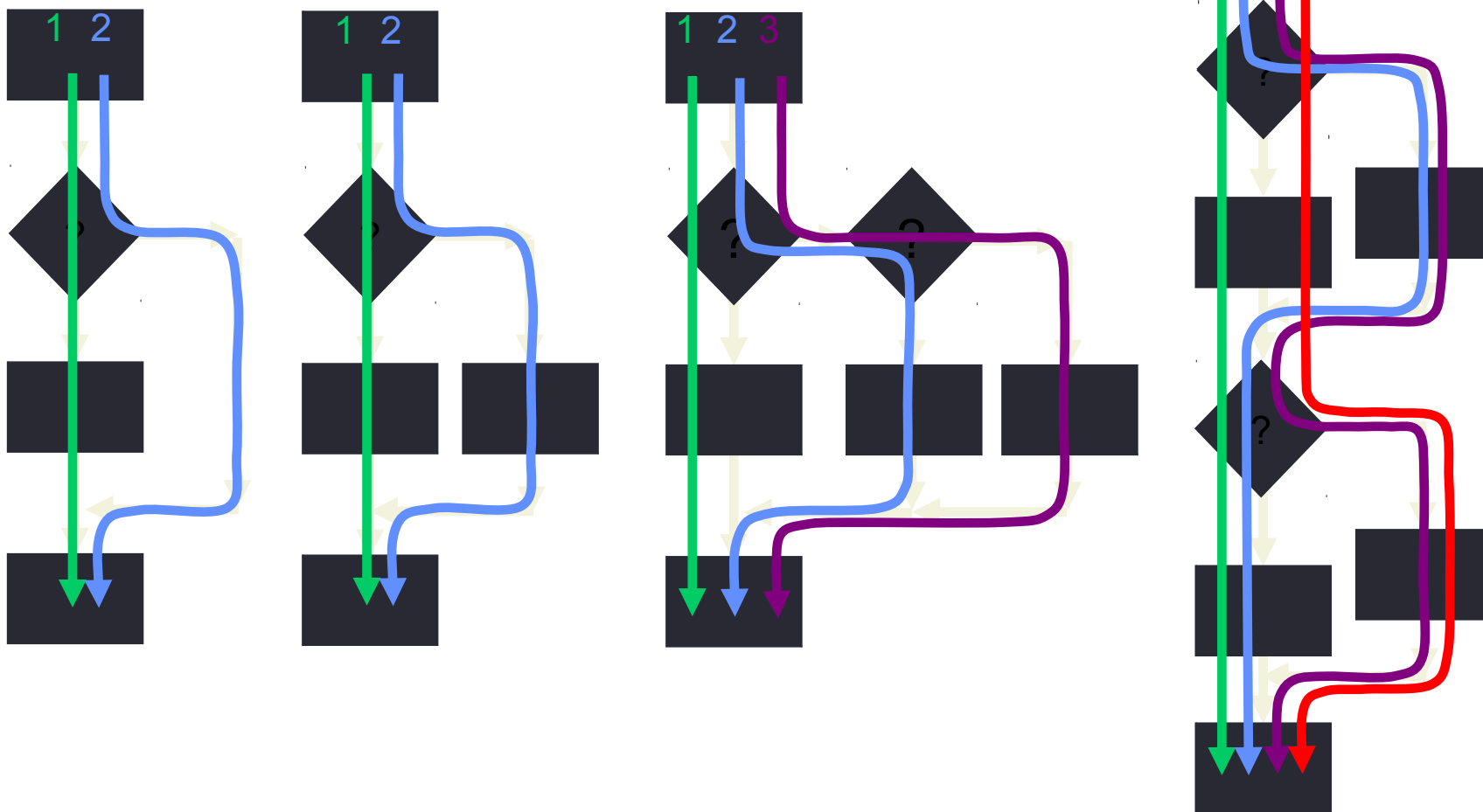
- **Nhược điểm:**

1. Không đủ khả năng kiểm thử hết các đường thi hành vì số lượng quá nhiều
2. Kiểm thử bằng hộp trắng không thể đảm bảo rằng chương trình đã tuân theo đặc tả
3. Không phát hiện ra chương trình sai do thiếu đường dẫn
4. Không phát hiện được lỗi do sai dữ liệu
5. Kiểm thử viên cần có các kỹ năng về lập trình để hiểu và đánh giá được phần mềm. Không may là hiện nay có nhiều kiểm thử viên không có được nền tảng tốt về lập trình

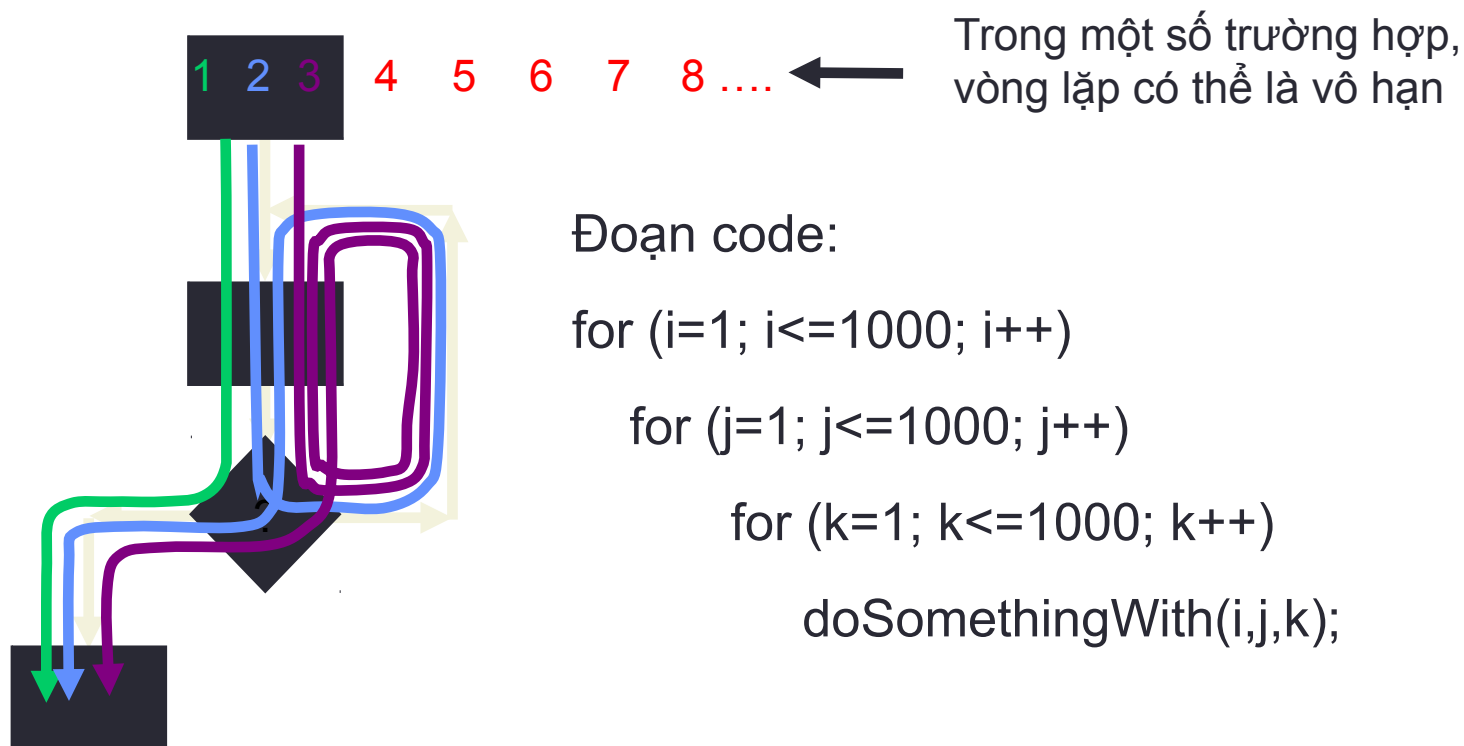
Khái niệm đường thi hành

- Đường thi hành (Execution path):
 - Là 1 kịch bản thi hành đơn vị phần mềm tương ứng
 - Là danh sách có thứ tự các lệnh được thi hành ứng với 1 lần chạy cụ thể của đơn vị phần mềm, bắt đầu từ điểm nhập của đơn vị phần mềm đến điểm kết thúc của đơn vị phần mềm.

Đường thi hành



Đường thi hành có vòng lặp



chỉ có 1 đường thi hành, nhưng rất dài : dài $1000 \times 1000 \times 1000 = 1$ tỉ lệnh gọi hàm `doSomething(i,j,k)` khác nhau.

Đường thi hành

- Cho dù có kiểm thử hết được toàn bộ các đường thi hành thì vẫn không thể phát hiện những đường thi hành cần có nhưng không (chưa) được hiện thực :

Vd:

```
if (a>0) dolsGreater();
```

```
if (a==0) dolsEqual();
```

```
// thiếu việc xử lý trường hợp  $a < 0$  - if (a<0) dolsLess();
```

Đường thi hành

Một đường thi hành đã kiểm tra là đúng nhưng vẫn có thể bị lỗi khi dùng thật.

Vd:

```
int phanso(int a, int b) {  
    return a/b;  
}
```

Chạy đúng với mọi giá trị trừ $b=0$

Đồ thị lưu trình

Là một trong các phương pháp mô tả thuật giải

Mô tả các cấu trúc điều khiển phổ dụng:

Câu lệnh

If...then

If...then...else

While...do

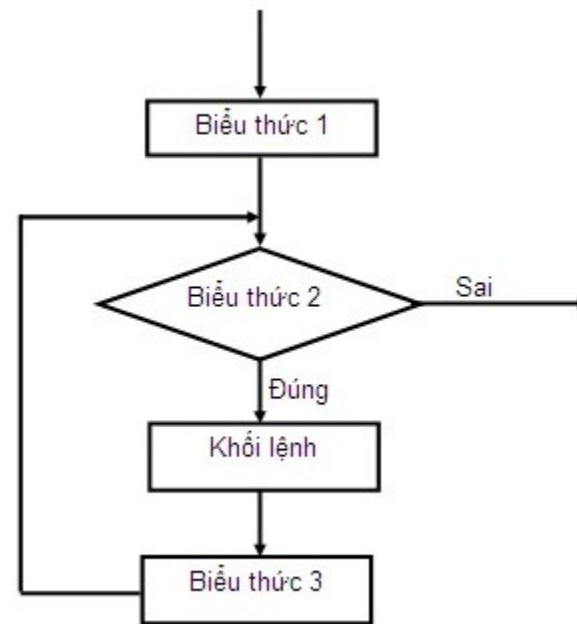
Do...while

For...

Switch...

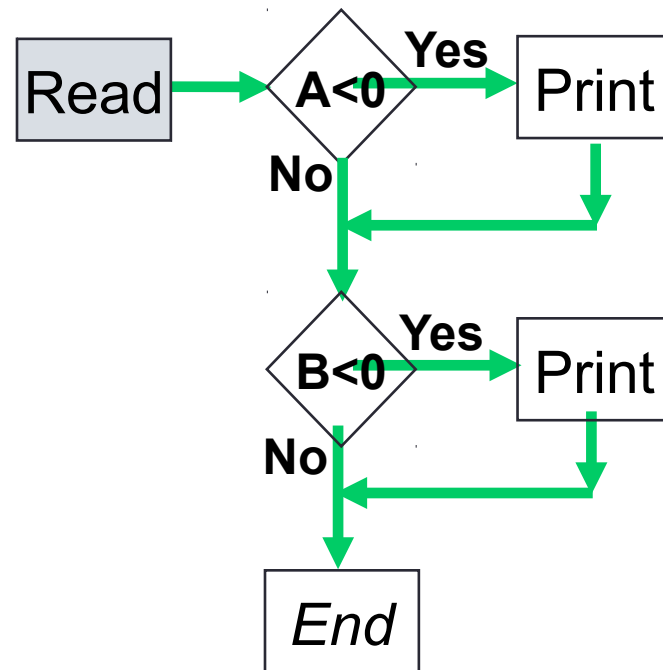
Đồ thị lưu trình

- For(biểu thức 1, biểu thức 2, biểu thức 3)
- { Khối lệnh }



Ví dụ 1:

```
Read A
Read B
IF A < 0 THEN
    Print "A negative"
ENDIF
IF B < 0 THEN
    Print "B negative"
ENDIF
```



Ví dụ 2

Wait for card to be inserted

IF card is a valid card THEN

display “Enter PIN number”

IF PIN is valid THEN

select transaction

ELSE (otherwise)

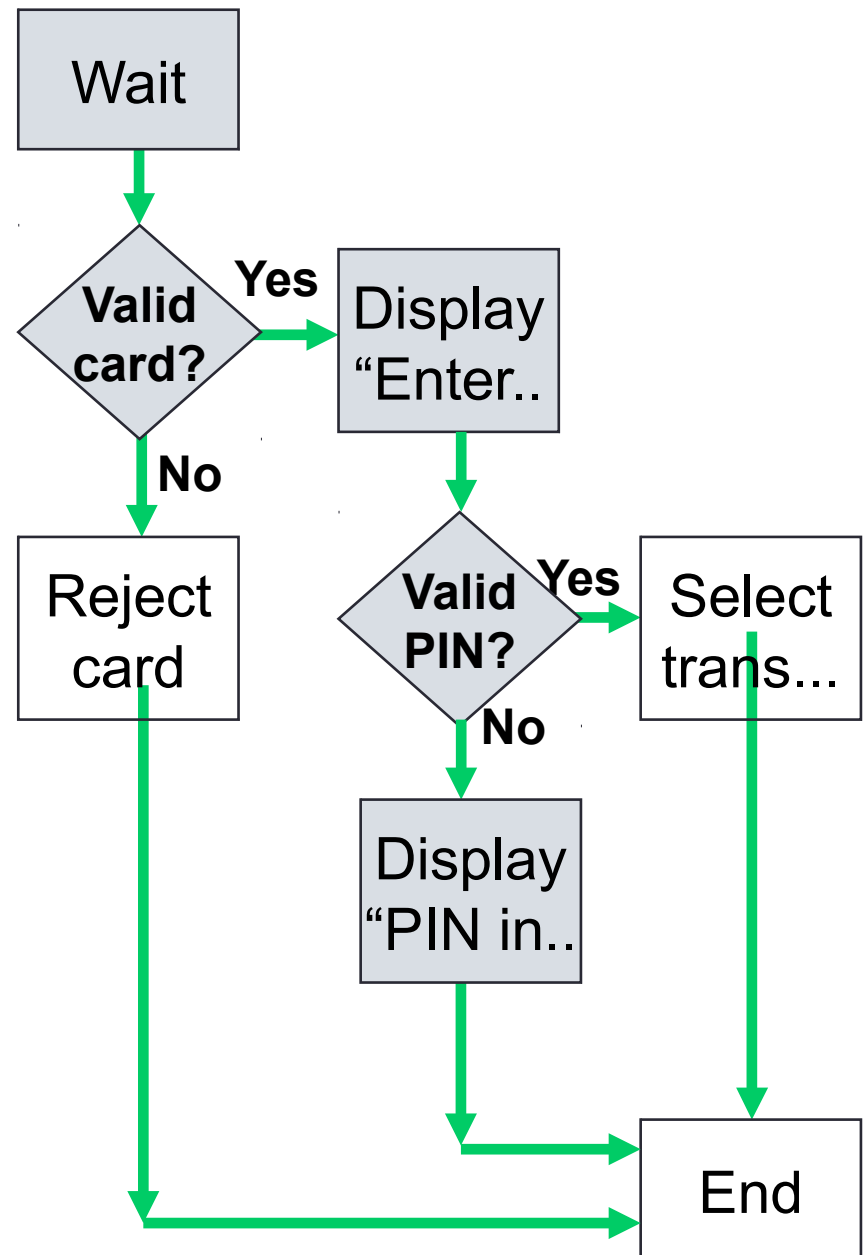
display “PIN invalid”

ELSE (otherwise)

reject card

End

Yêu cầu: Vẽ đồ thị lưu trình.



Ví dụ 3:

Wait to number to be entered

Enter number n

WHILE $n > 0$

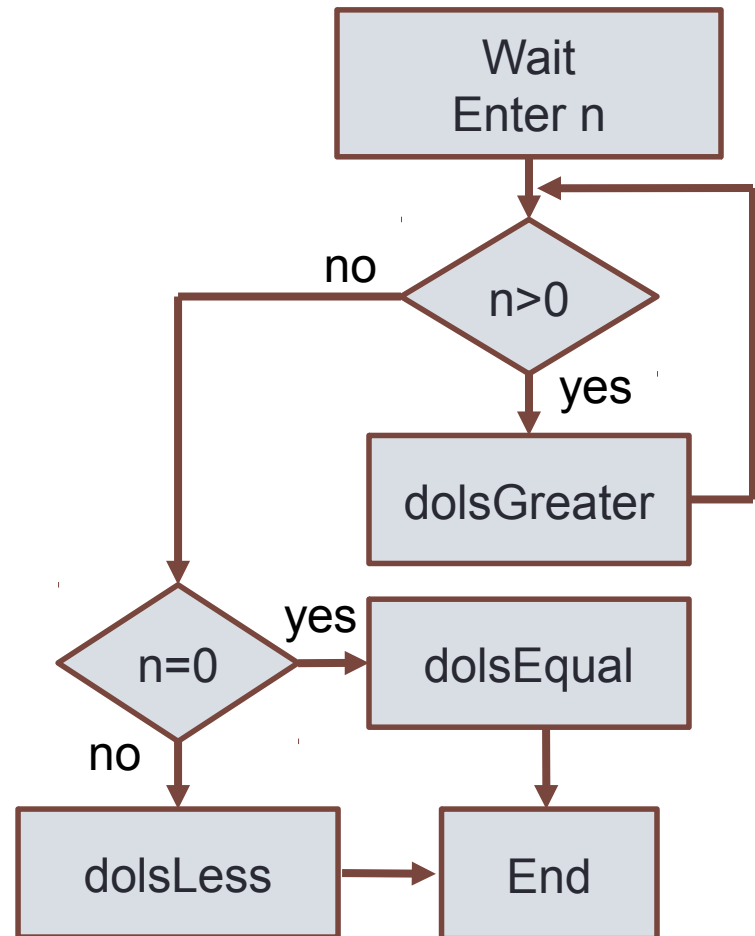
Do dolsGreater

IF $n = 0$ THEN dolsEqual

ELSE dolsLess

End.

Yêu cầu: Vẽ đồ thị lưu trình.



Đồ thị dòng

- Đồ thị dòng (flow graph) được Tom McCabe đề xuất năm 1976
- Đồ thị dòng được xây dựng từ đồ thị dòng điều khiển của chương trình bằng cách:
 - gộp các lệnh tuần tự liên tiếp thành một nút
 - thay các lệnh rẽ nhánh (độc lập) thành một nút
 - thay điểm hợp nhất của các lệnh rẽ nhánh bằng một nút.

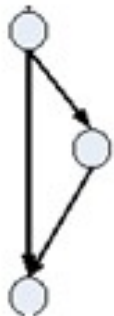
Đồ thị dòng (t)

- Cấu trúc của một đồ thị dòng bao gồm:
 - Mỗi nút (hình tròn) biểu thị một vài (có thể là 0) câu lệnh thủ tục.
 - Mỗi cạnh biểu diễn dòng điều khiển nối hai nút với nhau.
 - Miền là phần mặt phẳng giới hạn bởi các cung và nút của đồ thị.
 - Những nút biểu thị sự phân nhánh hoặc hội nhập của các cung được gọi là nút vị từ.

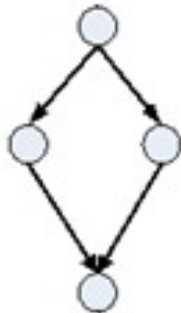
Đồ thị dòng (t)

- Cấu trúc cơ bản của đồ thị dòng

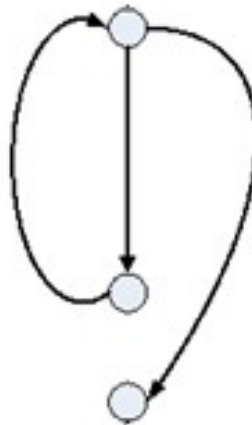
If



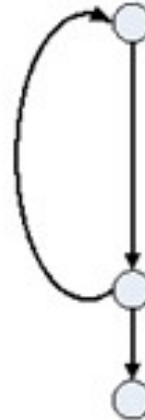
If- then-
else



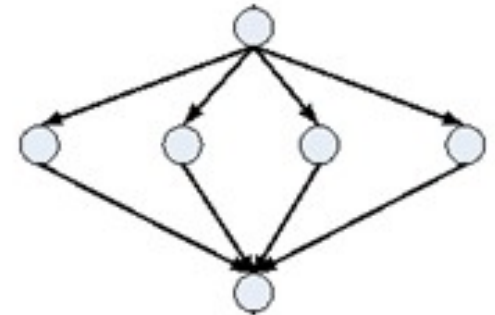
For or
while



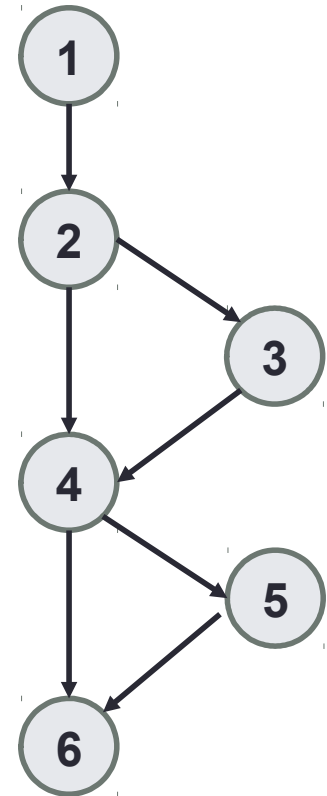
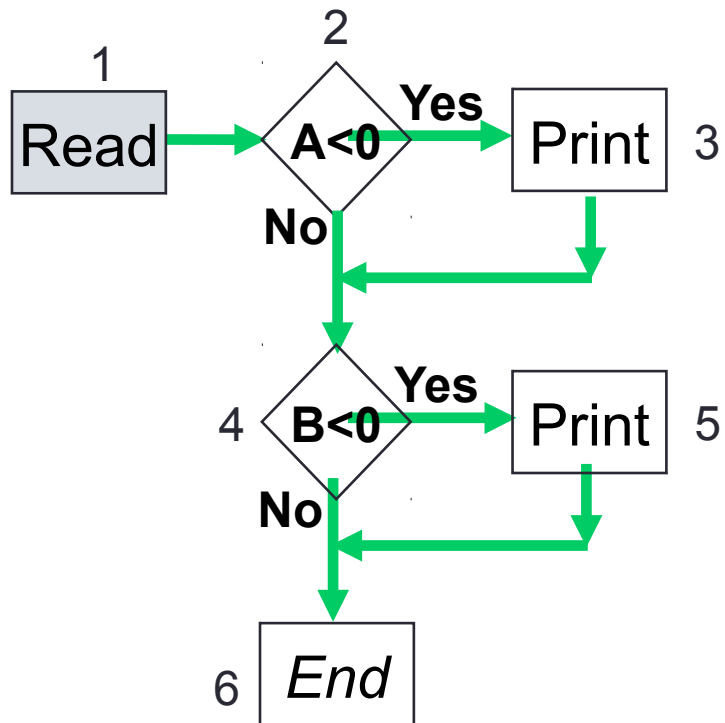
Do- while



Switch
(Case)



Chuyển đổi đồ thị lưu trình – Đồ thị dòng



Đồ thị dòng (t)

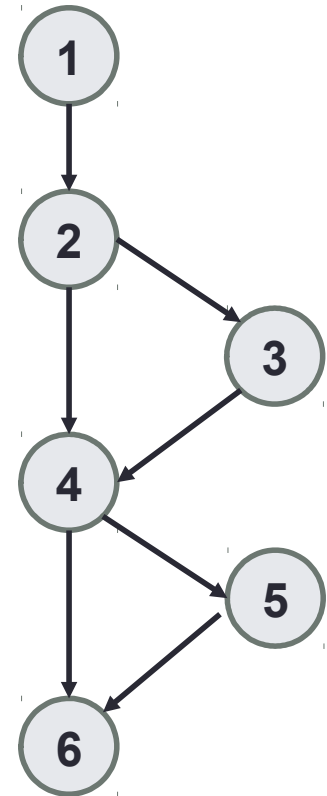
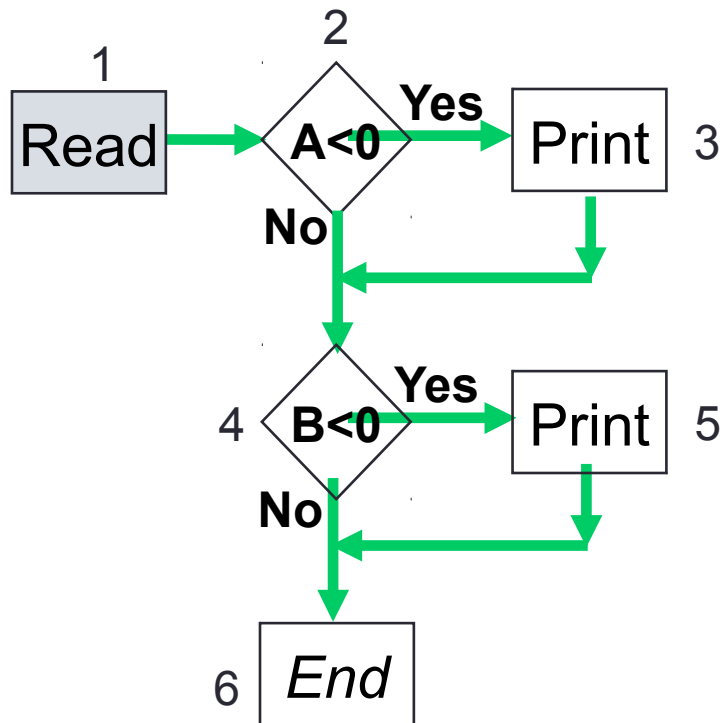
- **Độ phức tạp chu trình**

- Là số đo sự phức tạp logic của chương trình.
- Là số các đường đi độc lập cơ bản trong tập các con đường độc lập của một chương trình.
- Là số đường độc lập nhỏ nhất phủ hết các cạnh của đồ thị luồng.
- Số đo này là giới hạn trên của số ca kiểm thử cần phải tiến hành để đảm bảo rằng, tất cả các câu lệnh trong chương trình đều được thực hiện ít nhất một lần.

Độ phức tạp của chu trình

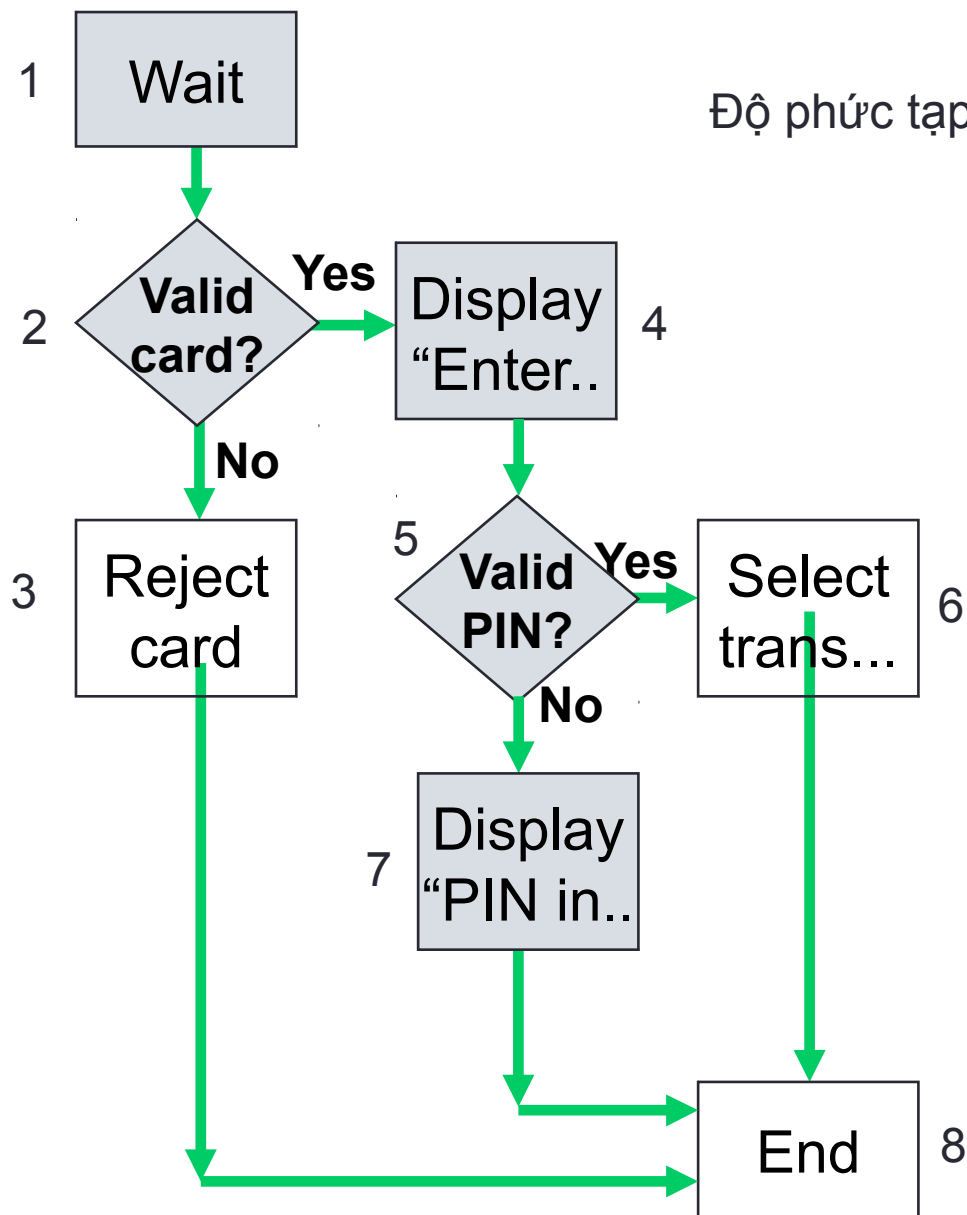
- Độ phức tạp Cyclomatic $C = V(G)$ của đồ thị dòng điều khiển được tính bởi 1 trong các công thức sau :
- $\square V(G) = E - N + 2$, trong đó E là số cung, N là số nút của đồ thị.
- $\square V(G) = P + 1$, P là số nút quyết định
- Độ phức tạp của chu trình (Cyclomatic) C chính là **số đường thi hành tuyến tính độc lập** của TPPM cần kiểm thử.

3.4. Độ phức tạp của chu trình

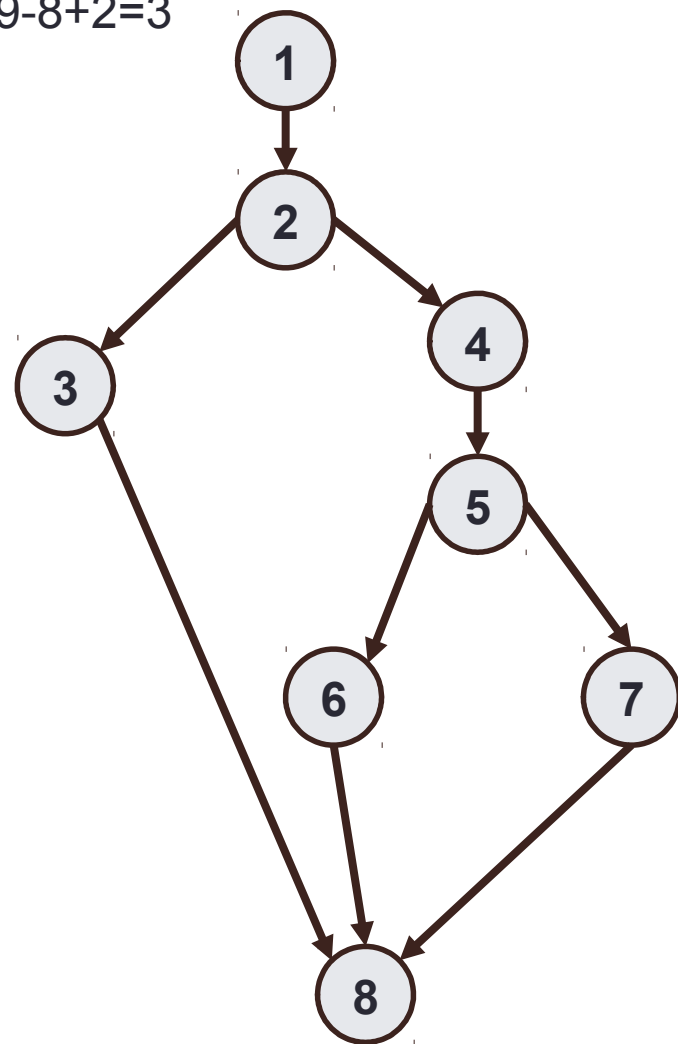


Độ phức tạp của chu trình = $7 - 6 + 2 = 3$

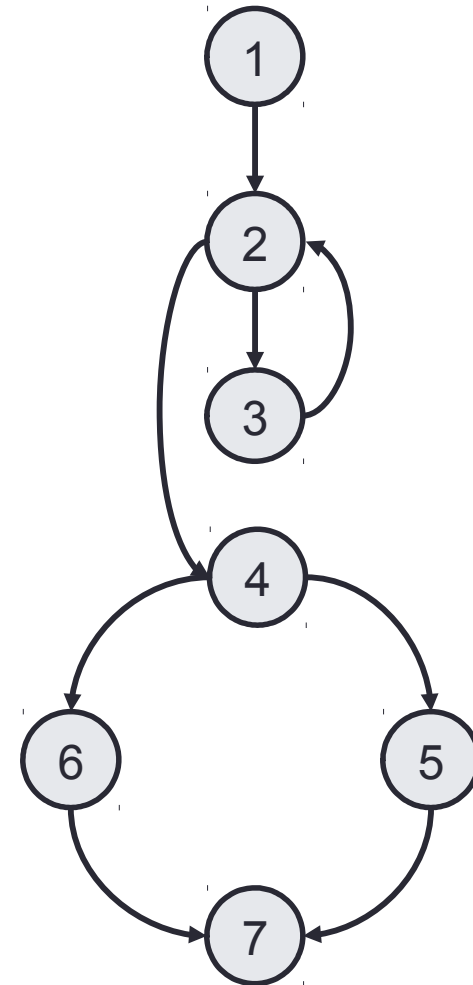
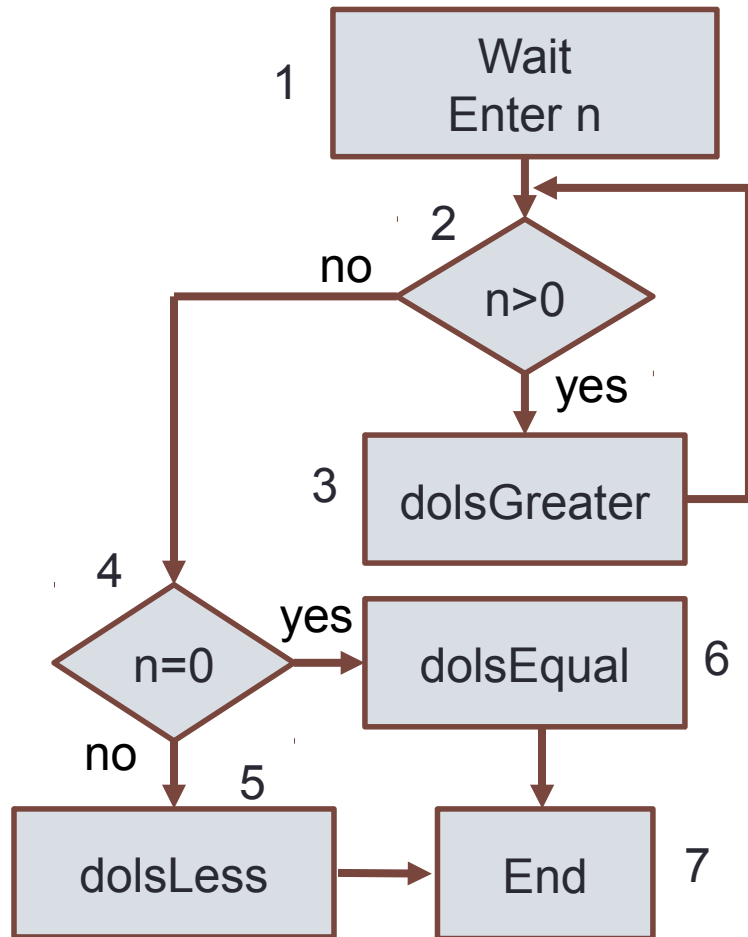
Chuyển sang đồ thị dòng và tính độ phức tạp của chu trình



Độ phức tạp $C = 9 - 8 + 2 = 3$



Độ phức tạp của chu trình $C = 8 - 7 + 2 = 3$



Các cấp bao phủ kiểm thử

Phủ kiểm thử (coverage): tỉ lệ các thành phần thực sự được kiểm thử so với tổng thể các thành phần.

- Các thành phần bao gồm: lệnh thực thi, điểm quyết định, điều kiện con hay sự kết hợp của chúng.
- Độ phủ càng lớn thì độ tin cậy càng cao.

3.3. Các cấp bao phủ kiểm thử

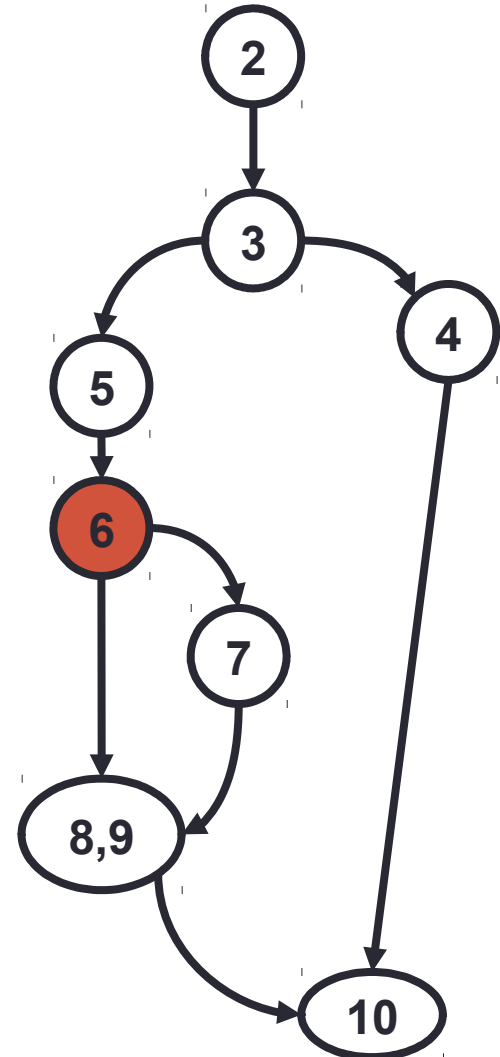
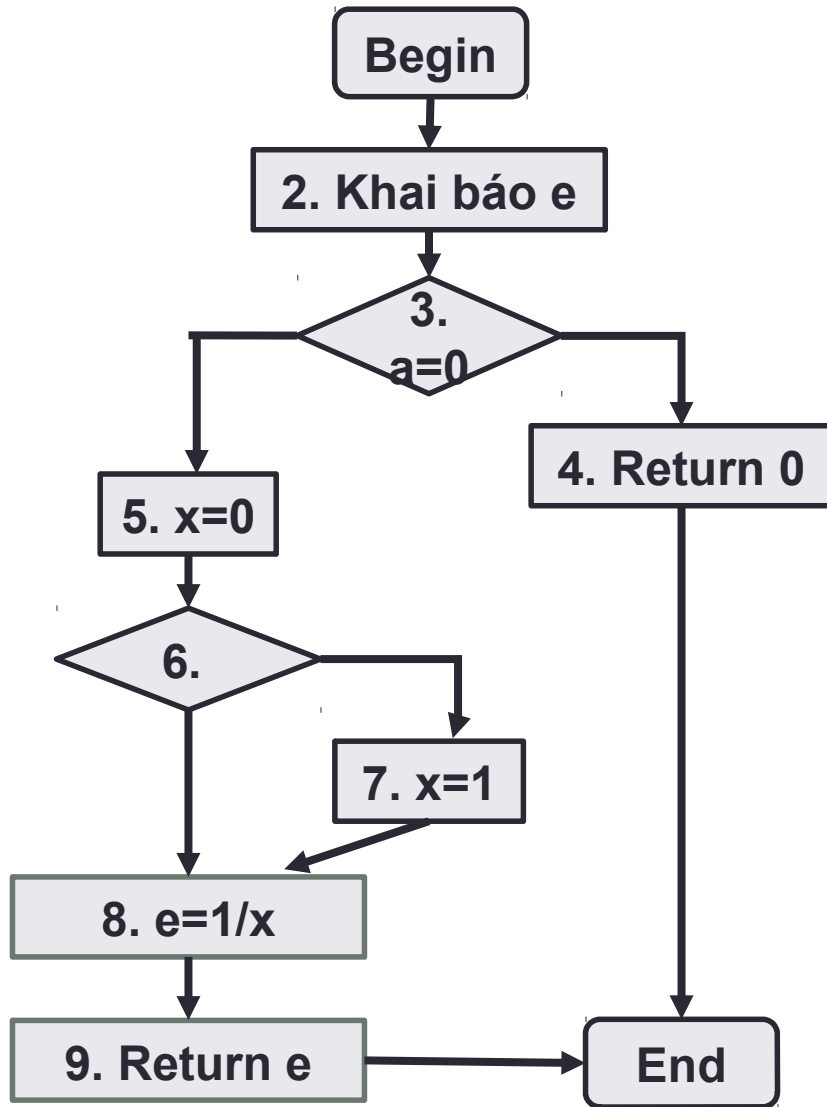
- Phủ cấp 0: kiểm thử những gì có thể kiểm thử được, phần còn lại để người dùng phát hiện và báo lại sau. Đây là kiểm thử không có trách nhiệm
- Phủ cấp 1: **Bao phủ câu lệnh (statement coverage)**: Các câu lệnh được thực hiện ít nhất 1 lần
- Phủ cấp 2: **Bao phủ nhánh (branch coverage)**: tại các điểm quyết định thì các nhánh đều được thực hiện ở cả hai phía T,F
- Phủ cấp 3: **Bao phủ điều kiện(condition coverage)**: Các điều kiện con của các điểm quyết định được thực hiện ít nhất 1 lần
- Phủ cấp 4: **Kết hợp phủ nhánh và điều kiện (branch & condition coverage)**

3.3. Các cấp bao phủ kiểm thử

Xem đoạn code sau đây:

```
1 float foo(int a, int b, int c, int d) {  
2     float e;  
3     if (a==0)  
4         return 0;  
5     int x = 0;  
6     if ((a==b) || (c==d) && bug(a))  
7         x = 1;  
8     e = 1/x;  
9     return e;  
10 }
```

Vd: Đồ thị dòng



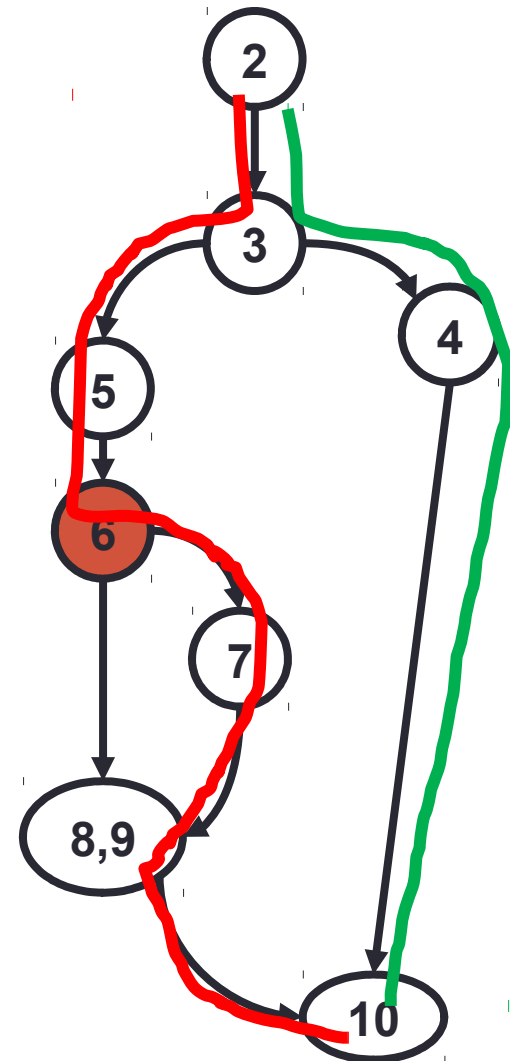
Bao phủ cấp 1

Với hàm foo trên, ta chỉ cần 2 test case là đạt 100% phủ cấp 1 :

1. foo(0,0,0,0), trả về 0
2. foo(1,1,1,1), trả về 1

Bao phủ cấp 1 không phát hiện lỗi chia 0 ở hàng lệnh 8.

Bao phủ cấp 1 = bao phủ dòng lệnh



Bao phủ cấp 2

Line	Predicate	True	False
3	(a == 0)	Test Case 1 foo(0, 0, 0, 0) return 0	Test Case 2 foo(1, 1, 1, 1) return 1
6	((a==b) OR ((c == d) AND bug(a)))	Test Case 2 foo(1, 1, 1, 1) return 1	Test Case 3 foo(1, 2, 1, 2) division by zero!

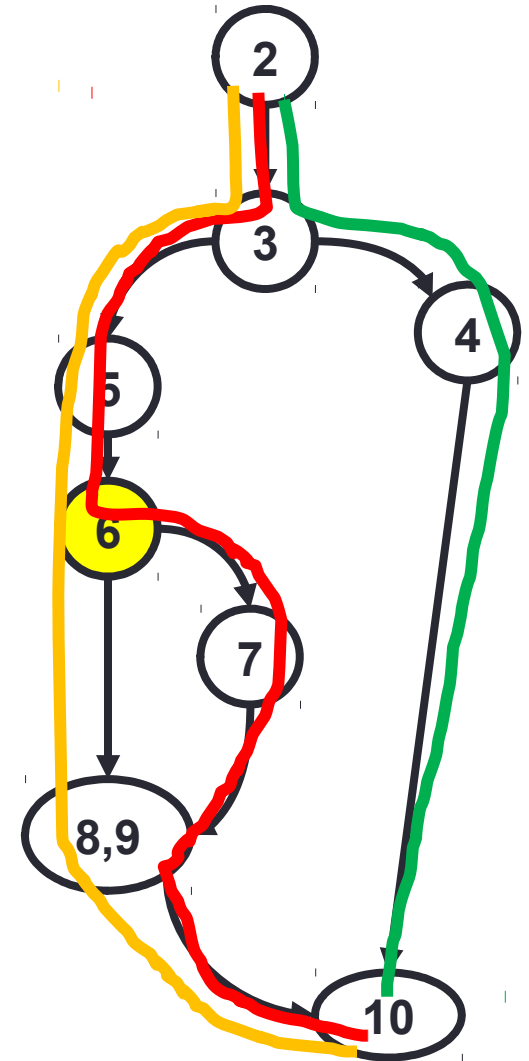
Với 2 test case xác định trong slide trước, ta chỉ đạt được $3/4 = 75\%$ phủ các nhánh.

Nếu thêm test case 3 :

3. foo(1,2,1,2), thì mới đạt 100% phủ các nhánh.

Bao phủ cấp 2

- Điểm quyết định số 3 đã đi theo đủ 2 nhánh.
- Bao phủ cấp 1, điểm quyết định số 6 mới đi 1 nhánh,
- Đi theo nhánh còn lại của điểm quyết định số 6 sẽ đạt bao phủ cấp 2
- **Bao phủ cấp 2= bao phủ nhánh**
- Tất cả các điểm quyết định đều được kiểm tra ở cả 2 nhánh True và False



Bao phủ cấp 3

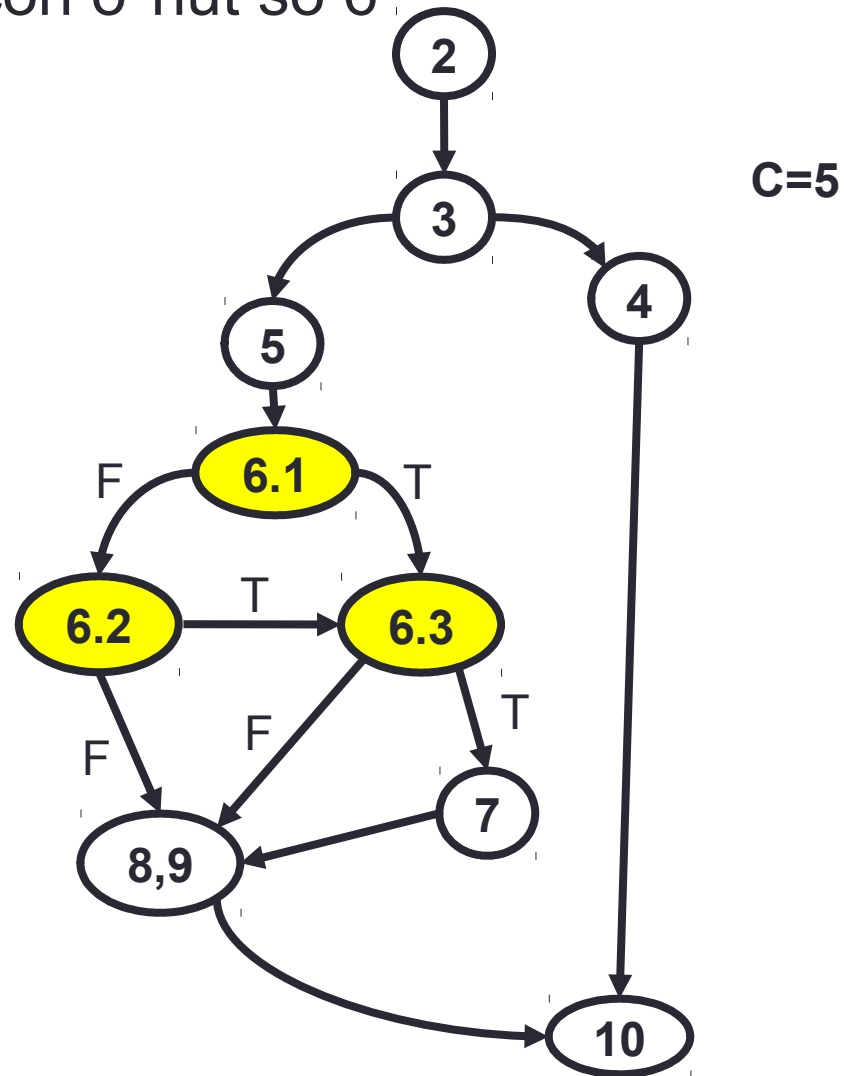
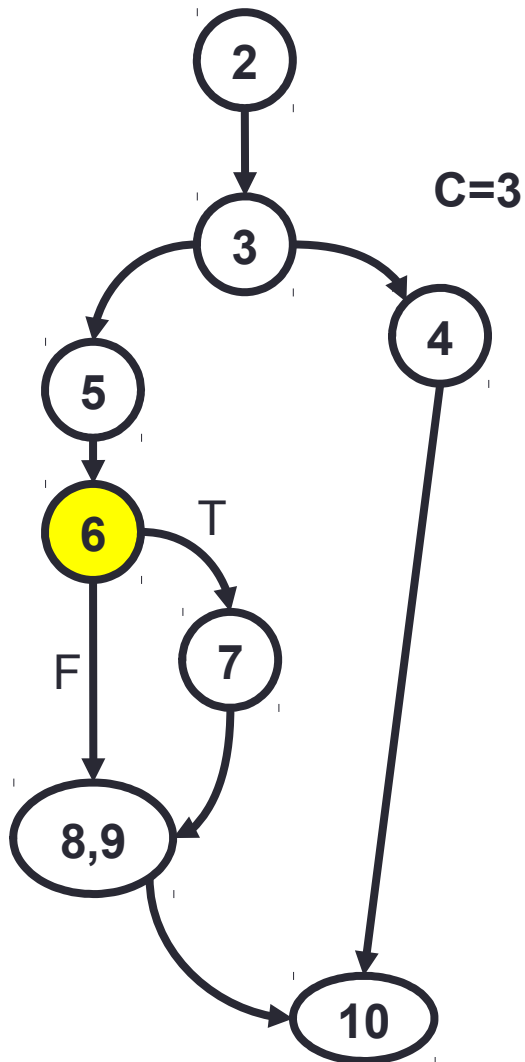
- Mỗi điều kiện con của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho cả 2 nhánh true và false
- **Bao phủ cấp 3= bao phủ điều kiện**
- Phủ các điều kiện con chưa chắc đảm bảo phủ các nhánh và ngược lại

Bao phủ cấp 3

Predicate	True	False
<code>a == 0</code>	TC 1 : <code>foo(0, 0, 0, 0)</code> return 0	TC 2 : <code>foo(1, 1, 1, 1)</code> return 1
<code>(a == b)</code>	TC 2 : <code>foo(1, 1, 1, 1)</code> return value 0	TC 3 : <code>foo(1, 2, 1, 2)</code> division by zero!
<code>(c == d)</code>	TC 4 : <code>foo(1, 2, 1, 1)</code> Return 1	TC 3 : <code>foo(1, 2, 1, 2)</code> division by zero!
<code>bug(a)</code>	TC 4 : <code>foo(1, 2, 1, 1)</code> Return 1	TC 5 : <code>foo(2, 1, 1, 1)</code> division by zero!

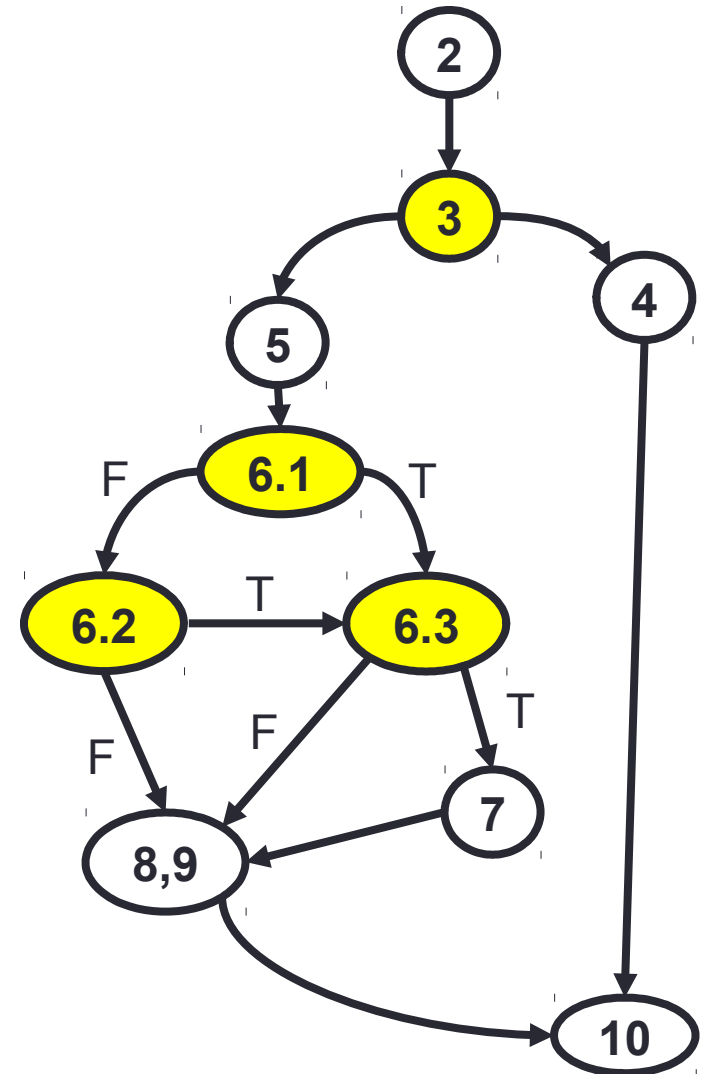
Bao phủ cấp 4

- Phân tích các điều kiện con ở nút số 6



Bao phủ cấp 4

- Kiểm thử sao cho mỗi điều kiện con của từng điểm quyết định được thực hiện ít nhất 1 lần cho trường hợp T lẫn F và điểm quyết định cũng phải được thực hiện cho cả hai nhánh T và F
- Đây là mức kiểm thử tốt nhất trên thực tế.

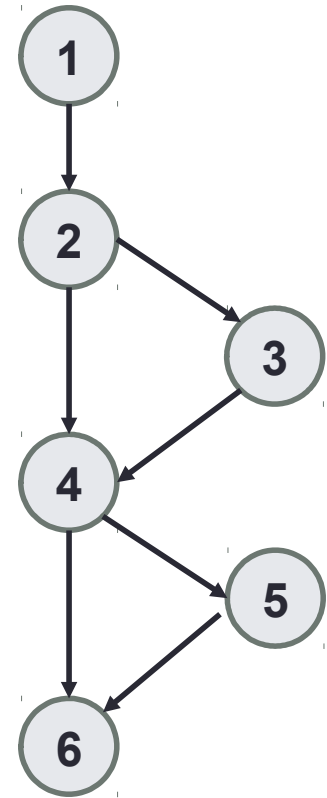


Xác định các đường thi hành tuyến tính độc lập

- Dựa trên đồ thị dòng và độ phức tạp của chu trình.
 - 1) Xác định đường tuyến tính đầu tiên bằng cách đi dọc theo 1 nhánh bất kì (việc chọn này phụ thuộc vào quan điểm của người làm kiểm thử).
 - 2) Dựa trên đường cơ sở vừa chọn, thay đổi cung xuất của nút quyết định đầu tiên và giữ tối đa phần còn lại
 - 3) Bắt đầu với đường cơ sở chọn lần 1, thay đổi cung xuất của nút quyết định số 2 và giữ tối đa phần còn lại.
 - 4) Tiếp tục thay đổi cung xuất cho từng nút quyết định trên đường cơ sở để xác định các đường tiếp theo cho đến khi không còn nút quyết định nào nữa.
 - 5) Lặp chọn tuần tự từng đường tìm được làm đường chính để xác định các đường mới cho tới khi đủ số C

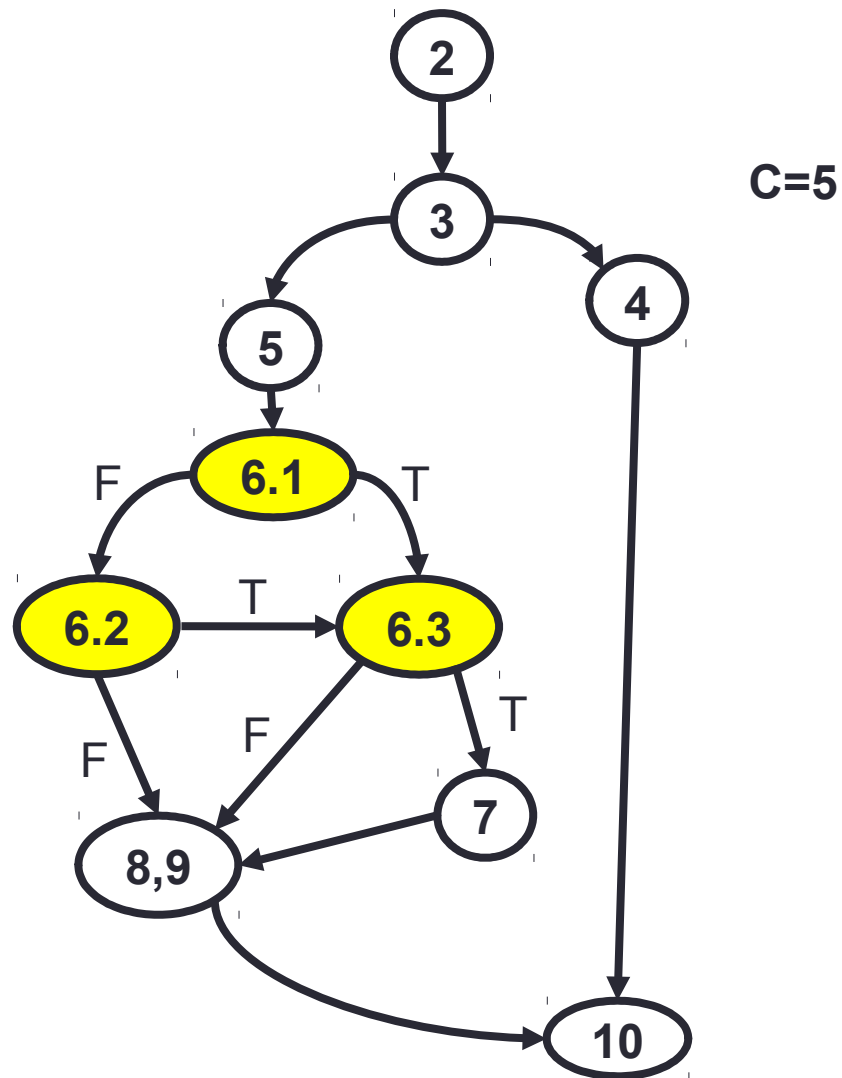
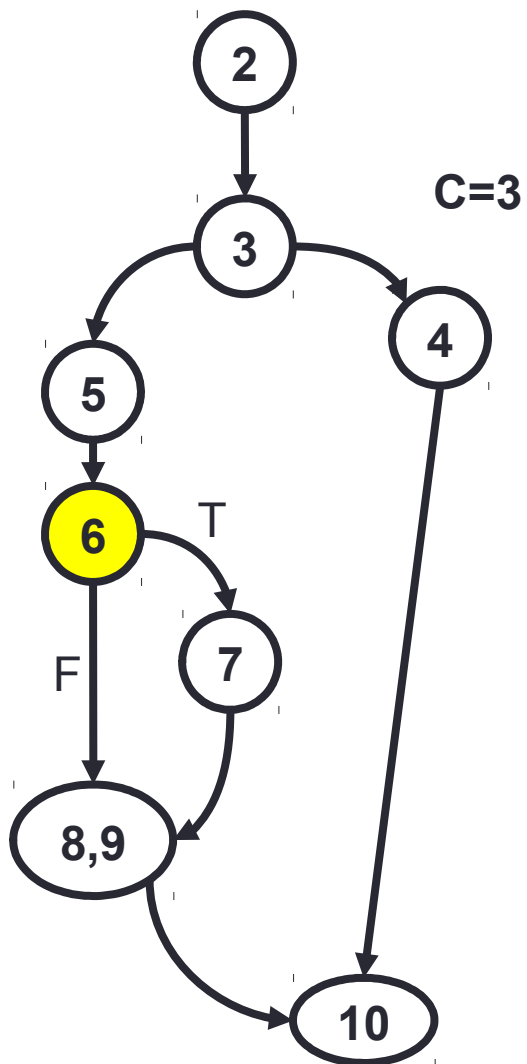
Vd: Xác định các đường độc lập tuyến tính cơ bản

- 1-2-3-4-5-6
- 1-2-4-5-6
- 1-2- 3-4-6



Độ phức tạp của chu trình = $7 - 6 + 2 = 3$

Bài tập, xác định các đường cơ bản



Kiểm thử đường thi hành cơ bản

- **B1:** Vẽ đồ thị lưu trình hoặc đồ thị dòng
- **B2:** Xác định độ phức tạp Cyclomat
- **B3:** Xác định 1 tập các đường dẫn cơ sở độc lập
- **B4:** Thiết kế các trường hợp kiểm thử với các đường dẫn độc lập
- **B5:** Thực hiện kiểm thử trên từng test case

Ví dụ

Read A

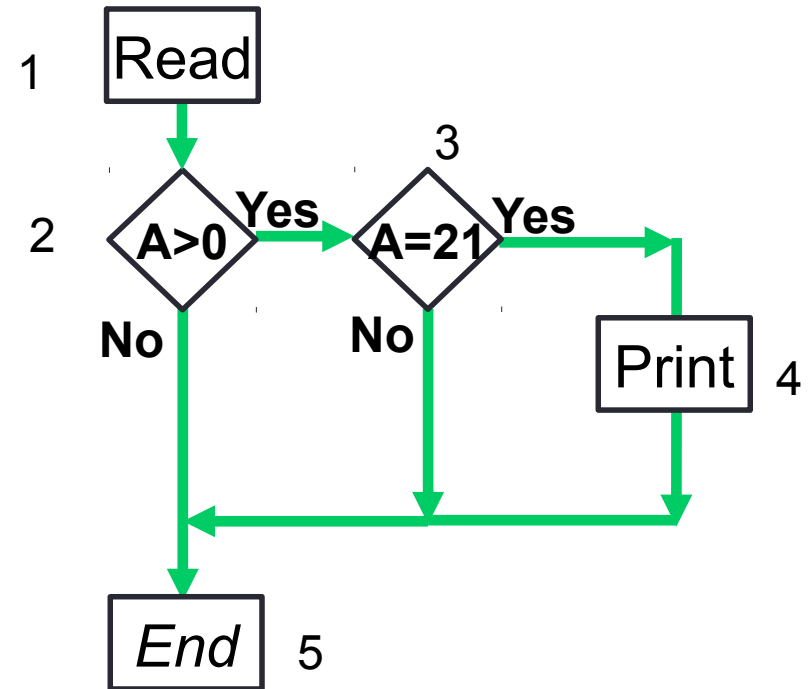
IF A > 0 THEN

IF A = 21 THEN

Print "Key"

ENDIF

ENDIF



1. Vẽ lưu đồ

2. Xác định độ phức tạp
chu trình= 3

3. Xác định đường thi hành cơ bản

1) 1-2-3-4-5

2) 1-2-5

3) 1-2-3-5

4. Các ca kiểm thử

TC	Đầu vào	Đầu ra mong đợi	Kết quả
1	A=21	In ra "Key"	
2	A=-5	Kết thúc CT	
3	A= 15	Kết thúc CT	

Bài tập 1

Xét đoạn code sau, yêu cầu: thiết kế các ca kiểm thử đạt bao phủ mức 4.

```
using namespace std;
#include<iostream>
main() {
    int n;
    cout<< "Nhap n"<<endl;
    cin >>n;
    for (n; n>0; n--) {
        cout<< n<< ",";
    }
    cout<< "Ket thuc";
    return 0;
}
```

Bài tập 1

```
using namespace std;
```

```
#include<iostream>
```

```
main() {
```

```
    int n;
```

```
    1 cout << " Nhập n" << endl;
```

```
    cin >> n;
```

```
    for (n; n>0; n--) {
```

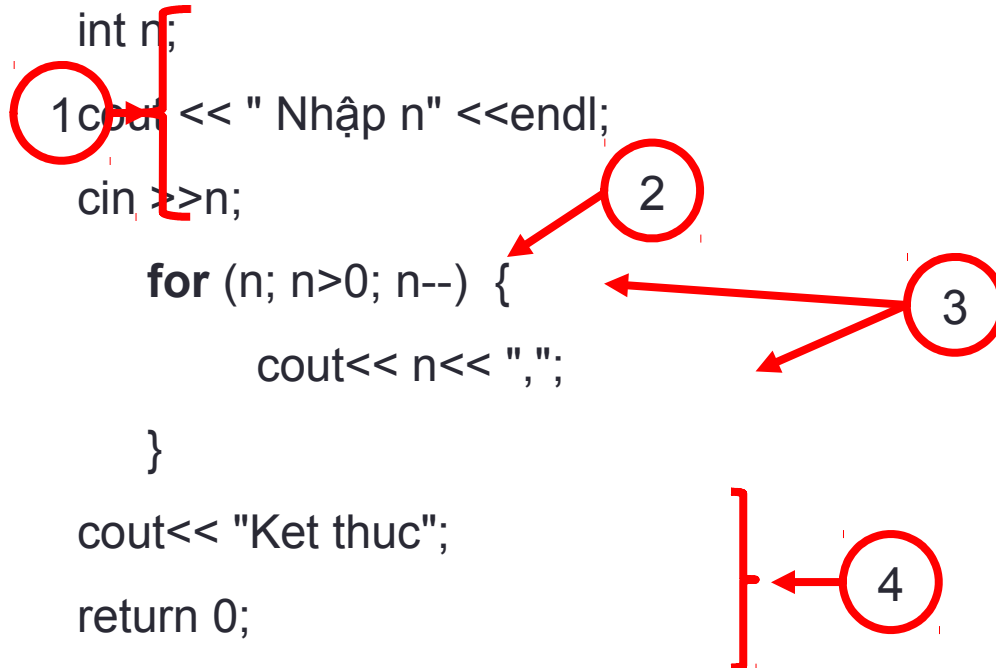
```
        cout<< n<< ",";
```

```
    }
```

```
    cout<< "Ket thuc";
```

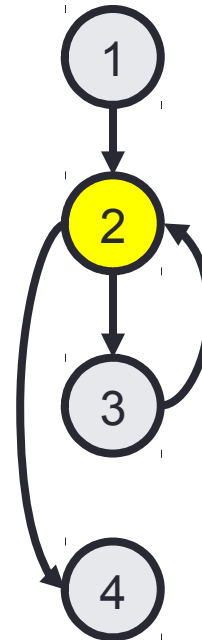
```
    return 0;
```

```
}
```



BT1.

- B1. Vẽ đồ thị dòng (hoặc lưu đồ)



- B2. Tính độ phức tạp của chu trình $C=4-4+2=2$

BT2

- B3. Xác định các đường độc lập

1. 1-2-3-2

2. 1-2-4

- B4. Xác định các trường hợp kiểm thử

TC	Đầu vào	Đầu ra mong đợi
1	n=10	In ra “Nhập n” In ra “10,9,8,7,6,5,4,3,2,1, Kết thúc”
2	n=0	In ra “Nhập n” In ra “Kết thúc”

Bài tập 2

a. Xét đoạn code sau, yêu cầu: thiết kế các ca kiểm thử đạt bao phủ mức 2.

```
using namespace std;
```

```
#include <iostream>
```

```
main() {
```

```
    int a,b,c,d,x,y;
```

```
    cout<<"Nhap a, b, c, d, x, y"<<endl;
```

```
    cin>>a>>b>>c>>d>>x>>y;
```

```
    if (a>0&&b==1){x=x+1;}
```

```
    if (c==3 || d<0) {y=0;}
```

```
    cout<<"x = "<<x<<endl;
```

```
    cout<<"y = "<<y<<endl;
```

```
}
```

BT 2.a

```
using namespace std;
```

```
#include <iostream>
```

```
main() {
```

```
    int a,b,c,d,x,y;
```

```
    cout<<"Nhap a, b, c, d, x, y"<<endl;
```

```
    2  >>a>>b>>c>>d>>x>>y;
```

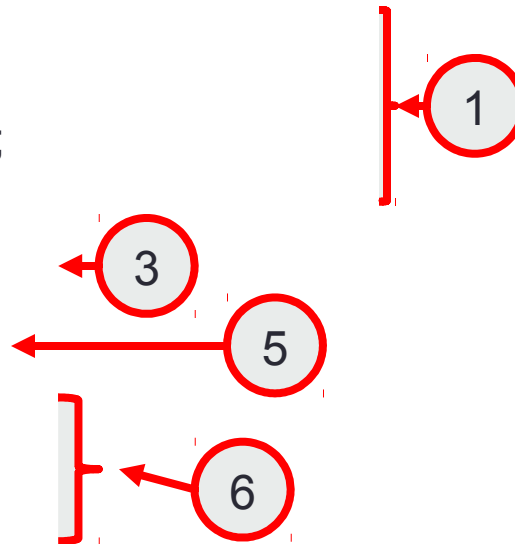
```
    if (a>0&&b==1){x=x+1;}
```

```
    if (c==3 || d<0) {y=0;}
```

```
    4  <<"x = "<<x<<endl;
```

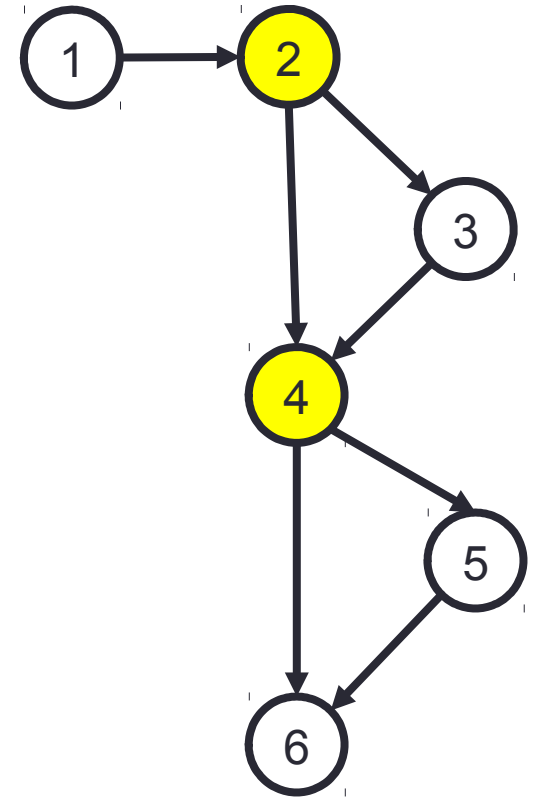
```
    cout<<"y = "<<y<<endl;
```

```
}
```



BT2.a

- B1. Vẽ đồ thị dòng (hoặc lưu đồ)



- B2. Tính độ phức tạp của chu trình $C=3$

BT2.a

- B3. Xác định các đường độc lập

1. 1-2-3-4-5-6
2. 1-2-4-5-6
3. 1-2-3-4-6

- B4. Xác định các ca kiểm thử

TC Đầu vào

1 a=5, b=1, c=3, d=2, x=7, y=7

2 a=5, b=3, c=3, d=2, x=7, y=7

3 a=5, b=1, c=0, d=2, x=7, y=7

Đầu ra mong đợi

Nhap a, b, c, d, x, y
x=8, y=0

Nhap a, b, c, d, x, y
x=7, y=0

Nhap a, b, c, d, x, y
x=8, y=7

Bài tập 2.b

b. Xét đoạn code sau, yêu cầu: thiết kế các ca kiểm thử đạt bao phủ mức 4.

```
using namespace std;
```

```
#include <iostream>
```

```
main() {
```

```
    int a,b,c,d,x,y;
```

```
    cout<<"Nhap a, b, c, d, x, y"<<endl;
```

```
    cin>>a>>b>>c>>d>>x>>y;
```

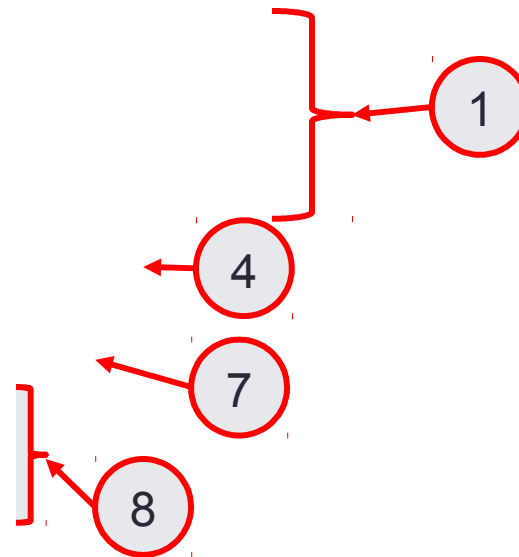
```
    if (a>0&&b==1)    {x=3;
```

```
        c==3|| d<0)    {y=6;
```

```
        cout<<"x = "<<x<<endl;
```

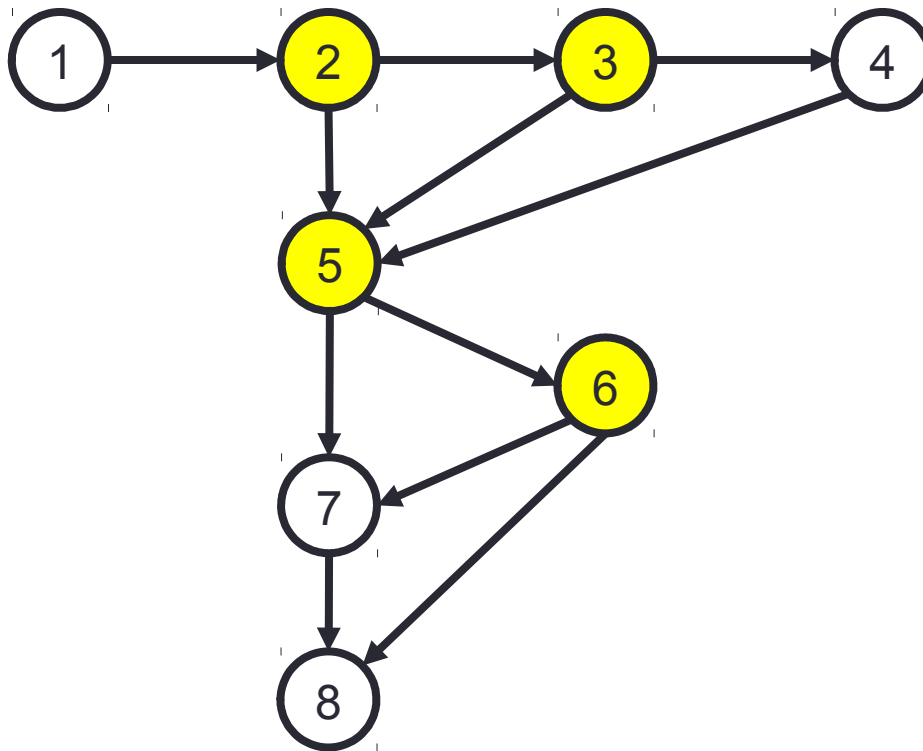
```
        cout<<"y = "<<y<<endl;
```

```
}
```



BT2.b

- B1. Vẽ đồ thị dòng (hoặc lưu đồ)



- B2. Tính độ phức tạp của chu trình $C=11-8+2=5$

BT2.b

- B3. Xác định các đường độc lập

1. 1-2-3-4-5-6-7-8
2. 1-2-5-6-7-8
3. 1-2-3-5-6-7-8
4. 1-2-3-4-5-7-8
5. 1-2-3-4-5-6-8

BT2.b

- B4. Xác định các trường hợp kiểm thử

TC	Đầu vào	Đầu ra mong đợi
1	a=5,0,b=1,c=4,d=-2, x=7,y=7	Nhap a, b, c, d, x, y x=8, y=0
2	a=-3,b=1,c=4,d=-2, x=7,y=7	Nhap a, b, c, d, x, y x=7,y=0
3	a=5 , b =10,c=4,d=-2, x=7,y=7	Nhap a, b, c, d, x, y x=7, y=0
4	a=5,b=1, c=3, d=5, x=7,y=7	Nhap a, b, c, d, x, y x=8, y=0
5	a=5, b=1, c=4, d=5, x=7, y=7	Nhap a, b, c, d, x, y x=8, y=7

Bài tập 3

Thiết kế các ca kiểm thử thỏa mãn tiêu chuẩn phủ cấp 4

```
double average(int[] values, int min, int max) {  
    int sum=0, count=0, item=0;  
    double average= 0;  
    while(values[item] !=-999 && item <100) {  
        if (values[item]>= min && values[ item] <= max) {  
            sum += values [item];  
            count ++;  
        }  
        item++;  
    }  
    if (count>0) average= (double) sum/count;  
    else average =-999;  
    return average;  
}
```

Bài tập 3

Thiết kế các ca kiểm thử thỏa mãn tiêu chuẩn phủ cấp 4

```
double average(int[] values, int min, int max) {
```

```
    int sum=0, count=0, item=0;
```

```
    double average= 0;
```

```
    while(values[item] !=-999 && item <100) {
```

```
        if (values[item]>= min && values[ item] <= max) {
```

```
            sum += values [item];
```

```
            count ++;
```

```
        }
```

```
        item++;
```

```
    }
```

```
    if (count>0) average= (double) sum/count;
```

```
    else average =-999;
```

```
    return average;
```

```
}
```

1

3

5

6

7

9

10

11

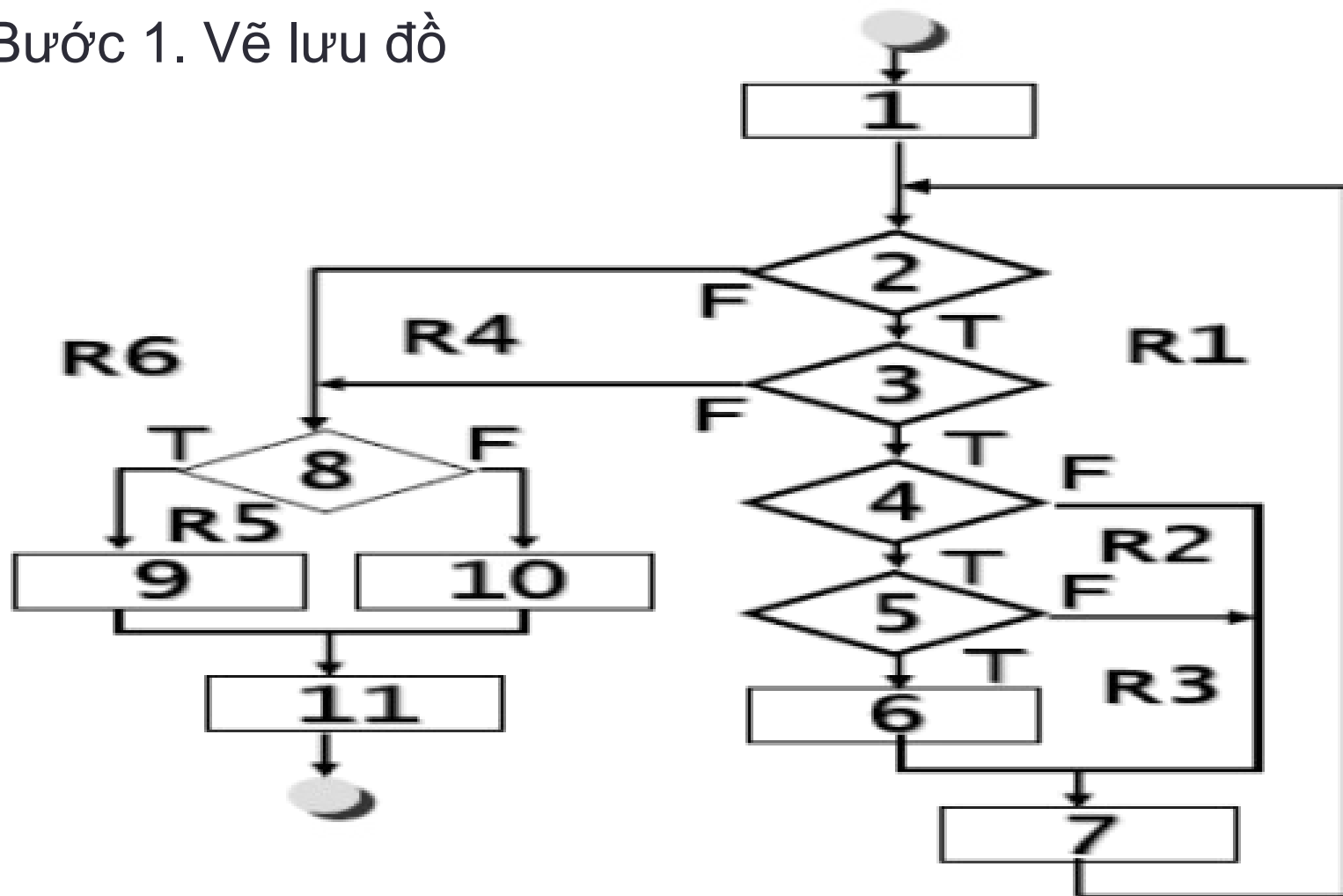
2

4

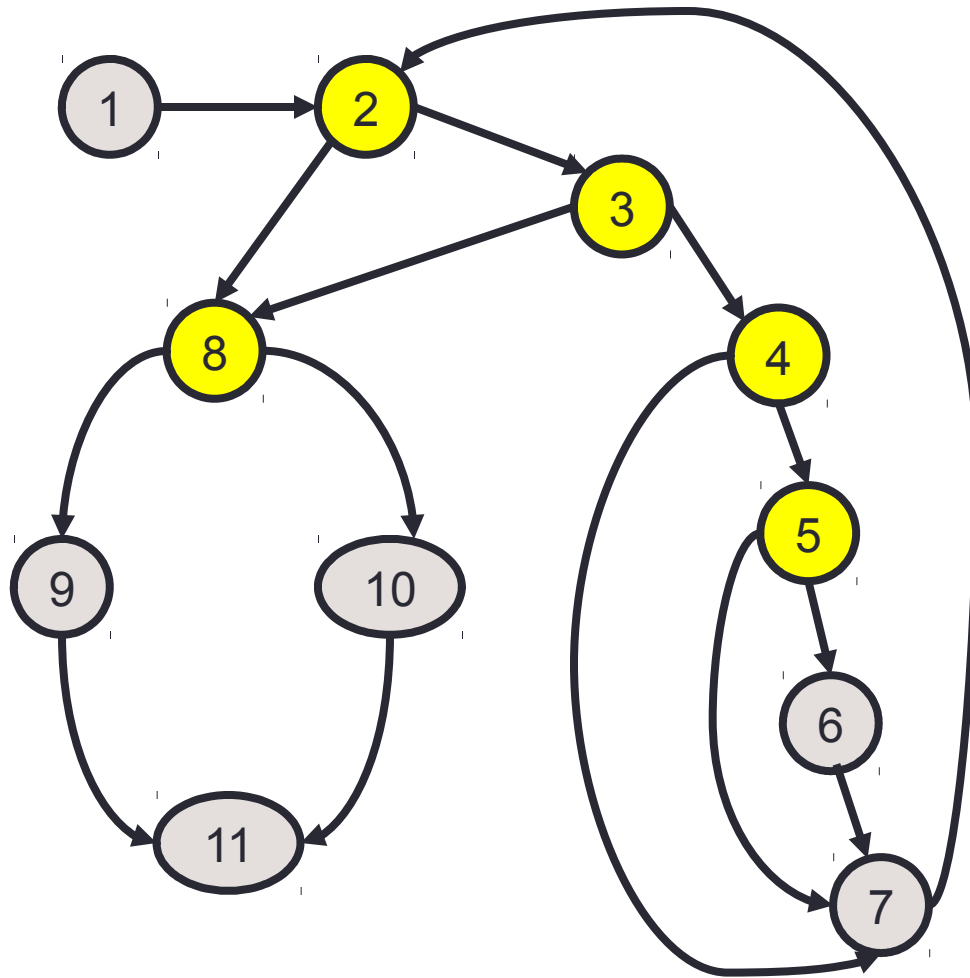
8

BT3.

- Bước 1. Vẽ lưu đồ

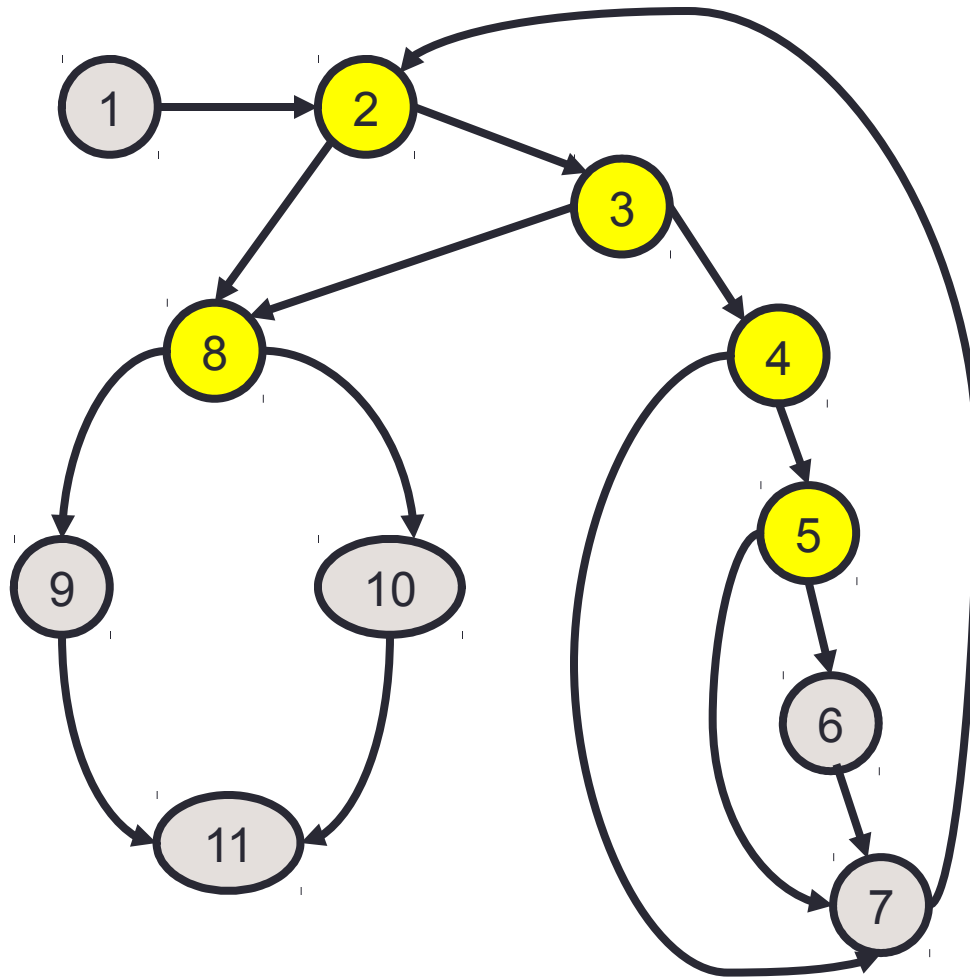


Bài tập 3.1. Đồ thị dòng



- Bước 1. Hoạch vẽ đồ thị dòng
- Bước 2. Xác định độ phức tạp của chu trình
- $C=6$

Bài tập 3.1. Đồ thị dòng



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

Bài tập 3

- Bước 3. Xác định các đường độc lập:

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

Bài tập 3

- Bước 4. Lập các ca kiểm thử

TC Đầu vào

Đầu ra mong đợi

1 $M = \{1, 9, -999, 37\}$, $\min = 0$, $\max = 10$

Giá trị trung bình $\text{avg} = (1+9)/2$

2 $M = \{-999, 1, 7, 21\}$, $\min = 0$, $\max = 10$

$\text{avg} = -999$

3 $M = \{5, 6, 7, 120, \dots, 100\}$ (>100 phần tử)
 $\min = 0$, $\max = 100$

avg = giá trị trung bình của
100 phần tử đầu tiên thỏa
mãn điều kiện \max , \min

4 $M = \{7, 8, 18, 20, -999\}$
 $\min = 10$, $\max = 100$

$\text{avg} = (18+20)/2$

5 $M = \{7, 80, 9, 20, 8, -999, 45\}$
 $\min = 0$, $\max = 10$

$\text{avg} = (7+8+9)/3$

6 $M = \{7, 8, 9, \dots, 100\}$ ($M[i] \neq -999$ và
 $\min \leq M[i] \leq \max$ với mọi $i \leq 100$)
 $\min = 0$, $\max = 100$

avg = giá trị trung bình của
100 phần tử hợp lệ

Bài tập 4

- Thiết kế các ca kiểm thử thỏa mãn tiêu chuẩn phủ cấp 4

```
function goodstring(var count: integer): boolean;
```

```
var ch: char;
```

```
begin
```

```
    goodstring:= false;
```

```
    count:=0;
```

```
    read(ch);
```

```
    if ch='a' then
```

```
        begin
```

```
            read(ch);
```

```
            while (ch='b') or (ch='c') do
```

```
                begin
```

```
                    count:= count+1;
```

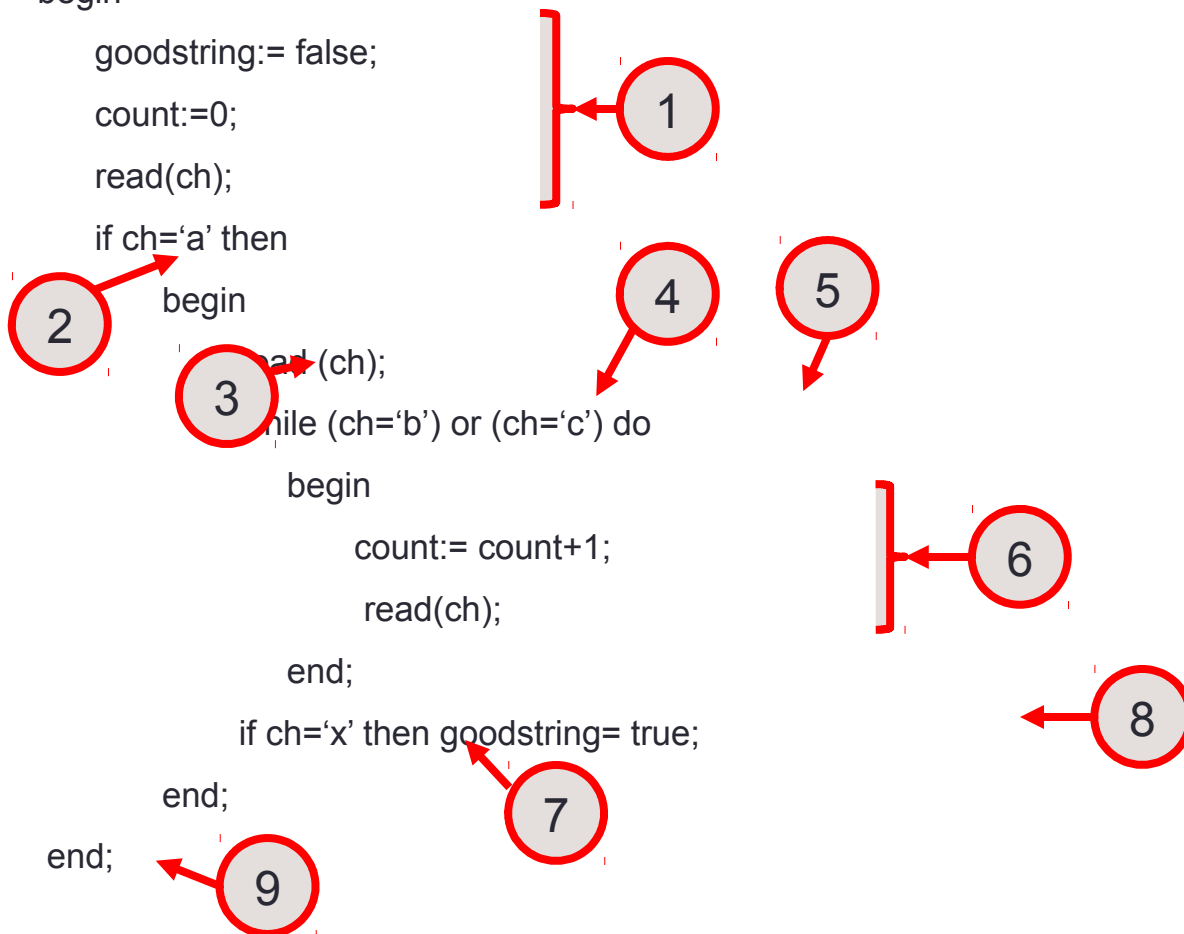
```
                    read(ch);
```

```
                end;
```

```
            if ch='x' then goodstring= true;
```

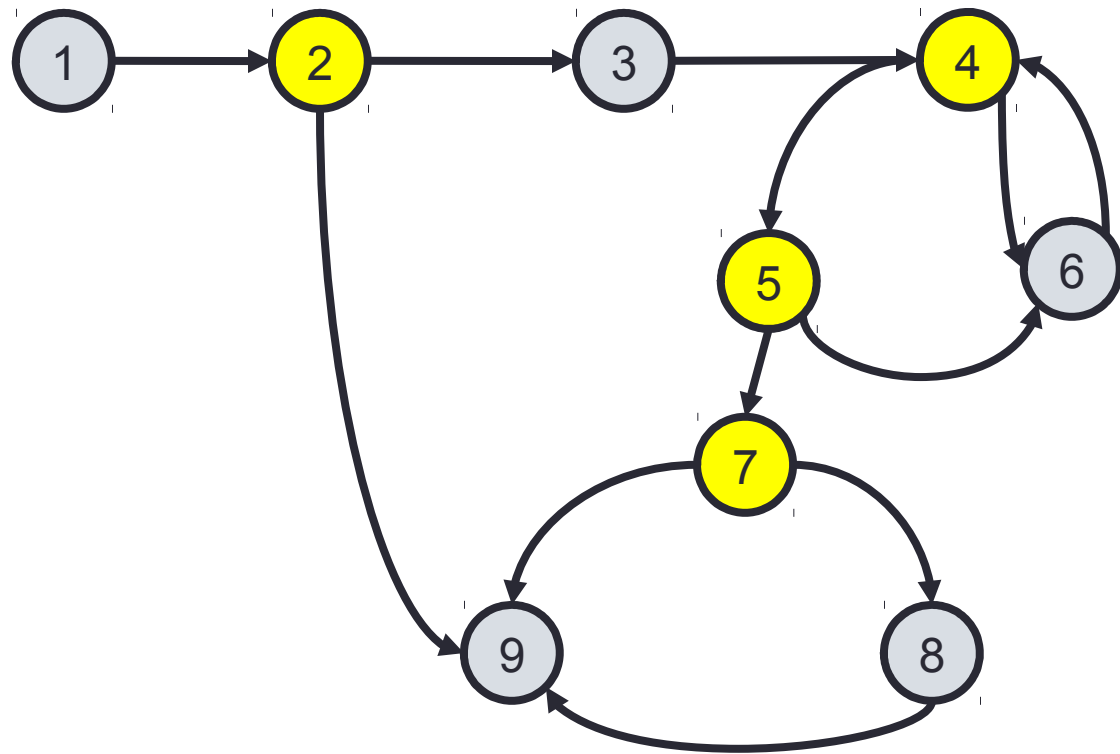
```
        end;
```

```
    end;
```



BT4

- B1. Vẽ đồ thị dòng



- B2. Tính độ phức tạp của chu trình $C=12-9+2=5$

BT4

- B3. Xác định các đường đi độc lập

1. 1-2-9
2. 1-2-3-4-6-4...
3. 1-2-3-4-5-6-4...
4. 1-2-3-4-5-7-8-9
5. 1-2-3-4-5-7-9

BT4.

- B4. Xác định các ca kiểm thử

TC	Đầu vào	Đầu ra mong đợi
1	Nhập “d”	Goodstring= false Count=0
2	Nhập “abbbx”	Goodstring= true Count=3
3	Nhập “ accccd”	Goodstring= false Count=4
4	Nhập “ax”	Goodstring= true Count=0
5	Nhập “ad”	Goodstring= false Count=0

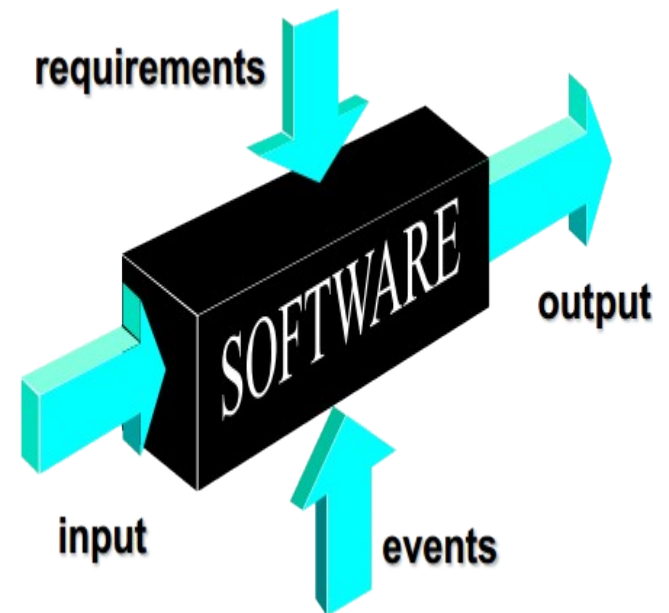
5.2. Kiểm thử hộp đen

5.2.1. Tổng quan về kiểm thử hộp đen

5.2.2. Các phương pháp kiểm thử hộp đen

5.2.1. Tổng quan về kiểm thử hộp đen

- Phương pháp kiểm thử hộp đen: coi hệ thống là một hộp đen, không thể thấy được cấu trúc logic bên trong. Người làm kiểm thử tập trung vào các yêu cầu chức năng của phần mềm dựa trên các dữ liệu lấy từ đặc tả
- Đặc trưng:
 - Nhằm thuyết minh: các chức năng phần mềm đủ & vận hành đúng
 - Thực hiện các phép thử qua giao diện



5.2.1. Tổng quan về kiểm thử hộp đen

- Kiểm thử hộp đen nhằm tìm ra các loại sai:
 - Chức năng thiếu hoặc không đúng đắn.
 - Sai về giao diện.
 - Sai trong cấu trúc hoặc trong truy cập dữ liệu ngoài.
 - Sai thực thi chức năng.
 - Sai khởi đầu hoặc kết thúc mô đun.

5.2.2. Các phương pháp kiểm thử hộp đen

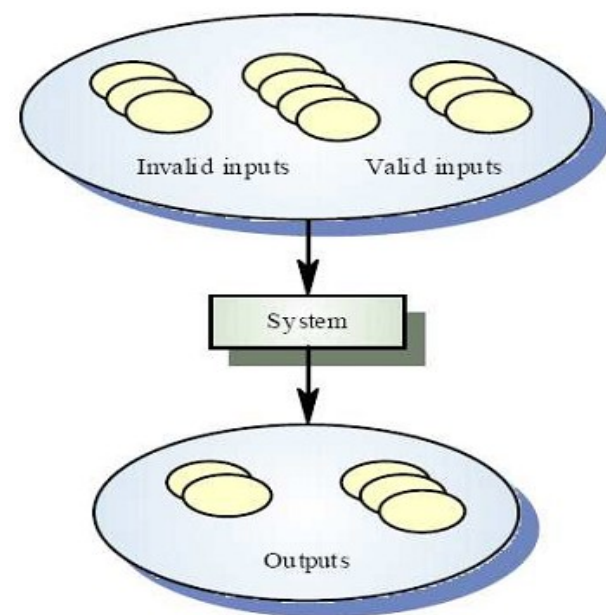
5.2.2.1. Phân hoạch tương đương (Equivalence Partitioning)

5.2.2.2. Phân tích giá trị biên (Boundary Value Analysis)

5.2.2.3. Bảng hỗ trợ quyết định

5.2.2.1. Phân hoạch tương đương

- Ý tưởng: Chia miền vào chương trình thành các lớp dữ liệu. Xác định đầu vào hợp lệ và không hợp lệ để lập các ca kiểm thử theo các lớp đó
- Mỗi lớp dùng để kiểm thử một chức năng, gọi là lớp tương đương.
- Thay vì kiểm tra tất cả các giá trị đầu vào, có thể lựa chọn từ đầu vào cho riêng từng lớp



5.2.2.1. Phân hoạch tương đương (t)

- Nguyên tắc xác định lớp tương đương:
 - Nếu điều kiện đầu vào định rõ giới hạn của một mảng, hoặc một giá trị xác định thì chia vùng tương đương thành:
 - Một lớp tương đương hợp lệ
 - Hai lớp không hợp lệ
 - Một lớp đặc biệt (nếu có)
 - Nếu điều kiện đầu vào chỉ định là một tập giá trị, hoặc xác định là một kiểu đúng sai thì chia vùng tương đương thành :
 - Một lớp tương đương hợp lệ.
 - Một lớp tương đương không hợp lệ.
 - Một lớp đặc biệt (nếu có)

5.2.2.1. Phân hoạch tương đương (t)

- Ví dụ:

STT	Trường hợp của điều kiện đầu vào	Lớp tương đương	
		Hợp lệ	Không hợp lệ
1	Là 1 khoảng các giá trị	1	2
	Ví dụ: $[3,10]$	$3 < x < 10$	$x < 3$ hoặc $x > 10$
2	Là 1 giá trị cụ thể	1	2
	Ví dụ: 6	$x = 6$	$x < 6$ hoặc $x > 6$
3	Là 1 tập hợp các giá trị	1	1
	Ví dụ: $A = \{1,5,4,8,6\}$	x	
4	Là 1 giá trị Boolean	1	1

5.2.2.1. Phân hoạch tương đương (t)

- Biểu diễn các lớp tương đương

Điều kiện đầu vào

Các lớp tương đương
hợp lệ

Các lớp tương đương
không hợp lệ

5.2.2.1. Phân hoạch tương đương (t)

- Ví dụ: Khi cấp số thẻ thành viên clb, 3 số đầu của số thẻ phải nằm trong khoảng $[111, 222]$, nếu sai sẽ có thông báo yêu cầu nhập lại, 2 số cuối phải thuộc khoảng $[11, 99]$.
- Các lớp tương đương:
 - Đối với 3 số đầu:
 - ≥ 111 và ≤ 222 : hợp lệ
 - > 222 : không hợp lệ
 - < 111 : không hợp lệ
 - Để trống : trường hợp đặc biệt thuộc không hợp lệ

5.2.2.1. Phân hoạch tương đương (t)

- Đối với 2 số đầu:
- ≥ 11 và ≤ 99 : hợp lệ
- < 11 : không hợp lệ
- > 99 : không hợp lệ
- Để trống : trường hợp đặc biệt thuộc không hợp lệ

Điều kiện đầu vào	Các lớp tương đương hợp lệ	Các lớp tương đương không hợp lệ
Đối với 3 số đầu	<ul style="list-style-type: none">• 1. ≥ 111 và ≤ 222	<ul style="list-style-type: none">2. < 1113. > 2224. Để trống
Đối với 2 số sau	<ul style="list-style-type: none">• 5. ≥ 11 và ≤ 99	<ul style="list-style-type: none">6. < 117. > 998. Để trống

5.2.2.1. Phân hoạch tương đương (t)

- Xác định các trường hợp kiểm thử
 1. Gán 1 số duy nhất cho mỗi lớp tương đương.
 2. Cho đến khi tất cả các lớp tương đương hợp lệ được bao phủ bởi (hợp nhất thành) các ca kiểm thử thì viết Viết 1 ca kiểm thử mới bao phủ càng nhiều các lớp tương đương đó chưa được bao phủ càng tốt.
 3. Cho đến khi các ca kiểm thử đã bao phủ tất cả các lớp tương đương không hợp lệ thì ta viết 1 ca kiểm thử mà bao phủ một và chỉ một trong các lớp tương đương không hợp lệ chưa được bao phủ.

Lý do mà mỗi ca kiểm thử riêng bao phủ các trường hợp không hợp lệ là vì các kiểm tra đầu vào không đúng nào đó che giấu hoặc thay thế các kiểm tra đầu vào không đúng khác.

5.2.2.1. Phân hoạch tương đương (t)

- Ví dụ: cấp số thẻ
 1. Đánh số các lớp tương đương(1-8)
 2. Viết 1 ca kiểm thử bao phủ tất cả các lớp tương đương hợp lệ. A(1,5)
 3. Từ một lớp hợp lệ, bao phủ những lớp không hợp lệ còn lại B(1,6), C(1,7), D (1,8), E(2,5), F(3,5), G(4,5)
 4. Trường hợp 1 ca kiểm thử bao phủ tất cả các lớp không hợp lệ là không cần thiết, vì nếu 3 số đầu bị nhập sai sẽ có ngay thông báo.

5.2.2.1. Phân hoạch tương đương (t)

- Các ca kiểm thử:

TC	Đầu vào	Đầu ra mong đợi	Bao phủ
1	3 số đầu= 200 2 số sau= 50	20050	(1,5)
2	3 số đầu =200 2 số sau= 10	Yêu cầu nhập lại 2 số sau	(1,6)
3	3 số đầu =200 2 số sau= 111	Yêu cầu nhập lại 2 số sau	(1,7)
4	3 số đầu =200 2 số sau= để trống	Yêu cầu nhập 2 số sau	(1,8)
5	3 số đầu= 100	Yêu cầu nhập lại 3 số đầu	(2,5)
6	3 số đầu= 250	Yêu cầu nhập lại 3 số đầu	(3,5)
7	3 số đầu= để trống	Yêu cầu nhập 3 số đầu	(4,5)

5.2.2.2. Phân tích giá trị biên

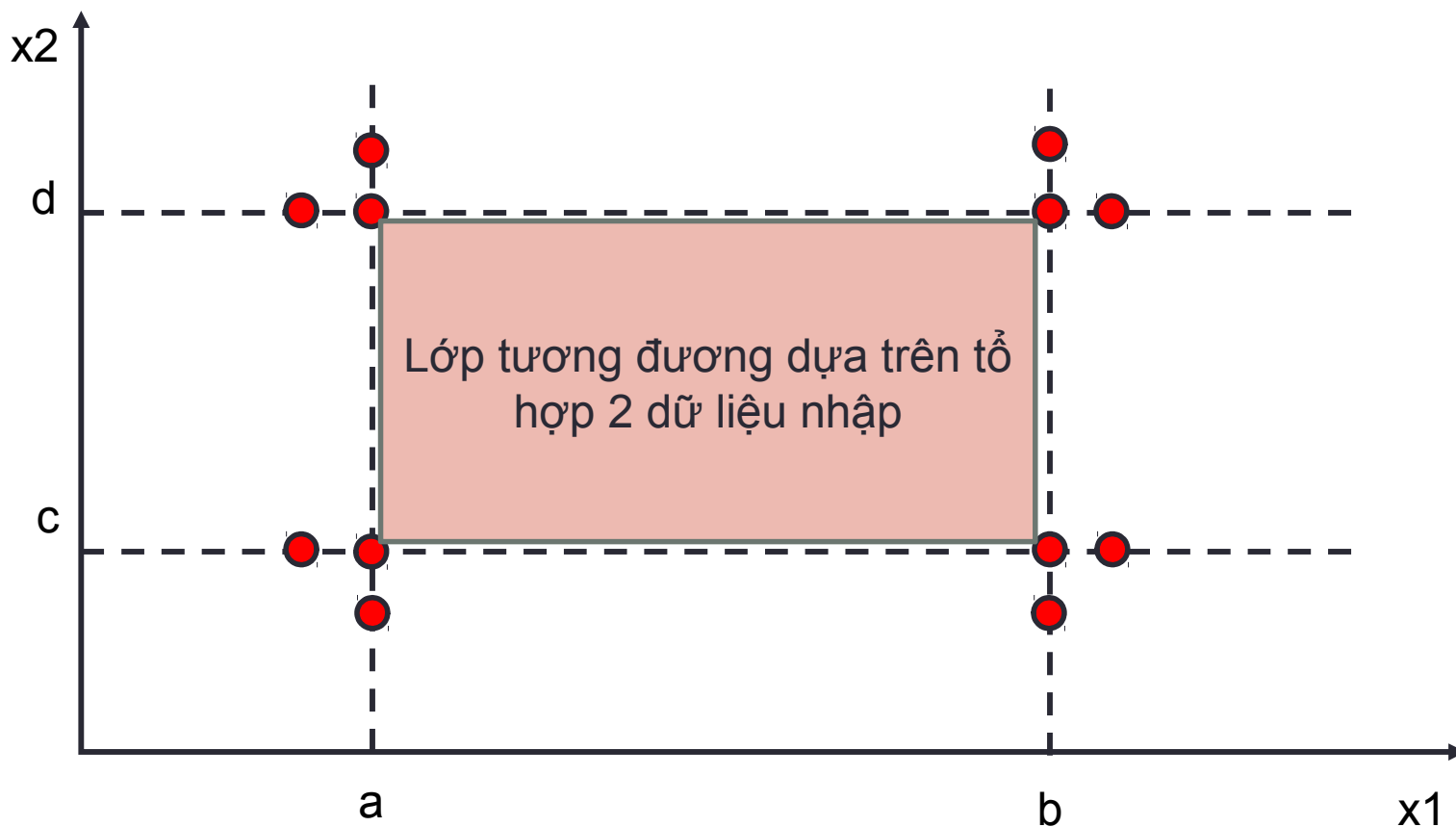
- **Cơ sở** : Tập trung phân tích các giá trị biên của miền dữ liệu để xây dựng dữ liệu kiểm thử
- **Nguyên tắc** : đối với một biến, kiểm thử các dữ liệu vào gồm:
 - Giá trị nhỏ nhất: \min
 - Giá trị gần kề lớn hơn giá trị nhỏ nhất: $\min+1$
 - Giá trị gần kề nhỏ hơn giá trị nhỏ nhất: $\min-1$
 - Giá trị lớn nhất : \max
 - Giá trị gần kề lớn hơn giá trị lớn nhất: $\max+1$
 - Giá trị gần kề nhỏ hơn giá trị lớn nhất: $\max-1$

5.2.2.2. Phân tích giá trị biên (t)

- Ví dụ phân tích giá trị biên của trường hợp sau:
 $20 \leq \text{Nhiệt độ} \leq 40$
- Các giá trị biên có thể:
 - Nhiệt độ = 20 độ
 - Nhiệt độ = 21 độ
 - Nhiệt độ = 19 độ
 - Nhiệt độ = 40 độ
 - Nhiệt độ = 39 độ
 - Nhiệt độ = 41 độ

1.2.2. Phân tích giá trị biên (t)

- Đối với 2 đầu vào x_1 , x_2 có điều kiện $x_1 \in [a, b]$, $x_2 \in [c, d]$



1.2.2. Phân tích giá trị biên (t)

- Ví dụ: Khi cấp số thẻ thành viên clb, 3 số đầu của số thẻ phải nằm trong khoảng [111, 222], nếu sai sẽ thông báo yêu cầu nhập lại, 2 số cuối phải thuộc khoảng [11,99].
- Ca kiểm thử các trường hợp đúng: (111,11), (111,99), (222,11),(222,99)
- Ca kiểm thử các trường hợp sai: (111,10), (111,100), (222,10), (222,100), (223,99), (223,11),(110,11),(110,99)

Ví dụ: (tiếp)

- Các ca kiểm thử

TC	Đầu vào	Đầu ra mong đợi
1	3 số đầu= 111, 2 số sau= 11	11111
2	3 số đầu =111, 2 số sau =99	11199
3	3 số đầu =222, 2 số sau= 11	22211
4	3 số đầu =222, 2 số sau= 99	22299
5	3 số đầu= 111, 2 số sau= 10	Yêu cầu nhập lại 2 số sau
6	3 số đầu= 111, 2 số sau= 100	Yêu cầu nhập lại 2 số sau
7	3 số đầu= 222, 2 số sau= 10	Yêu cầu nhập lại 2 số sau
8	3 số đầu= 222, 2 số sau= 100	Yêu cầu nhập lại 2 số sau
9	3 số đầu= 223, 2 số sau= 99	Yêu cầu nhập 3 số đầu
10	3 số đầu= 223, 2 số sau= 11	Yêu cầu nhập 3 số đầu
11	3 số đầu= 110, 2 số sau= 11	Yêu cầu nhập 3 số đầu
12	3 số đầu= 110, 2 số sau= 99	Yêu cầu nhập 3 số đầu

Nhận xét:

- Mỗi giá trị giới hạn đều nằm trong một phân vùng nào đó. Nếu chỉ sử dụng giá trị giới hạn thì ta cũng có thể kiểm tra luôn phân vùng đó
- Khi giá trị giới hạn bị sai thì cả phân vùng sẽ bị sai
- Nếu chỉ dùng giá trị giới hạn thì không đem lại sự tin tưởng cho người dùng vì các giá trị biên không xét được hết các trường hợp đặc biệt (để ô trống, nhập kí tự đặc biệt...)
- Vì vậy: nên kết hợp cả hai phương pháp phân tích giá trị biên và phân vùng tương đương

Ví dụ 1:

- TPPM “ Quản lý hồ sơ nhân lực” với đặc tả chức năng như sau: sau mỗi lần nhận 1 hồ sơ xin việc, TPPM sẽ ra quyết định ban đầu dựa và tuổi của ứng viên theo bảng sau:

Tuổi ứng viên	Kết quả sơ bộ
0-15	Không thuê
16-17	Thuê dạng bán thời gian
18-54	Thuê toàn thời gian
55-99	Không thuê

- Bảng phương pháp phân hoạch tương đương và phân tích giá trị biên, hãy thiết kế các trường hợp kiểm thử cho TPPM trên.

Ví dụ 1:

- Bảng phân lớp tương đương:

Tuổi	<0	0-15	16-17	18-54	55-99	>99
Kết quả	Nhập sai dữ liệu	Ko thuê	Thuê bán thời gian	Thuê toàn thời gian	Ko thuê	Nhập sai dữ liệu
Lớp tương đương	Ko hợp lệ	Hợp lệ	Hợp lệ	Hợp lệ	Hợp lệ	Ko hợp lệ
Đánh dấu	I1	V1	V2	V3	V4	I2

- Từ mỗi lớp tương đương, xét các biên cần kiểm thử
- Lớp I1: {-2,-1,0} Lớp I2: {99,100,101}
- Lớp V1: {-1,0,1} {14,15,16}
- Lớp V2: { 15,16,17} {16,17,18}
- Lớp V3: {17,18,19}, {53,54,55}
- Lớp V4: {54,55,56}, {98,99,100}

Ví dụ 1:

- Xét trong các trường hợp trên thấy có nhiều giá trị test case trùng nhau, nếu loại bỏ các test case trùng nhau đó thì ta còn: -1, 0, 1, 14, 15, 16, 17, 18, 19, 53, 54, 55, 56, 98, 99, 100.
- Do trường hợp -1 và 100 đã nằm tại biên của hai lớp tương đương ko hợp lệ nên ta ko xét trường hợp -2, và 101 ở lớp này.
- 16 ca kiểm thử được thiết kế như sau:

Ví dụ 1: Các ca kiểm thử

TC	Đầu vào	Đầu ra mong đợi
1	-1	Nhập sai dữ liệu
2	0	Không thuê
3	1	Không thuê
4	14	Không thuê
5	15	Không thuê
6	16	Thuê bán thời gian
7	17	Thuê bán thời gian
8	18	Thuê toàn thời gian

TC	Đầu vào	Đầu ra mong đợi
9	19	Thuê toàn thời gian
10	53	Thuê toàn thời gian
11	54	Thuê toàn thời gian
12	55	Không thuê
13	56	Không thuê
14	98	Không thuê
15	99	Không thuê
16	100	Nhập sai dữ liệu

Ví dụ 2: Ứng dụng cho vay nợ

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
<input type="checkbox"/> Term of loan	<input type="text"/>	1 to 30 years
<input type="checkbox"/> Monthly repayment	<input type="text"/>	Minimum £10

Term:

Repayment:

Interest rate:

Total paid back:

Vd2:Customer name

Số lượng kí tự:



Kiểu kí tự:



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

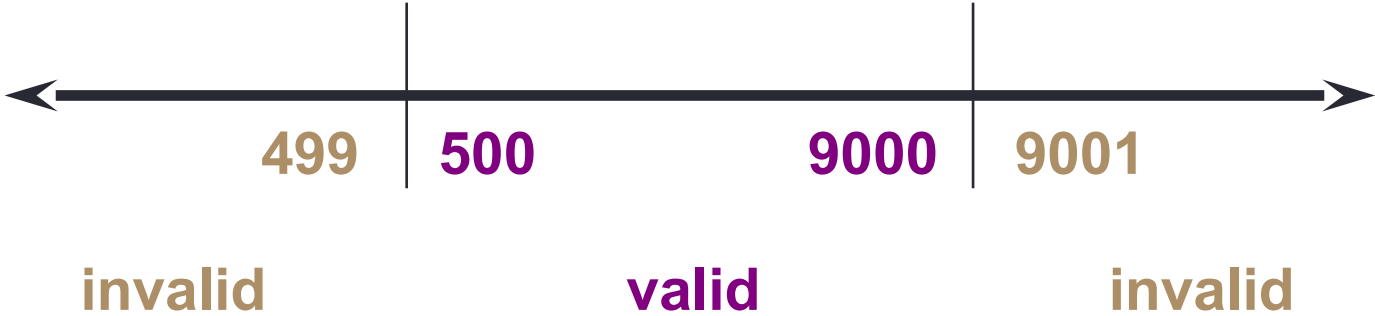
Vd2: Account number

Kí tự đầu tiên: **valid: non-zero**
invalid: zero



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

VD3: Loan amount



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

Tổng hợp các điều kiện

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

Các ca kiểm thử

Test Case	Description	Expected Outcome	New Tags Covered
1	Name: John Smith Acc no: 123456 Loan: 2500 Term: 3 years	Term: 3 years Repayment: 79.86 Interest rate: 10% Total paid: 2874.96	V1, V2, V3, V4, V5
2	Name: AB Acc no: 100000 Loan: 500 Term: 1 year	Term: 1 year Repayment: 44.80 Interest rate: 7.5% Total paid: 537.60	B1, B3, B5,
3
4

Bài tập 1

- Nếu bạn đi xe điện chuyển trước 9:30 sáng hoặc từ sau 4:00 chiều đến 7:30 tối (giờ cao điểm), thì bạn phải mua vé thường. Vé tiết kiệm (giá thấp hơn vé thường) có hiệu lực cho các chuyến xe từ 9:30 sáng đến 4:00 chiều và sau 7:30 tối. Tàu hoạt động từ 4:00 sáng tới 23:00 đêm
- **Thiết kế các ca kiểm thử để kiểm tra yêu cầu trên dựa vào phương pháp phân vùng tương đương và phân tích giá trị biên.**

BT1

- Bảng phân vùng tương đương

Lịch	00:00- 3:59	4:00- 9:29	9:30- 16:00	16:01- 19:30	19:31- 22:59	23:00- 23:59
Loại vé		Vé thường	Vé tiết kiệm	Vé thường	Vé tiết kiệm	
Vùng tương đương	Không hợp lệ	Hợp lệ	Hợp lệ	Hợp lệ	Hợp lệ	Không hợp lệ

- Các ca kiểm thử
- (bảng 1)

TC	Đầu vào	Đầu ra mong đợi
1	Xuất phát lúc 2h	Không hợp lệ
2	Xuất phát lúc 6h	Vé thường
3	Xuất phát lúc 12h	Vé tiết kiệm
4	Xuất phát lúc 18h	Vé thường
5	Xuất phát lúc 21h	Vé tiết kiệm
6	Xuất phát lúc 23h30	Không hợp lệ

BT1

- Các giá trị biên cần kiểm thử (Bảng 2)

TC	Đầu vào	Đầu ra mong đợi
1	Xuất phát lúc 0h	Không hợp lệ
2	Xuất phát lúc 3h59	Không hợp lệ
3	Xuất phát lúc 4h	Vé thường
4	Xuất phát lúc 9h29	Vé thường
5	Xuất phát lúc 9h30	Vé tiết kiệm
6	Xuất phát lúc 16h	Vé tiết kiệm
7	Xuất phát lúc 16h01	Vé thường
8	Xuất phát lúc 19h30	Vé thường
9	Xuất phát lúc 19h31	Vé tiết kiệm
10	Xuất phát lúc 22h59	Vé tiết kiệm
11	Xuất phát lúc 23h	Không hợp lệ

BT1

- Kết hợp bảng 1, 2 ta được các ca kiểm thử sau

TC	Đầu vào	Đầu ra mong đợi
1	Xuất phát lúc 0h	Không hợp lệ
2	Xuất phát lúc 2h	Không hợp lệ
3	Xuất phát lúc 3h59	Không hợp lệ
4	Xuất phát lúc 4h	Vé thường
5	Xuất phát lúc 6h	Vé thường
6	Xuất phát lúc 9h29	Vé thường
7	Xuất phát lúc 9h30	Vé tiết kiệm
8	Xuất phát lúc 12h	Vé tiết kiệm
9	Xuất phát lúc 16h	Vé tiết kiệm
10	Xuất phát lúc 16h01	Vé thường
11	Xuất phát lúc 18h	Vé thường
12	Xuất phát lúc 19h30	Vé thường
13	Xuất phát lúc 19h31	Vé tiết kiệm
14	Xuất phát lúc 21h	Vé tiết kiệm
15	Xuất phát lúc 22h59	Vé tiết kiệm
16	Xuất phát lúc 23h	Không hợp lệ
17	Xuất phát lúc 23h30	Không hợp lệ

Bài tập 2

- TPPM “ xét đơn cầm cố nhà” với đặc tả như sau: mỗi lần nhận 1 đơn xin cầm cố, phần mềm sẽ ra quyết định chấp thuận nếu 4 điều kiện sau thỏa mãn:
 - Thu nhập hàng tháng của người nộp đơn nằm trong khoảng từ 1000\$ với 83333\$
 - Số nhà xin cầm cố từ 1-5
- Dùng phương pháp phân hoạch tương đương và phân tích giá trị biên để thiết kế các trường hợp kiểm thử cho TPPM trên.

BT2

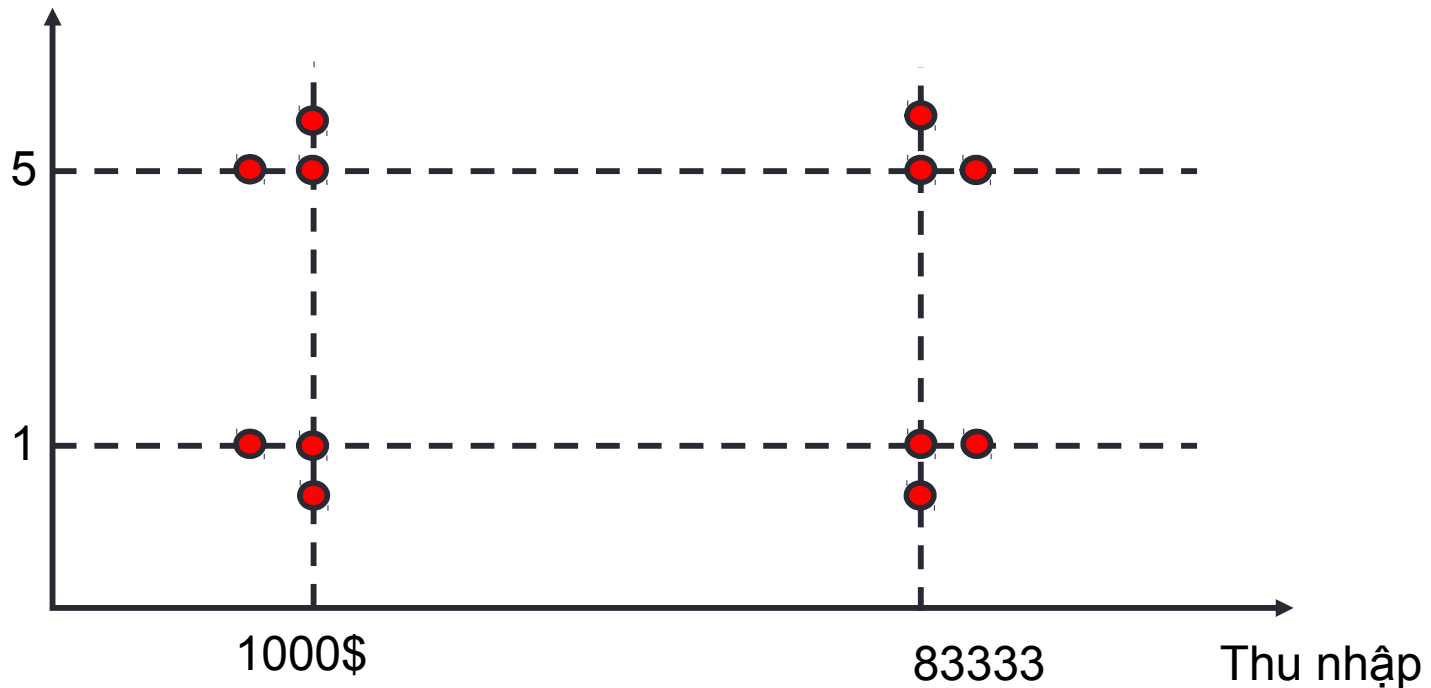
- Thu nhập hàng tháng

Đầu vào	Lớp hợp lệ	Đánh dấu	Lớp ko hợp lệ	Đánh dấu
Thu nhập hàng tháng	[1000\$,83333\$]	H1	<1000\$	K1
			>8333\$	K2
Số nhà cầm cổ	[1,5]	H2	<1	K3
			>5	K4

- Các ca kiểm thử TC1(H1,H2), TC2(H1,K3), TC3(H1,K4), TC4(K1,H2), TC5(K2, H2)

BT2

- Các giá trị biên cần kiểm tra
 - Với Thu nhập hàng tháng {999\$, 1000\$, 83333\$, 83334\$}
 - Với số nhà cầm cố {0, 1, 5, 6}
- Số lượng nhà



BT2. Các ca kiểm thử

TC	Đầu vào	Đầu ra mong đợi	
	Thu nhập	Số lượng nhà	
1	1000\$	1	Được thể chấp
2	50000\$	3	Được thể chấp
3	83333\$	1	Được thể chấp
4	1000\$	5	Được thể chấp
5	83333\$	5	Được thể chấp
6	1000\$	0	Không được thể chấp
7	1000\$	6	Không được thể chấp
8	83333\$	0	Không được thể chấp
9	83333\$	6	Không được thể chấp
10	999\$	1	Không được thể chấp
11	83334\$	1	Không được thể chấp
12	999\$	5	Không được thể chấp
13	83334\$	5	Không được thể chấp

Bài tập 3

- Viết chương trình dịch, trong đó có câu lệnh FOR, đặc tả câu lệnh FOR như sau: “Lệnh FOR chỉ chấp nhận một tham số duy nhất là biến đếm. Tên biến không được sử dụng quá hai ký tự khác rỗng. Sau ký hiệu = là cận dưới và cận trên của biến đếm. Các cận trên và cận dưới là các số nguyên dương và được đặt giữa từ khóa TO”.
- **Dùng phương pháp phân hoạch tương đương, thiết kế các ca kiểm thử cho câu lệnh FOR**

BT3: bảng phân vùng tương đương

Đầu vào	Hợp lệ	Đánh dấu	Ko hợp lệ	Đánh dấu
Tên biến	1-2 ký tự	H1	Rỗng	K1
			>2 ký tự	K2
Số lượng biến	1	H2	0	K3
			>1	K4
Cận trên	Số nguyên dương	H3	Không phải số nguyên dương	K5
			Không phải số nguyên	K6
			Không phải số	K7
			Rỗng	K8
Cận dưới	Số nguyên dương	H4	Không phải số nguyên dương	K9
			Không phải số nguyên	K10
			Không phải số	K11
			Rỗng	K12
Toán tử gán	=	H5	Ký tự khác dấu =	K13
Từ khóa To	To	H6	Ký tự khác To	K14

BT3. Test case

TC	Inputs	E. Output	Cover
1	i= 10 To 100	Vòng lặp hợp lệ	(H1,H2,H3,H4,H5,H6)
2	= 10 To 100	Không tìm thấy biến	(K1,H2,H3,H4,H5,H6) (H1,K3,H3,H4,H5,H6)
3	Bien= 10 To 100	Tên biến ko hợp lệ	(K2,H2,H3,H4,H5,H6)
4	i,j=10 To 100	Thừa biến	(H1,K4,H3,H4,H5,H6)
5	i=-10 To 100	Sai cận dưới	(H1,H2,K5,H4,H5,H6)
	i= 1.5 To 100	Sai cận dưới	(H1,H2,K6,H4, H5,H6)
6	i= a To 100	Sai cận dưới	(H1,H2,K7,H4,H5,H6)
7	i= To 100	Không tìm thấy cận dưới	(H1,H2,K8,H4,H5,H6)
8	i= 10 To -100	Sai cận trên	(H1,H2,H3,K9,H5,H6)
9	i=10 To 10.25	Sai cận trên	(H1,H2,H3,K10,H5,H6)
10	i =10 To a	Sai cận trên	(H1,H2,H3,K11,H5,H6)
11	i= 10 To	Không tìm thấy cận trên	(H1,H2,H3,K12,H5,H6)
13	i <> 1 To 100	Sai cú pháp lệnh	(H1,H2,H3,H4,K13,H6)
14	i=1 Tuo 100	Sai từ khóa	(H1,H2,H3,H4,H5,K14)

Bài tập 3

Bài toán tìm nghiệm thực cho phương trình bậc 2:

Calculator for the quadratic equation:
 $0 = ax^2 + bx + c$
 $a =$ $b =$ $c =$

Biết a, b, c là các số thực $\in [-10, 100]$

Đầu ra có thể gặp sau khi nhập bộ 3 số a, b, c và bấm nút Calculate là:

1. Không phải là phương trình bậc 2.
2. Phương trình vô nghiệm
3. Phương trình có một nghiệm (đưa ra giá trị của nghiệm)
4. Phương trình có 2 nghiệm (đưa ra giá trị của 2 nghiệm)
5. Nhập sai dữ liệu

Thiết kế các ca kiểm thử dùng phương pháp phân hoạch tương đương

Bảng phân vùng tương đương

Đầu vào	Vùng hợp lệ	Đánh dấu	Vùng ko hợp lệ	Đánh dấu
A	[-10, 100]	H1	<-10	K1
			>100	K2
			Không phải số	K3
			=0	K4
			Để trống	K5
B	[-10, 100]	H2	<-10	K6
			>100	K7
			Không phải số	K8
			Để trống	K9
C	[-10, 100]	H3	<-10	K10
			>100	K11
			Không phải số	K12
			Để trống	K13

Ca kiểm thử

TC	inputs		E. output	Cover
	3 số a,b,c	Delta		
1	a=2, b=5, c= 2	Delta >0	Delta =9 Pt có 2 nghiệm $\{-1/2, -2\}$	H1,H2,H3
2	a=4, b=-4, c= 1	Delta=0	Delta=0 Pt có 1 nghiệm $\{1/2\}$	H1,H2,H3
3	a=1, b=-7, c= 16	Delta<0	Delta <0 Pt vô nghiệm	H1,H2,H3
4	a=-20, b=5, c= 2		Nhập sai dữ liệu	K1,H2,H3
5	a=200, b=5, c= 2		Nhập sai dữ liệu	K2,H2,H3
6	a= abc, b=5, c=2		Nhập sai dữ liệu	K3,H2,H3
7	a=0, b=5, c=2		Không phải pt bậc 2	K4,H2,H3
8	Để trống, b=5, c=2		Nhập sai dữ liệu	K5,H2,H3
9	a= 2, b=-50, c=2		Nhập sai dữ liệu	H1, K6,H3

Ca kiểm thử (tiếp)

TC	inputs	E. output	Cover
	3 số a,b, c		
10	a= 2, b=500, c=2	Nhập sai dữ liệu	H1,K7,H3
11	a= 2, Nhập abcde, c=2	Nhập sai dữ liệu	H1,K8, H3
12	a= 2, để trống, c=2	Nhập sai dữ liệu	H1,K9,H3
13	a= 2, b=5, c=-20	Nhập sai dữ liệu	H1,H2,K10
14	a= 2, b=5, c=200	Nhập sai dữ liệu	H1,H2,K11
15	a= 2, b=5, Nhập abcde	Nhập sai dữ liệu	H1,H2,,K12
16	a= 2, b=5 , Để trống	Nhập sai dữ liệu	H1,H2,K13

Bài tập 5

- Chương trình tính chi phí cho bệnh nhân dựa trên độ tuổi và giới tính có màn hình và các yêu cầu như sau:

Calculate the Payment for the Patient

2 ☒ Male ☐ Female ☐ Child (0 - 17 years)

Age (Years) **3**

4

Payment is **5** euro €

Male	
Age	Payment
18-35	100 euro
36-50	120 euro
51-145	140 euro
Female	
Age	Payment
18-35	80 euro
36-50	110 euro
51-145	140 euro
Child	
Age	Payment
0-17	50 euro

Bài tập 5

- Mô tả chức năng:

1. Khởi tạo màn hình:

- Item 2 được check mặc định ở "Male"

- Item 3 và 5 null

- Item 4 ở trạng thái enable (có thể click được)

2. Mô tả xử lý chính:

- Khi click vào item 4 thì xử lý như sau:

- + Nếu là Male

- ++ Độ tuổi từ 18 đến 35 thì nhận được 100€

- ++ Độ tuổi từ 36 đến 50 thì nhận được 120€

- ++ Độ tuổi từ 51 đến 145 thì nhận được 140€

- ++ Độ tuổi khác thì hiển thị thông báo lỗi: "Xin vui lòng nhập độ tuổi chính xác"

- + Nếu là Female

- ++ Độ tuổi từ 18 đến 35 thì nhận được 80€

- ++ Độ tuổi từ 36 đến 50 thì nhận được 110€

- ++ Độ tuổi từ 51 đến 145 thì nhận được 140€

- ++ Độ tuổi khác thì hiển thị thông báo lỗi: "Xin vui lòng nhập độ tuổi chính xác"

Bài tập 5

- Dùng phương pháp phân hoạch tương đương và phân tích giá trị biên để xây dựng các ca kiểm thử cho chương trình trên.

5.2.2.3. Bảng quyết định

- Cung cấp cách biểu diễn chính xác các điều kiện logic và các hành động tương ứng
- Bảng quyết định có dạng:

	Luật 1	Luật 2	...	Luật p
Điều kiện				
Đk 1				
...				
Đk m				
Hành động				
Hđ1				
...				
Hđ n				

4.2.3. Bảng quyết định

- Quy trình dùng bảng quyết định để xây dựng các ca kiểm thử:
- Dựa vào đặc tả yêu cầu phần mềm, lập bảng quyết định.
- Từ bảng quyết định chuyển thành các ca kiểm thử trong đó mỗi cột miêu tả 1 luật được chuyển thành 1 tới n cột miêu tả các ca kiểm thử tương ứng với luật đó:
 - Nếu điều kiện nhập thử là giá trị luận lý thì mỗi cột luật được chuyển thành 1 ca kiểm thử
 - Nếu điều kiện nhập là 1 lớp tương đương(nhiều giá trị liên tục thì mỗi cột luật được chuyển thành nhiều ca kiểm thử dựa trên phương pháp phân hoạch tương đương hoặc giá trị biên)

Ví dụ:

- Công ty Honda trao học bổng cho những bạn sinh viên thỏa mãn ít nhất 1 trong 2 điều kiện sau: Là sinh viên giỏi , là cán bộ lớp.
- Nếu thỏa mãn cả 2 điều kiện sẽ được học bổng 600\$, nếu thỏa mãn là sinh viên giỏi được học bổng 400\$, nếu là cán bộ lớp được học bổng là 300\$.
- Lập bảng hỗ trợ quyết định để thiết kế các ca kiểm thử

VD:

- Bảng hỗ trợ quyết định:

		Luật 1	Luật 2	Luật 3	Luật 4
Điều kiện	Là cán bộ lớp	Y	Y	N	N
	Là sv giỏi	Y	N	Y	N
Hành động	Học bổng	600\$	300\$	400\$	0

- Các TC đầu vào: Đầu ra mong đợi
- 1 Là cán bộ lớp, là sv giỏi 600\$
- 2 Là cán bộ lớp, ko phải là sv giỏi 300\$
- 3 Ko phải cán bộ lớp, là sv giỏi 400\$
- 4 Ko phải cán bộ lớp, ko phải sv giỏi 0

VD2

- Chương trình quản lý tiền vé và số lượng mũ phát cho khách hàng vào thăm quan bảo tàng được mô tả như sau:
- Vé bán có các mức sau: đối với trẻ em dưới 5 tuổi được miễn phí, đối tượng từ 5 tuổi tới 65 tuổi phải trả 20\$/vé, đối tượng lớn hơn 65 tuổi phải trả 10\$/vé.
- Chương trình tặng mũ cho khách: với những đối tượng là nữ sẽ tặng mũ hồng, đối tượng là nam sẽ tặng mũ xanh
- Dùng bảng hỗ trợ quyết định xây dựng các ca kiểm thử cho chương trình trên

Vd2:

- Lập bảng hỗ trợ quyết định

		Luật 1	Luật 2	Luật 3	Luật 4	Luật 5	Luật 6
Điều kiện	Tuổi <5	Y	Y				
	5<=tuổi<=65			Y	Y		
	Tuổi >65					Y	Y
	Giới tính	Y	N	Y	N	Y	N
	Miễn phí	Y	Y				
	Giá 10\$					Y	Y
	Giá 20\$			Y	Y		
	Tặng mũ hồng	Y		Y		Y	
	Tặng mũ xanh		Y		Y		Y
Hành động							

VD2

- Các ca kiểm thử dựa trên bảng hỗ trợ quyết định

TC	Đầu vào	Đầu ra mong đợi
1	Tuổi <5, giới tính: Nữ	Miễn phí vé vào, tặng mũ đỏ
2	Tuổi <5, giới tính: Nam	Miễn phí vẽ vào, tặng mũ xanh
3	5 ≤ tuổi ≤ 65, giới tính: Nữ	Giá vé 20\$, tặng mũ đỏ
4	5 ≤ tuổi ≤ 65, giới tính: Nam	Giá vé 20\$, tặng mũ xanh
5	Tuổi >65, giới tính: Nữ	Giá vé 10\$, tặng mũ đỏ
6	Tuổi >65, giới tính: Nam	Giá vé 10\$, tặng mũ xanh

Bài tập 1

- Nếu bạn có thẻ đường sắt "over 60s" thì được giảm giá 34% trên tất cả các vé bạn mua.
- Khi bạn đi cùng với trẻ em (dưới 16 tuổi), thì bạn sẽ được giảm 50% nếu bạn có thẻ "family rail card", trong trường hợp ko có thẻ bạn chỉ được giảm 10%
- Bạn chỉ được sử dụng 1 hình thức khuyến mại trong 1 giao dịch
- Hãy viết bảng quyết định liệt kê toàn bộ các kết hợp loại thẻ và kết quả giảm giá. Và viết test case từ bảng quyết định này

BT1

- Lập bảng quyết định

		L1	L2	L3	L4	L5	L6	L7	L8
Điều kiện	Thẻ over 60s	Y	Y	Y	Y	N	N	N	N
	Thẻ family	Y	Y	N	N	Y	Y	N	N
	Đi cùng trẻ em <16t	Y	N	Y	N	Y	N	Y	N
Hành động	Giảm 34%		Y	Y	Y				
	Giảm 50%	Y				Y			
	Giảm 10%							Y	
	Không giảm						Y		Y

BT1

- Rút gọn bảng

		L1	L2,4	L3	L5	L6,8	L7
Điều kiện	Thẻ over 60s	Y	Y	Y	N	N	N
	Thẻ family	Y	-	N	Y	-	N
	Đi cùng trẻ em	Y	N	Y	Y	N	Y
Hành động	Giảm 34%		Y	Y			
	Giảm 50%	Y			Y		
	Giảm 10%						Y
	Không giảm					Y	

- Có tất cả 6 test case

BT1

- Xây dựng test case

TC	Inputs	Expected outputs
1	Gia đình Smiths có thẻ over 60s, có thẻ family rail card, đi cùng Oliver (10 tuổi)	Được giảm 50%
2	Gia đình Smiths có thẻ over 60s, không đi cùng trẻ em	Được giảm 34%
3	Gia đình Smiths có thẻ over 60s, đi cùng Oliver (10 tuổi)	Được giảm 34%
4	Gia đình Smiths ko có thẻ over 60s, có thẻ family raid card, đi cùng Oliver (10t)	Được giảm 50%
5	Ông Smith ko có thẻ over 60s, ko có thẻ family raid card, đi cùng Oliver (10t)	Được giảm 10%
6	Ông Smith đi một mình, ko có thẻ nào	Không được giảm

BT 2

- Một chương trình phân loại kết quả học của sinh viên dựa trên tổng điểm. Biết tổng điểm của sinh viên (tối đa là 100) trong một kì bằng điểm thành phần cộng điểm thi.
- Trong đó điểm thi tối đa là 75 điểm, điểm thành phần tối đa là 25 điểm

- Kết quả được phân loại như sau

Tổng điểm TĐ

Kết quả

$TĐ > 70$

A

$50 < TĐ \leq 70$

B

$30 < TĐ \leq 50$

C

$TĐ \leq 30$

D

Xây dựng các ca kiểm thử dựa trên:

1. PP phân hoạch tương đương
2. PP phân tích giá trị biên
3. PP bảng hỗ trợ quyết định

a. PP phân hoạch tương đương

- Bảng phân hoạch

Đầu vào	Điểm thi	Điểm thành phần	Điểm tổng	Kết quả
Vùng hợp lệ	[0,75] (H1)	[0,25] (H2)	TĐ>70	A
			50<TĐ<=70	B
			30< TĐ<=50	C
			TĐ<=30	D
Vùng không hợp lệ	<0 (K1)	<0 (K3)		
	>75 (K2)	>25 (K4)		

a. PP phân hoạch tương đương

- Các ca kiểm thử

TC	Inputs		E. Output	Cover
	Điểm thi	Điểm TP		
1	60	15	75 A	H1,H2
2	30	20	60 B	H1,H2
3	20	20	35 C	H1,H2
4	10	20	20 D	H1,H2
5	60	-10	Không hợp lệ	H1,K3
6	60	30	Không hợp lệ	H1,K4
7	-60	20	Không hợp lệ	K1, H2
8	80	20	Không hợp lệ	K2, H2

b. Phân tích giá trị biên

TC	Inputs		E. Output	TC	Inputs		E. Output
	Điểm thi	Điểm TP			Điểm thi	Điểm TP	
1	-1	0	Ko hợp lệ	13	0	1	1 D
2	-1	25	Ko hợp lệ	14	13	16	29 D
3	75	26	Ko hợp lệ	15	14	16	30 D
4	76	25	Ko hợp lệ	16	15	16	31 C
5	75	-1	Ko hợp lệ	17	20	29	49 C
6	76	0	Ko hợp lệ	18	21	29	50 C
7	0	26	Ko hợp lệ	19	22	29	51 B
8	0	-1	Ko hợp lệ	20	49	20	69 B
9	0	0	0 D	21	50	20	70 B
10	0	25	25 D	22	51	20	71 A
11	75	0	75 A	23	75	24	99 A
12	75	25	100 A				

c. Bảng quyết định

[illegible]

Bài 6. Kiểm thử tự động

6.1. Tổng quan về kiểm thử tự động

6.2. Quy trình kiểm thử tự động

6.3. Ưu, nhược điểm của kiểm thử tự động

6.4. Công cụ tự động QTP, Junit, Selenium, Robotium...

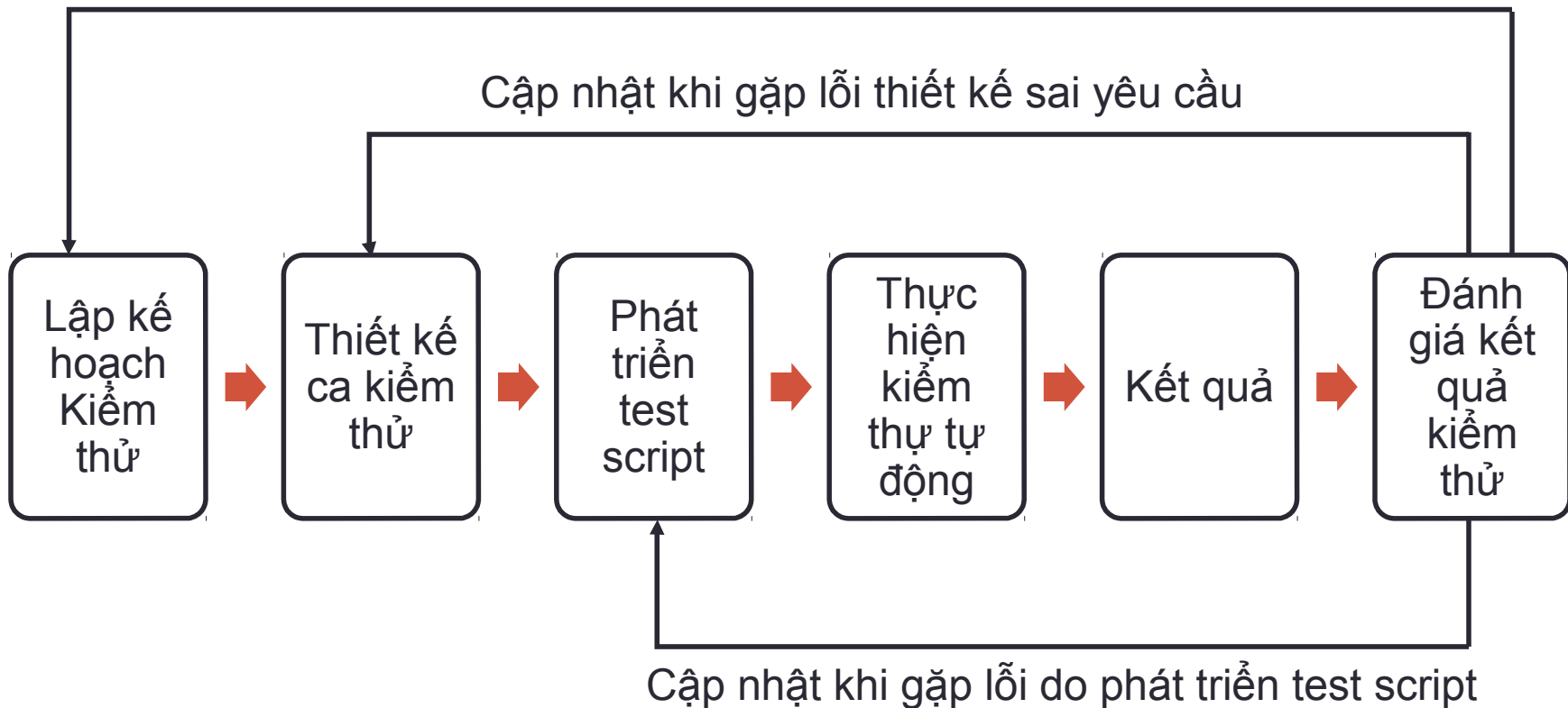
6.1. Tổng quan kiểm thử tự động

- Kiểm thử tự động: áp dụng các công cụ giúp thực hiện việc kiểm thử phần mềm.
- Nên sử dụng công cụ tự động khi:
 - Không đủ tài nguyên
 - Kiểm thử hồi quy
 - Kiểm tra khả năng vận hành của phần mềm trong môi trường đặc biệt.
- Test script: nhóm mã lệnh đặc tả kịch bản dùng để tự động hóa một trình tự kiểm thử.
- Test script: có thể tạo thủ công hoặc tạo tự động dùng công cụ kiểm thử tự động

6.2. Quy trình kiểm thử tự động

Cập nhật khi kiểm thử chưa thỏa mức độ bao phủ yêu cầu phần mềm

Cập nhật khi gặp lỗi thiết kế sai yêu cầu



5.2. Quy trình kiểm thử tự động

1. *Tạo test script*

- Giai đoạn này ta dùng test tool để ghi lại các thao tác lên PM cần kiểm tra và tự động sinh ra test script

2. *Chỉnh sửa test script*

- chỉnh sửa lại test script thực hiện kiểm tra theo đúng yêu cầu đặt ra, cụ thể là làm theo test case cần thực hiện

3. *Chạy test script để kiểm thử tự động*

- Giám sát hoạt động kiểm tra phần mềm của test script

4. *Đánh giá kết quả*

- Kiểm tra kết quả thông báo sau khi thực hiện kiểm thử tự động. Sau đó bổ sung, chỉnh sửa những sai sót

6.3. Ưu, nhược điểm của kiểm thử tự động

- Ưu điểm:

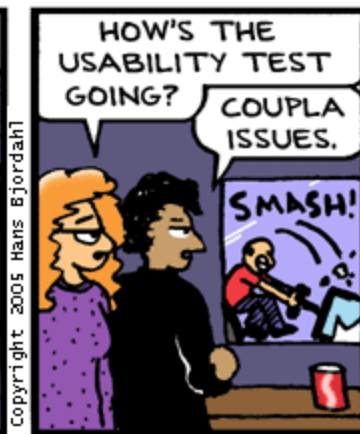
- Kiểm thử phần mềm không cần can thiệp của tester
- Giảm chi phí thực hiện kiểm tra số lượng lớn các test case hoặc test case lặp lại nhiều lần
- Giả lập tình huống khó có thể thực hiện bằng tay

- Nhược điểm:

- Mất chi phí tạo các script để thực hiện kiểm thử tự động
- Tốn chi phí dành cho bảo trì các script
- Đòi hỏi tester phải có kỹ năng tạo và thay đổi script cho phù hợp test case
- Không áp dụng tìm được các lỗi mới cho phần mềm

5.4. Kiểm thử tự động với JUnit

- <Sinh viên thuyết trình>



Copyright 2005 Hans Bjordahl

Bug Bash by Hans Bjordahl

<http://www.bugbash.net/>



Copyright 2005 Hans Bjordahl

Bug Bash by Hans Bjordahl

<http://www.bugbash.net/>



Bug Bash by Hans Bjordahl



Copyright 2005 Hans Bjordahl

<http://www.bugbash.net/>



Bug Bash by Hans Bjordahl

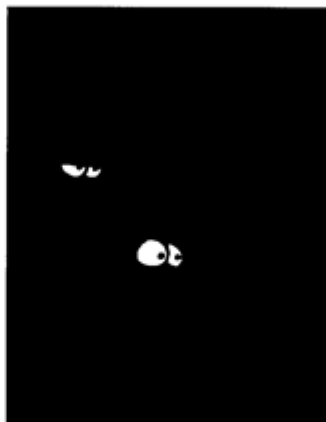


Copyright 2005 Hans Bjordahl

<http://www.bugbash.net/>



Bug Bash by Hans Bjordahl



Copyright 2005 Hans Bjordahl



<http://www.bugbash.net/>

Are you sure you're sure?



Bug Bash by Hans Bjordahl



Copyright 2005 Hans Bjordahl

<http://www.bugbash.net/>

Câu hỏi kiểm tra

1. Khái niệm kiểm thử đơn vị. Nêu mối quan hệ giữa kiểm thử đơn vị và tạo lập mã nguồn.
2. Tại sao kiểm thử dữ liệu qua giao diện lại có đặc trưng liên quan đến định dạng và số lượng.
3. Nêu khái niệm cuống và bánh lái. Khi nào cần sử dụng cuống và bánh lái, lấy ví dụ.

- Khái niệm phần mềm, chất lượng phần mềm
- Trình bày về lỗi, sai sót, hỏng hóc, phân tích và minh họa các nguyên nhân gây ra lỗi
- Trình bày về quy trình kiểm thử, các sản phẩm của quy trình kiểm thử
- Trình bày sự khác biệt của kiểm thử hộp đen và hộp trắng, lấy các ví dụ minh họa