TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

CHƯƠNG 4

Kiểm thử và bảo trì phần mềm

Nghệ An, 2021

📖 Nội dung giảng dạy

4.1. Giới thiệu

4.2. Các chiến lược kiểm thử

4.3. Bảo trì phần mềm

4.1. Giới thiệu

- Kiểm thử phần mềm:
  - Khái niệm
    - Kiểm thử phần mềm là quá trình thực thi phần mềm với mục tiêu tìm ra lỗi (Glen Myers, 1979)
    - Khẳng định được chất lượng của phần mềm đang xây dựng (Hetzel, 1988)
  - Kiểm tra tính chính xác, an toàn, bảo mật và riêng tư của phần mềm bằng cách chạy thử từng thành phần hoặc cả hệ thống để so sánh kết quả thực tế với lý thuyết nhằm mục đích phát hiện lỗi (nếu có).

📖 Đặc điểm kiểm thử phần mềm

- Mỗi phép kiểm thử chỉ thành công khi
  - Một lỗi được phát hiện
  - Một kết quả chỉ ra sự thất bại của thủ tục kiểm thử được trả lại
- Điều gì xảy ra nếu việc kiểm thử không tìm được lỗi trong phần mềm hoặc phát hiện quá ít lỗi
  - Phần mềm có chất lượng quá tốt
  - Quy trình/Đội ngũ kiểm thử hoạt động không hiệu quả

📖 Đặc điểm kiểm thử phần mềm

- Tại sao kiểm thử lại cần thiết:
  - Nhiều phần mềm không hoạt động như mong muốn → lãng phí tiền bạc, thời gian, uy tín của doanh nghiệp.
  - Kiểm thử có thể đem lại sự tin tưởng đối với chất lượng phần mềm nếu có ít lỗi hoặc không có lỗi nào được tìm thấy.
  - Nếu lỗi tìm thấy và được sửa thì chất lượng phần mềm càng được tăng → Giảm chi phí trong quá trình phát triển, nâng cấp, bảo trì phần mềm.

Đặc điểm kiểm thử phần mềm

- Chi phí sửa lỗi
  - Chi phí cho việc tìm thấy và sửa lỗi tăng dần trong suốt chu kỳ sống của phần mềm. Lỗi tìm thấy càng sớm thì chi phí để sửa càng thấp và ngược lại.

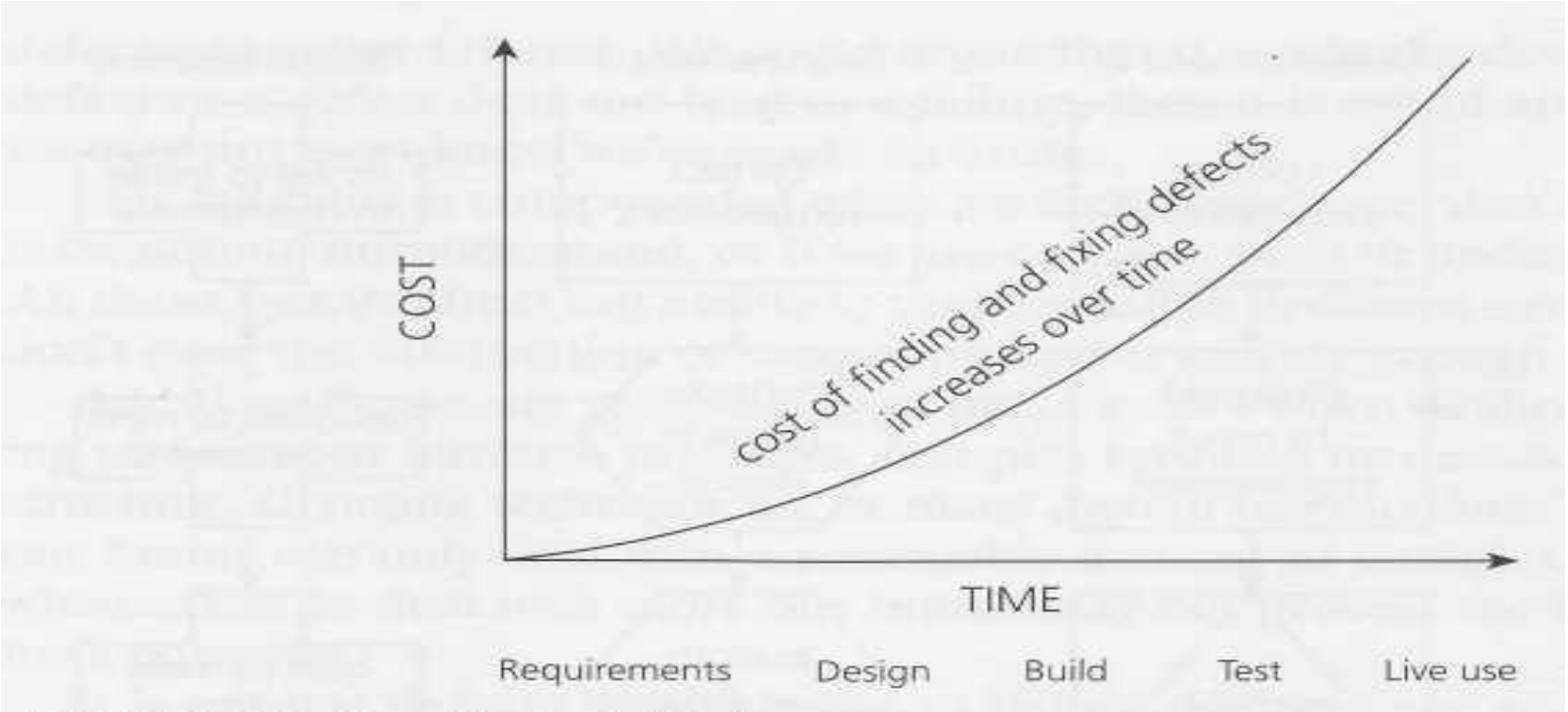


FIGURE 1.2 Cost of defects

Vai trò kiểm thử

- Vai trò kiểm thử trong suốt quy trình sống của phần mềm
  - Kiểm thử không tồn tại độc lập.
  - Các hoạt động của kiểm thử luôn gắn liền với các hoạt động phát triển phần mềm.
  - Các mô hình phát triển phần mềm khác nhau cần các cách tiếp cận kiểm thử khác nhau.

Các khái niệm

- Một sai sót (error) là một sự nhầm lẫn hay một sự hiểu sai trong quá trình phát triển phần mềm của người phát triển.
- Một lỗi (fault, defect) xuất hiện trong phần mềm như là kết quả của một sai sót.
- Một hỏng hóc (failure) là kết quả của một lỗi xuất hiện làm cho chương trình không hoạt động được hay hoạt động nhưng cho kết quả không như mong đợi.

Các khái niệm

- Dữ liệu thử (test data): dữ liệu vào cần cung cấp cho phần mềm trong khi thực thi
- Kịch bản kiểm thử (test scenario): các bước thực hiện khi khi kiểm thử
- Phán xét kiểm thử (test oracle): đánh giá kết quả của kiểm thử
  - Tự động: chương trình
  - Thủ công: con người
- Kiểm thử viên (tester): người thực hiện kiểm thử
- Ca kiểm thử (test case):
  - Tập dữ liệu thử
  - Điều kiện thực thi
  - Kết quả mong đợi

Các khái niệm

- Kiểm chứng (Verification)
  - Có đúng đặc tả không, có đúng thiết kế không
  - Phát hiện lỗi lập trình
- Thẩm định (Validation)
  - Có đáp ứng nhu cầu người dùng không
  - Có hoạt động hiệu quả không
  - Phát hiện lỗi phân tích, lỗi thiết kế (lỗi mức cao)
- Verification and Validation
  - Mục tiêu là phát hiện và sửa lỗi phần mềm, đánh giá tính dùng được của phần mềm, tạo sự tự tin về phần mềm đạt được mục tiêu đề ra
- Thứ tự thực hiện: Verification → Validation

Các khái niệm

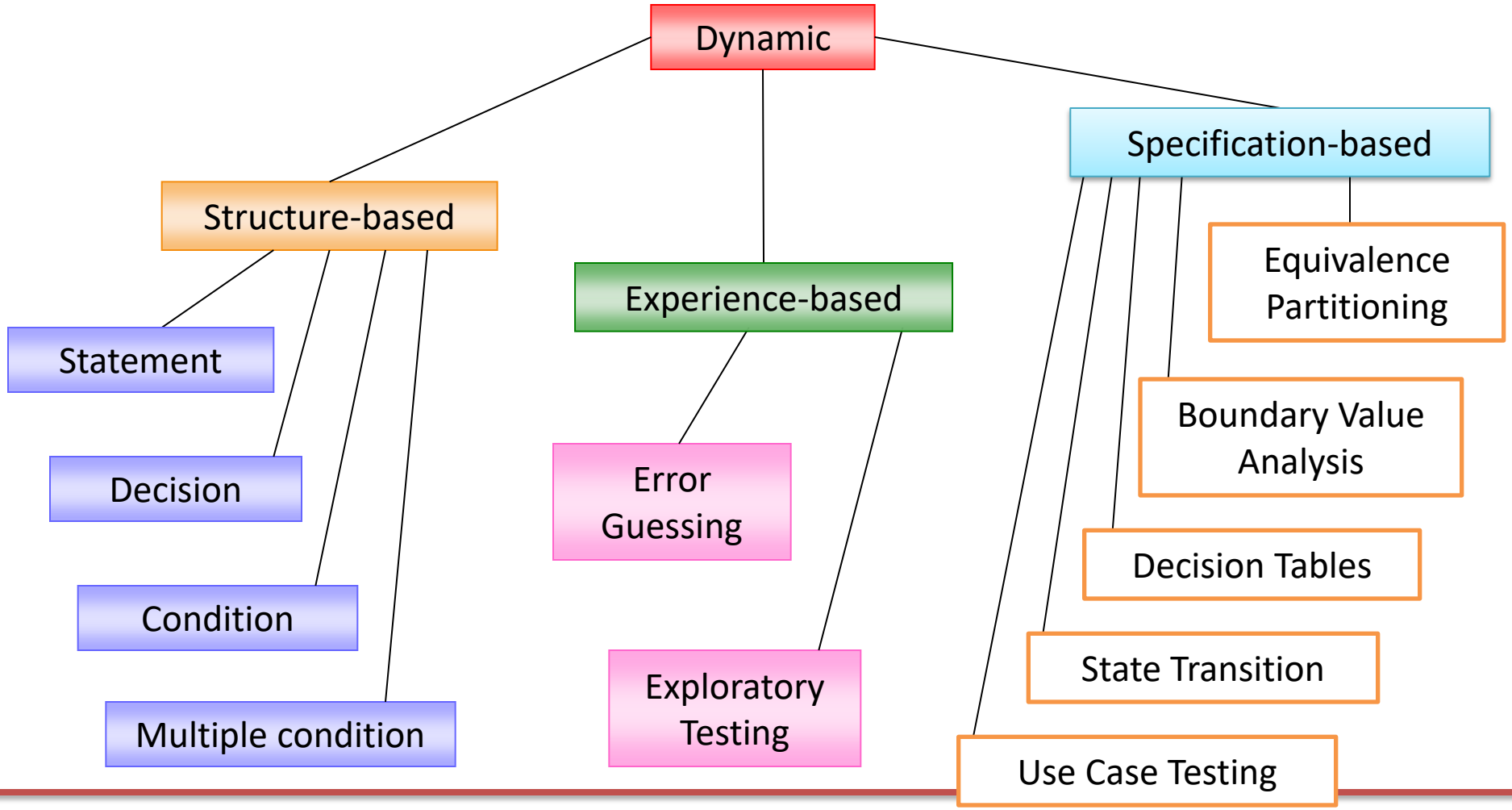
- Kiểm chứng, thẩm định tĩnh
  - Không thực hiện chương trình
  - Dựa vào việc kiểm tra tài liệu, source code…
  - Tiến hành ở mọi công đoạn phát triển phần mềm
  - Đánh giá:
    - Các lỗi được tìm thấy trong quá trình kiểm tra có thể dễ dàng được loại bỏ và chi phí rẻ hơn nhiều so với khi tìm thấy trong test động.
    - Khó đánh giá tính hiệu quả của sản phẩm
  - Một số lợi ích
    - Lỗi sớm được tìm thấy và sửa chữa
    - Giảm thời gian lập trình
    - Giảm thời gian và chi phí test

📖 Các khái niệm

- Kiểm chứng/Thẩm định động
  - Thực hiện chương trình
  - Phát hiện lỗi lập trình
  - Đánh giá tính hiệu quả phần mềm
  - Là cách duy nhất để kiểm tra yêu cầu phi chức năng

Các khái niệm

- Kiểm chứng/Thẩm định động



Các khái niệm

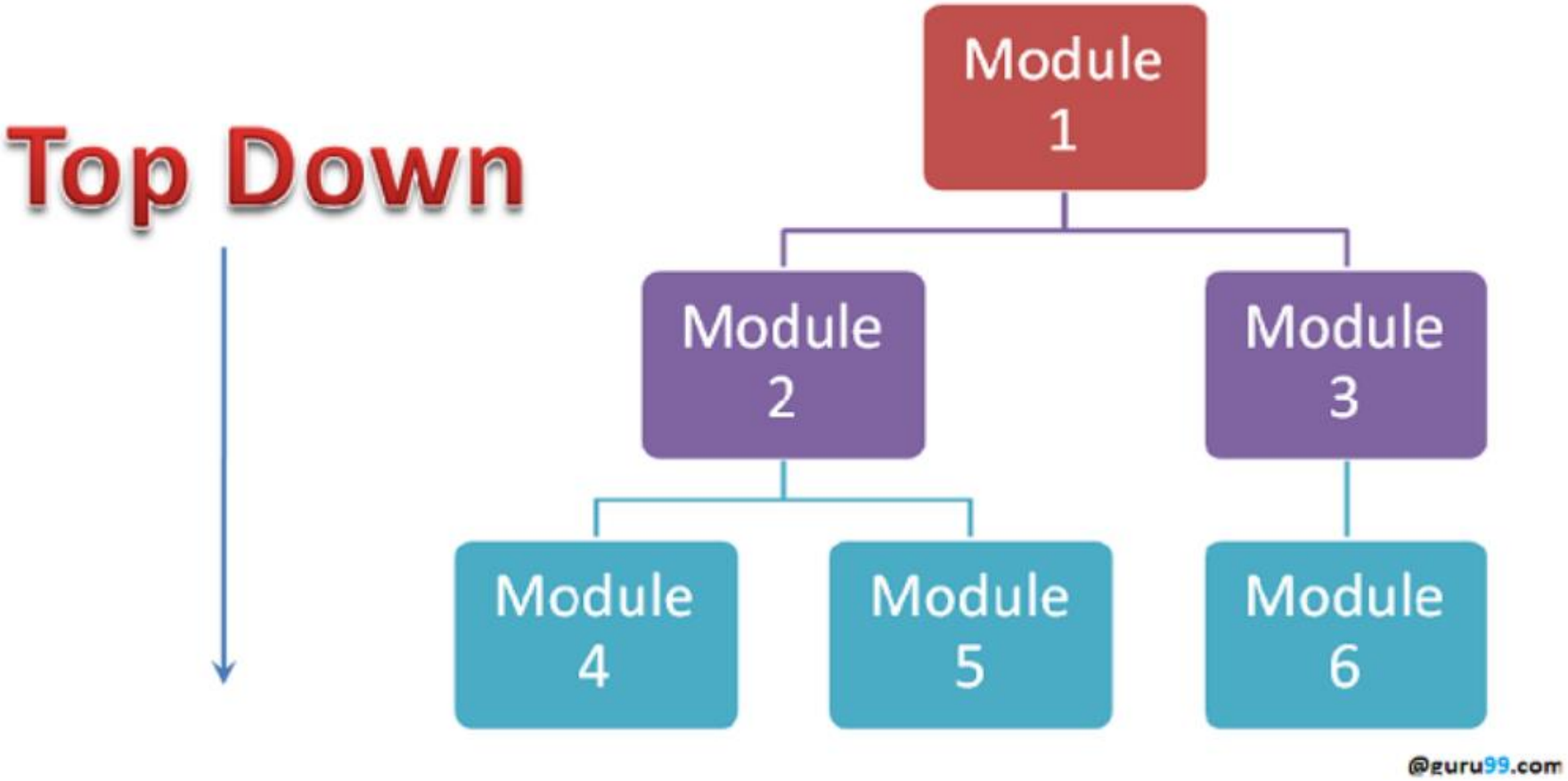
- Kiểm chứng/Thẩm định động
  - Test dựa trên mô tả (specification-based) hay còn gọi test chức năng (functional testing): Test những gì mà phần mềm phải làm, không cần biết phần mềm làm như thế nào (kỹ thuật black box)
  - Test dựa trên cấu trúc (structure-based) hay còn gọi test phi chức năng (non-functional testing): Test phần mềm hoạt động như thế nào (kỹ thuật white box)
  - Test dựa trên kinh nghiệm (experience-based): đòi hỏi sự hiểu biết, kỹ năng và kinh nghiệm của người test

4.2. Các chiến lược kiểm thử

- Chiến lược big bang
  - Tất cả các thành phần được tích hợp cùng một lúc, sau đó tiến hành kiểm thử.
- Chiến lược tăng dần:
  - Ghép hai hoặc nhiều module có liên quan đến logic. Sau đó, các module liên quan khác được thêm vào và kiểm thử chức năng thích hợp.
  - Quá trình tiếp tục cho đến khi tất cả các module được thêm và hoàn thành quá trình kiểm thử.
  - Các dạng kiểm thử
    - Kiểm thử từ trên xuống (Top-Down)
    - Kiểm thử từ dưới lên (Bottom-Up)
- Chiến lược kết hợp trên xuống và dưới lên (Sandwich)
- Kiểm thử hồi quy

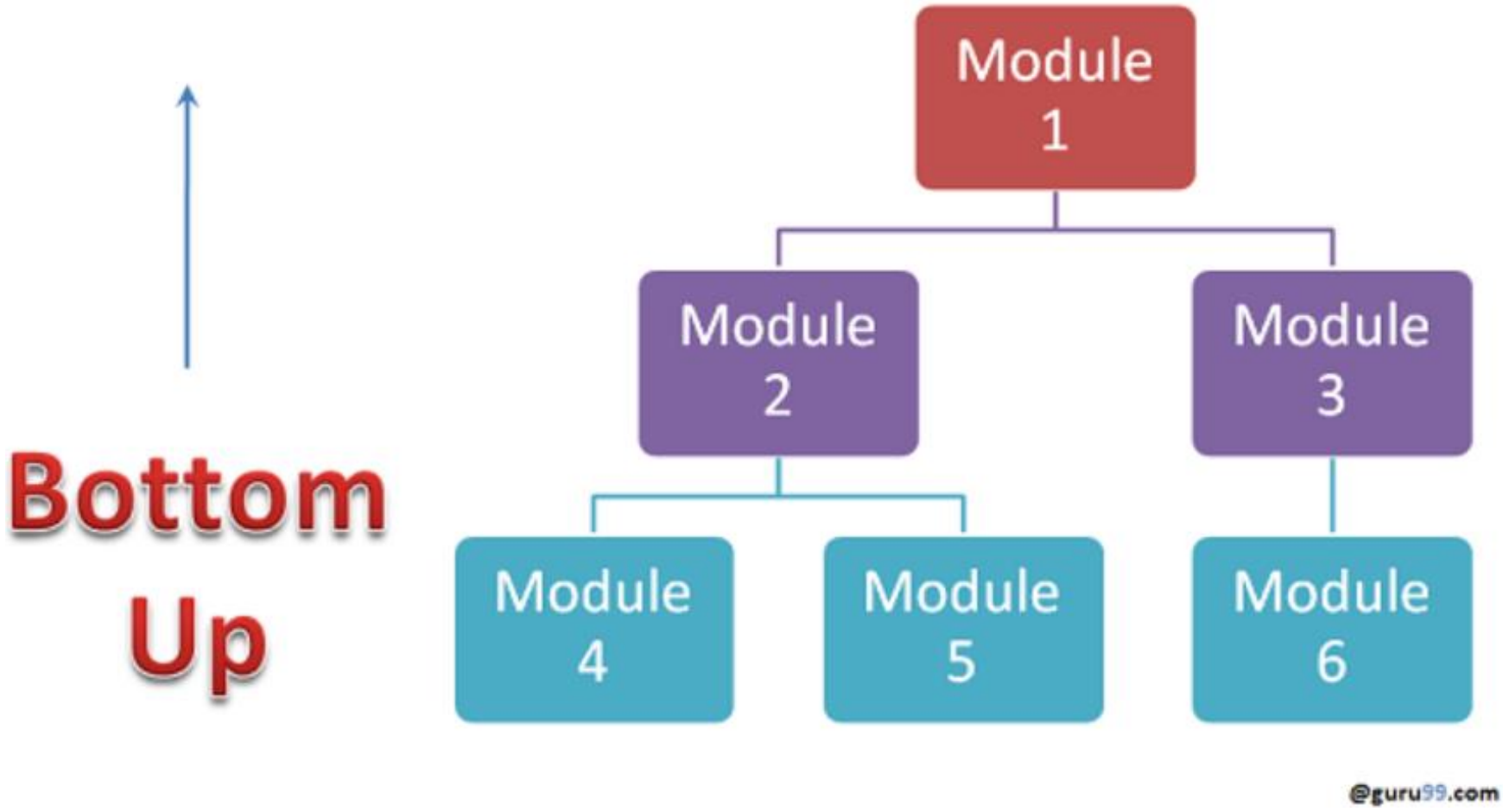
Các chiến lược kiểm thử

- Kiểm thử từ trên xuống



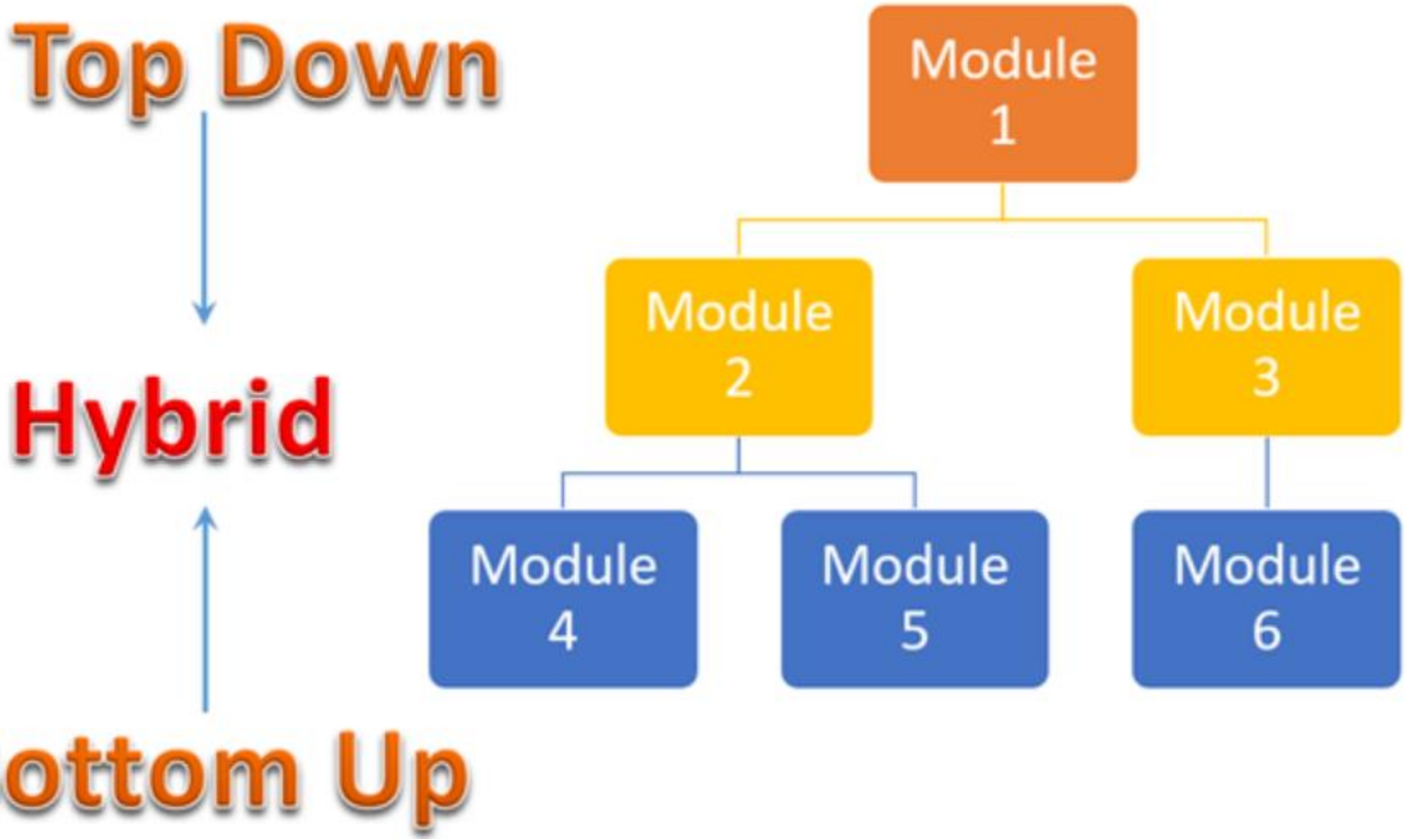
Các chiến lược kiểm thử

- Kiểm thử từ dưới lên



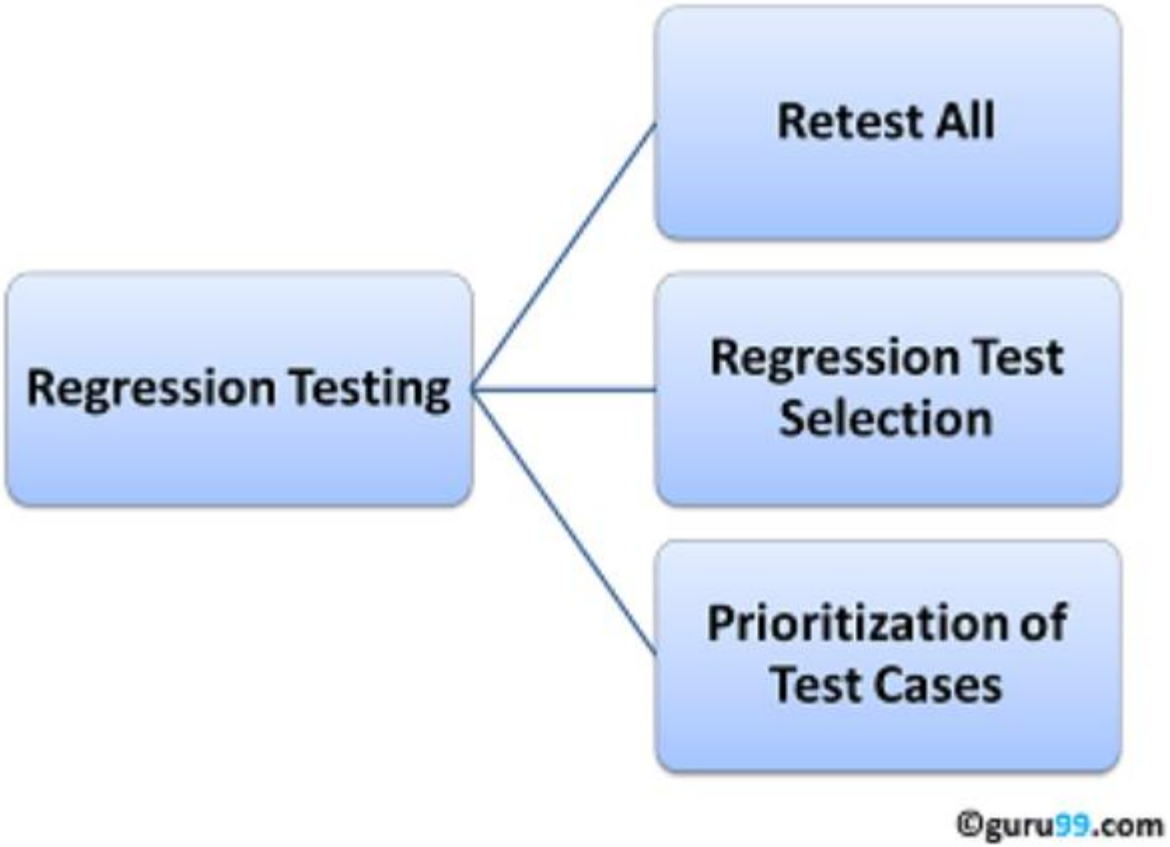
Các chiến lược kiểm thử

- Kiểm thử kết hợp trên xuống và dưới lên



Các chiến lược kiểm thử

- Kiểm thử hồi quy



Các vấn đề cần kiểm thử

- Tính chính xác của hệ thống thể hiện ở chỗ hệ thống làm việc luôn luôn đúng đắn và dữ liệu dùng trong hệ thống là xác thực và phi mâu thuẫn.
- Tính an toàn của hệ thống thể hiện ở chỗ hệ thống không bị xâm hại hay bị xâm hại không nhiều khi xảy ra sự cố kỹ thuật.
- Tính bảo mật của hệ thống thể hiện ở chỗ hệ thống có khả năng ngăn ngừa các xâm phạm vô tình hay cố ý từ phía con người.
- Tính riêng tư của hệ thống thể hiện ở chỗ hệ thống bảo đảm được các quyền truy nhập riêng tư đối với các loại người dùng khác nhau.

Các nguyên tắc đảm bảo

- Tính chính xác: kiểm tra các thông tin nhập và xuất.
  - Áp dụng các hình thức kiểm tra như tự động/bằng tay, đầy đủ/chọn đặc trưng, trực tiếp/gián tiếp.
- Tính an toàn: đảm bảo sự an toàn của thông tin là quan trọng nhất, sử dụng các cách sau:
  - Khóa từng phần cơ sở dữ liệu: khóa bản gốc và tiến hành việc cập nhật trên bản sao, việc thay thế dữ liệu mới chỉ được thực hiện khi thao tác cập nhật trên bản sao được thực hiện hoàn tất.
  - Sử dụng các tệp sao lưu:
    - Tệp nhật ký: là một tệp tuần tự chứa các bản sao của các đơn vị cơ sở dữ liệu trước và sau khi chúng được cập nhật.
    - Tệp lưu: chứa bản sao toàn bộ hoặc một phần của cơ sở dữ liệu được thực hiện theo chu kỳ.

Các nguyên tắc đảm bảo

- Tính an toàn:
  - Thực hiện các thủ tục phục hồi: thủ tục phục hồi nhằm đưa cơ sở dữ liệu trở về trạng thái đúng đắn trước khi bị hỏng vì sự gián đoạn chương trình (hư hỏng phần cứng, chương trình bị treo…).
  - Cân nhắc giữa việc chạy lại từ đầu và chạy từ chỗ bị ngắt.
- Tính bảo mật: nhận diện các điểm hở và dự đoán các mối đe dọa tiềm ẩn từ các điểm hở này.
  - Các điểm hở là chỗ mà tác nhân bên ngoài vô tình hay cố ý có thể gây ra tác động tiêu cực cho hệ thống.
  - Các điểm hở có thể là:
    - Thủ tục vào/ra.
    - Kho dữ liệu.
    - Đường truyền.

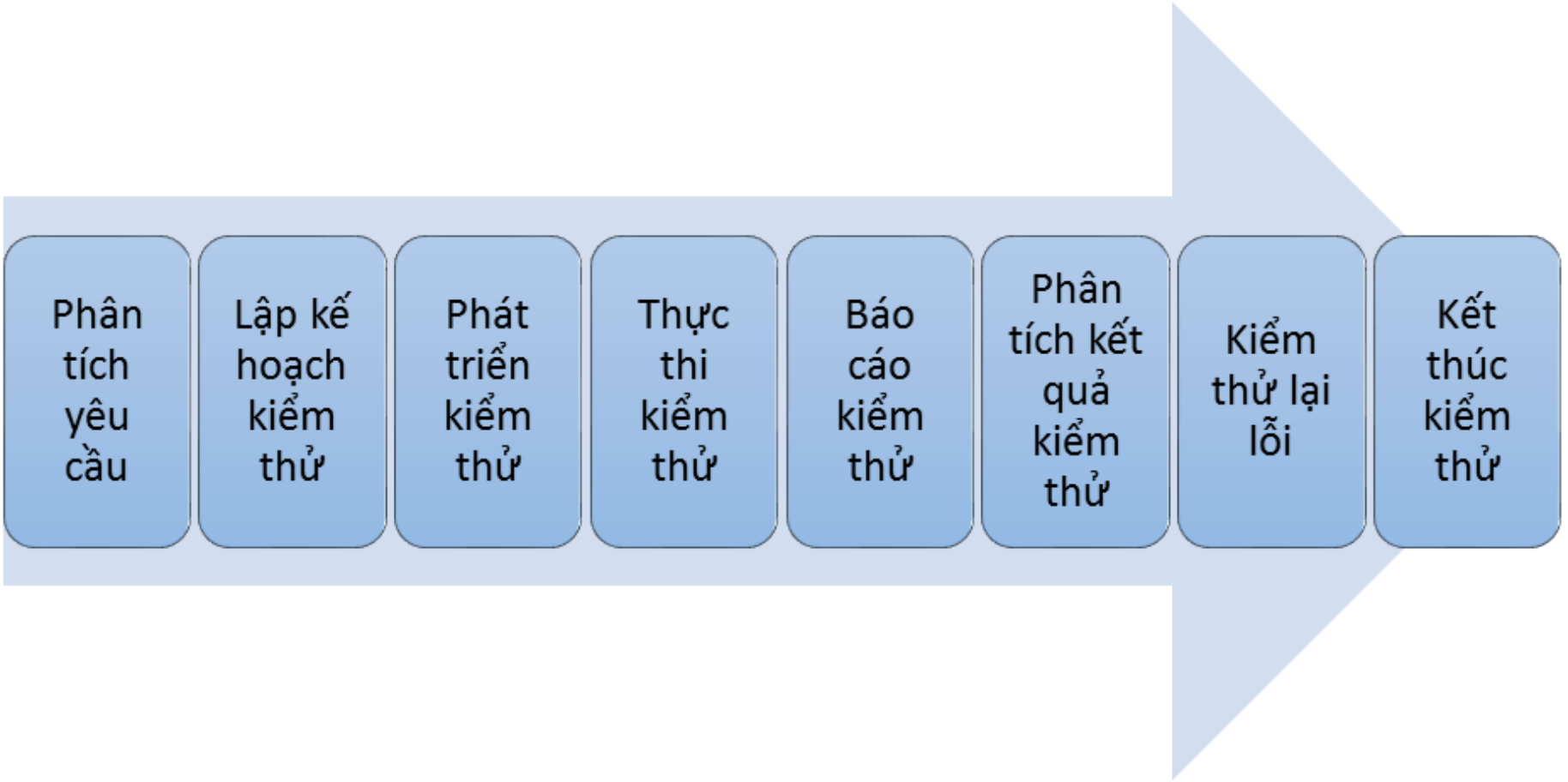
Các nguyên tắc đảm bảo

- Tính bảo mật:
  - Các biện pháp bảo mật:
    - Bảo mật vật lý
    - Tài khoản người dùng
    - Mã hóa
    - Truy nhập gián tiếp
    - Tường lửa
- Tính riêng tư: phân loại người dùng để:
  - Gán cho mỗi loại người dùng một số quyền truy nhập nhất định.
    - Các quyền truy nhập: read, insert, update, delete, expand, drop, index, run.
  - Cho phép một số người dùng được phép ủy quyền (giao quyền truy nhập cho người khác).

Quy trình kiểm thử phần mềm (test process)

- Các bước trong quy trình kiểm thử phần mềm:
  - Phân tích yêu cầu: xác định phạm vi test.
  - Lập kế hoạch kiểm thử: chiến lược kiểm thử (test strategy), test plan.
  - Phát triển kiểm thử: viết test procedure, test scenario, test case, test data và test script.
  - Thực thi kiểm thử: tester thực thi phần mềm dựa trên test plan và test case.
  - Báo cáo kiểm thử: tester điền kết quả test vào test case và tạo báo cáo kết quả test.
  - Phân tích kết quả kiểm thử: hoặc còn gọi là phân tích lỗi để quyết định lỗi nào sẽ được sửa và lỗi nào sẽ không sửa.
  - Kiểm thử lại lỗi: sau khi một lỗi (defect) được dev sửa xong, chuyển phần mềm cho tester test lại.
  - Kết thúc kiểm thử: khi test đã đáp ứng được điều kiện dừng, từ đó rút ra các bài học kinh nghiệm.

Quy trình kiểm thử phần mềm (Test Process)



Quy trình kiểm thử phần mềm (Test Process)

- Khi nào nên dừng kiểm thử:
  - Rất khó để xác định khi nào thì dừng kiểm thử, kiểm thử là một quá trình có thể không bao giờ kết thúc. Vì vậy cần phải có 1 vài yếu tố để dựa vào đó và xác định việc dừng kiểm thử:
    - Đáp ứng yêu cầu của khách hàng.
    - Hoàn thành toàn bộ kịch bản kiểm thử và đóng hết lỗi.
    - Hoàn thành kiểm thử các chức năng và đạt được mức độ, tiêu chí hoàn hành kiểm thử.
    - Tỉ lệ lỗi giảm và không còn các lỗi có độ ưu tiên cao.
    - Hết ngân sách dành cho kiểm thử.
    - Quyết định của quản lý.

Bộ kiểm thử

- Các nguyên lý kiểm thử
  - Các phép kiểm thử phải tương ứng với các yêu cầu của hệ thống.
  - Mỗi phép kiểm thử nên được lập kế hoạch từ rất sớm trước khi tiến hành kiểm thử.
- Bộ kiểm thử
  - Bộ kiểm thử: dữ liệu dùng để kiểm tra hoạt động của chương trình
  - Bộ kiểm thử tốt: có khả năng bao phủ được các trường hợp có thể xảy ra lỗi.

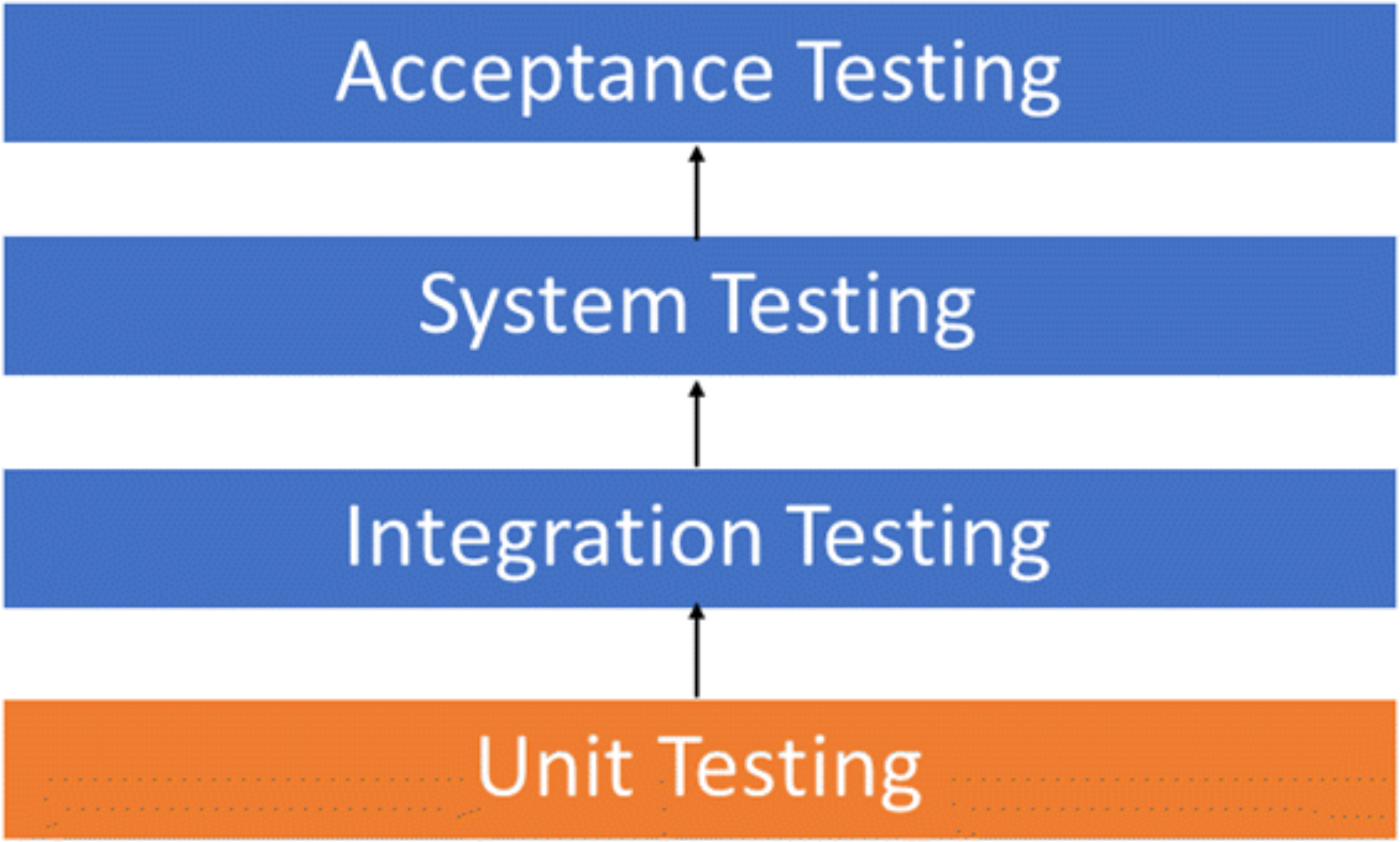
Bộ kiểm thử

- Nội dung của bộ kiểm thử
  - Tên module/chức năng muốn kiểm thử dữ liệu vào
    - Dữ liệu thông thường: số, xâu kí tự, file...
    - Môi trường thử nghiệm: phần cứng, OS...
    - Thứ tự thao tác (khi kiểm thử giao diện)
  - Kết quả mong muốn
    - Thông thường: số, xâu kí tự, file...
    - Màn hình, thời gian phản hồi
  - Kết quả thực tế

Yêu cầu đối với kiểm thử

- Tính hệ thống
  - Phải đảm bảo đã kiểm tra hết các trường hợp.
- Được lập tài liệu
  - Phải lập tài liệu cho quá trình kiểm thử (kiểm soát tiến trình/kết quả).
- Tính lặp lại
  - Phải tiến hành kiểm tra lại để đảm bảo rằng các lỗi đã được sửa chữa.

Tiến trình kiểm thử



Tiến trình kiểm thử

- Kiểm thử đơn vị (unit testing): kiểm tra các thành phần phần mềm riêng lẻ
- Kiểm thử tích hợp (integration testing): tích hợp các thành phần riêng lẻ đã được kiểm tra và kiểm tra các nhóm này
- Kiểm thử hệ thống (system testing): kiểm tra toàn bộ hệ thống như một khối tổng thể
- Kiểm thử chấp nhận (acceptance testing): kiểm tra toàn bộ hệ thống nhìn từ phía khách hàng

Tiến trình kiểm thử

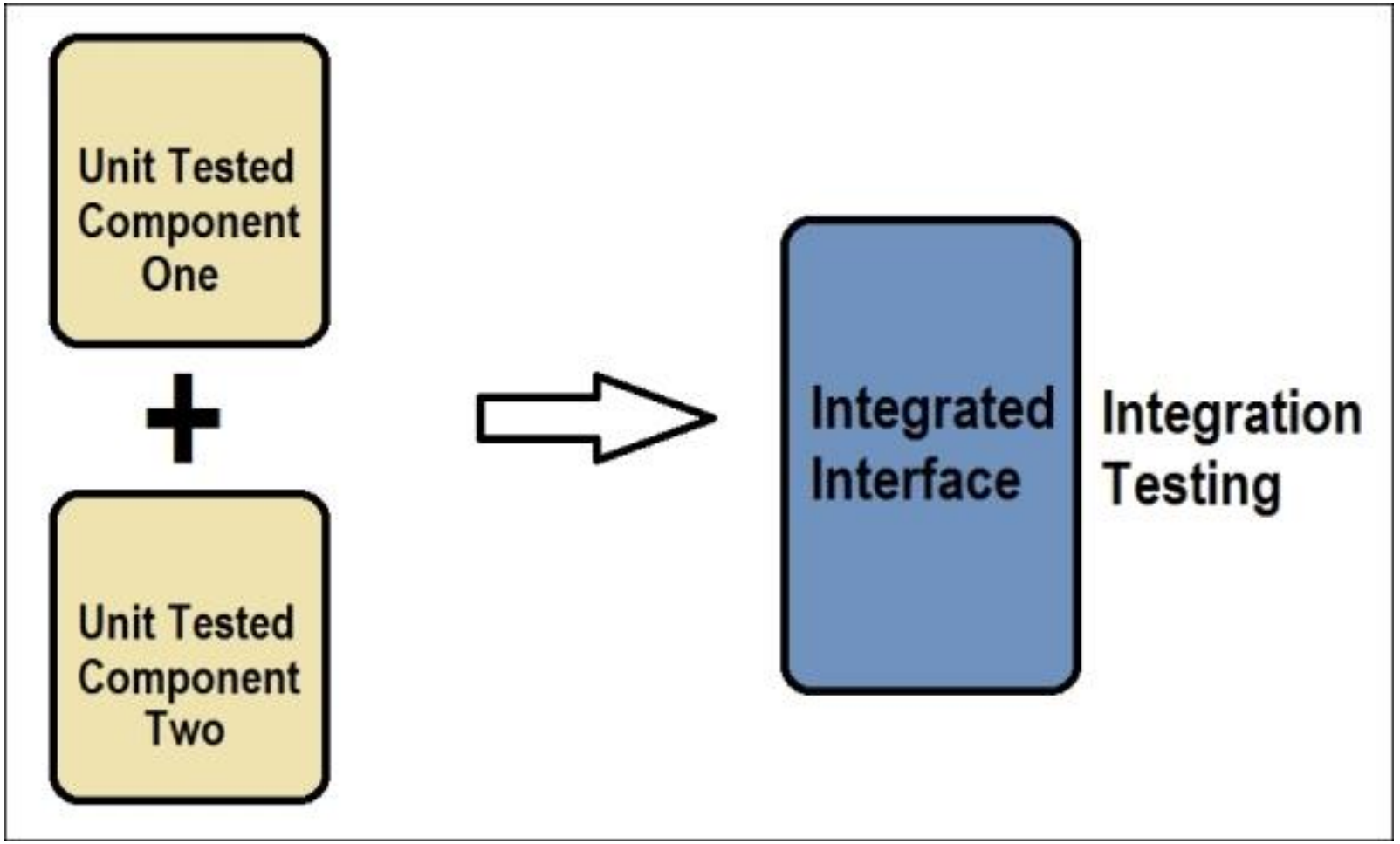
- Kiểm thử đơn vị (Unit testing)
  - Một Unit là một thành phần nhỏ nhất của phần mềm có thể kiểm tra được
    - Lớp
    - Các hàm
    - Thủ tục
    - Phương thức
    - Thực hiện các câu lệnh trong phần mềm
  - Người thực hiện: Lập trình viên
  - Yêu cầu:
    - Đòi hỏi tất cả các nhánh bên trong Unit đều phải được kiểm tra để phát hiện lỗi phát sinh
    - Phải chuẩn bị trước các tình huống kiểm thử trong đó chỉ định rõ dữ liệu vào, các bước thực hiện và dữ liệu mong chờ xuất ra

Tiến trình kiểm thử

- Kiểm thử tích hợp (Intergration testing)
  - Kiểm thử tích hợp các unit lại để ra một trường hợp kiểm thử
  - Ví dụ: Có các chức năng A, B, C, D. Hỏi có bao nhiêu trường hợp có thể kiểm thử? (15 trường hợp)
    - A, B, C, D, AB, AC, AD, BC, BD, CD, ABC, ABD, BCD, ACD, ABCD
  - Khi kiểm thử phải quét hết tất cả các trường hợp
  - Người thực hiện tích hợp: lập trình viên
  - Intergrationg test có 2 mục tiêu chính:
    - Phát hiện lỗi giao tiếp xảy ra giữa các unit không?
    - Tích hợp các unit đơn lẻ thành các hệ thống nhỏ và cuối cùng là hoàn chỉnh hệ thống để chuẩn bị cho kiểm tra ở mức độ hệ thống

Tiến trình kiểm thử

- Kiểm thử tích hợp (Intergration testing)



Tiến trình kiểm thử

- Kiểm thử tích hợp (Intergration testing): có 4 loại kiểm tra
  - Kiểm tra cấu trúc (structure)
    - Tương tự như White Box Testing
    - Nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng
    - Chú trọng đến hoạt động thành phần cấu trúc của chương trình
  - Kiểm tra chức năng (functional)
    - Tương tự Black Box Testing
    - Kiểm tra chỉ chú trọng đến chức năng chương trình

Tiến trình kiểm thử

- Kiểm thử tích hợp (Intergration testing):
  - Kiểm tra hiệu năng (performance test)
    - Kiểm tra các giới hạn (sử dụng tài nguyên) của hệ thống.
    - Phản ứng của hệ thống khi đạt mức tới hạn.
  - Kiểm tra khả năng chịu tải (stress test)
    - Tạo áp lực lên hệ thống để kiểm tra những hoạt động lỗi
    - Là kiểm tra vận hành của hệ thống, ngưỡng tối đa là bao nhiêu (thường kiểm tra hệ thống web)
    - Thích hợp với những hệ thống phân bố mà thể hiện sự giảm sút giá trị nghiêm trọng khi mạng quá tải

Tiến trình kiểm thử

- Kiểm thử hệ thống (System Testing)
  - Mục đích là kiểm tra thiết kế về toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không?
  - Phải thực hiện Unit Test và Integration Test để đảm bảo mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test
  - Đặc điểm
    - Đặc điểm kiểm tra hệ thống
    - Tốn rất nhiều công sức
    - Mất nhiều thời gian
  - 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.

Tiến trình kiểm thử

- Kiểm thử hệ thống (System Testing):
  - Các hoạt động kiểm thử hệ thống
    - Kiểm tra chức năng
    - Kiểm tra hiệu năng
    - Kiểm tra khả năng chịu tải
    - Kiểm tra cấu hình
    - Kiểm tra khả năng bảo mật
    - Kiểm tra khả năng phục hồi
  - Sau giải đoạn System Test, phần mềm thường đã sẵn sàng cho khách hàng hoặc người dùng cuối cùng kiểm tra để chấp nhận sản phẩm

Tiến trình kiểm thử

- Kiểm thử hệ thống (System Testing)
  - Ví dụ: Hệ thống quản lý thông tin sinh viên trường Đại học có các chức năng chính sau:
    - 1. Nhập dữ liệu sinh viên trúng tuyển vào hệ thống
    - 2. Gọi nhập học và phân lớp
    - 3. Quản lý thông tin sinh viên, kết quả học tập, điểm thi
    - 4. Xét tốt nghiệp
  - 1 → 2 → 3 → 4. Test từng chức năng, sau đó kiểm tra theo luồng xem có lỗi không và lặp lại quá trình đó.

Tiến trình kiểm thử

- Kiểm thử hệ thống (System Testing)
  - Lưu ý: Không nhất thiết thực hiện kiểm tra tất cả các loại nêu trên mà sẽ
    - Tùy yêu cầu và đặc trưng của từng hệ thống
    - Tùy khả năng và thời gian cho phép của dự án, khi lập kế hoạch, trưởng dự án sẽ quyết định áp dụng những loại kiểm tra nào.

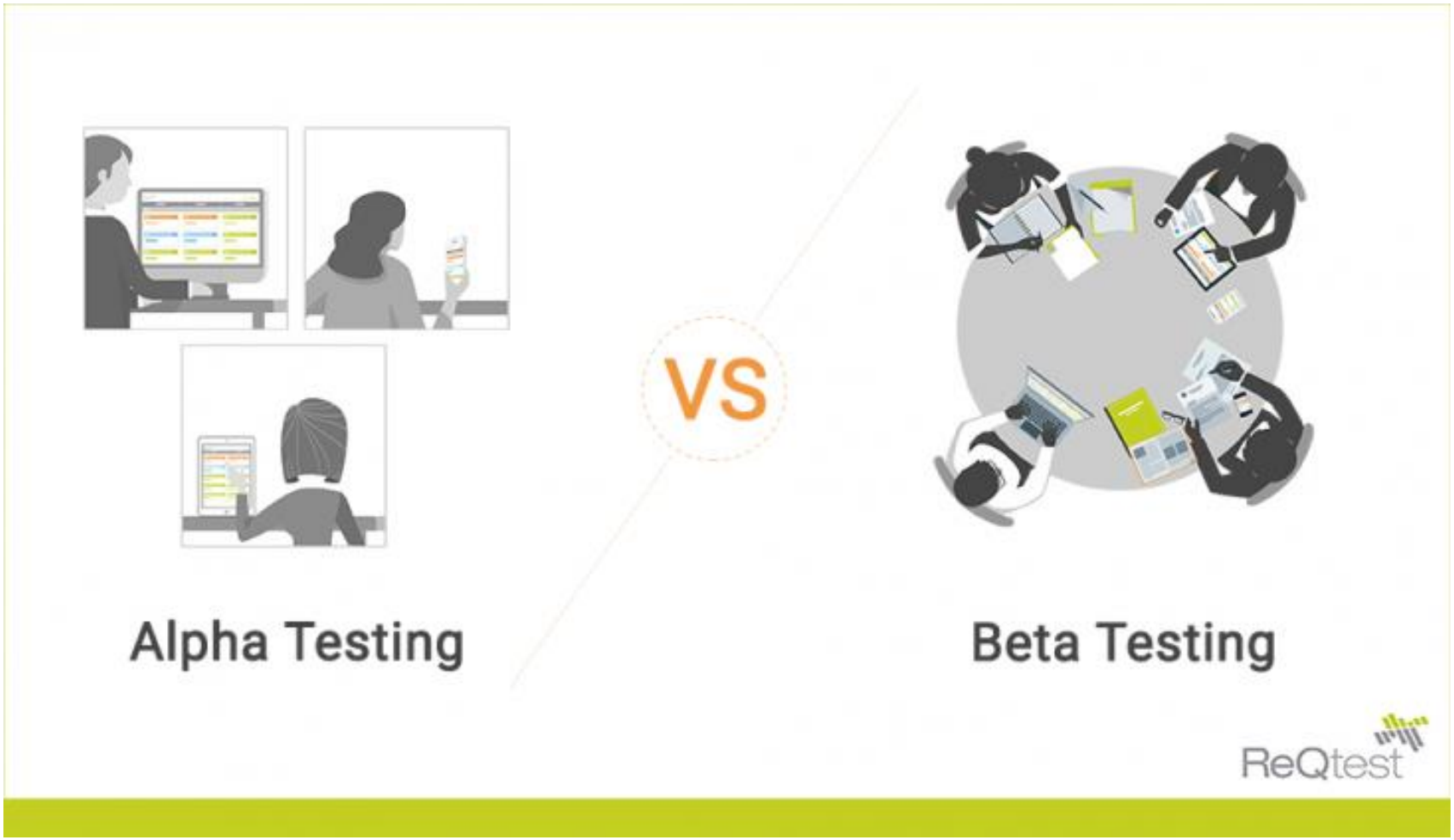
Tiến trình kiểm thử

- Kiểm thử chấp nhận (kiểm thử nghiệm thu)
  - Kiểm thử Alpha
    - Là một hình thức kiểm tra nội bộ, thường thực hiện từ khi phần mềm sắp được hoàn thiện/chuẩn bị bàn giao.
    - Thường được thực hiện bởi người dùng/khách hàng tiềm năng, người phát triển/nhóm kiểm thử độc lập.
    - Sử dụng các dữ liệu thực do user cung cấp
    - Rất quan trọng nhằm hoàn thiện phần mềm trước khi bàn giao cho khách hàng

Tiến trình kiểm thử

- Kiểm thử chấp nhận
  - Kiểm thử Beta (mở rộng của Alpha)
    - Là một hình thức kiểm tra bên ngoài, bước kiểm thử mà về cơ bản các bước phát triển cũng như kiểm thử trước đó đã hoàn thành.
    - Được tiến hành với số lượng lớn user, User tiến hành kiểm thử không có sự hướng dẫn của người phát triển.
    - Quá trình tiến hành thử lần cuối nhằm tìm ra lỗi cũng như kiểm chứng các ràng buộc trước khi công bố và tiến hành phân phối sản phẩm.
    - Chỉ được thực hiện bởi người dùng/khách hàng tiềm năng, không liên quan đến các nhà phát triển (có thể tester ghi nhận lỗi phát sinh để dev xử lý, hoặc lấy phản hồi từ thị trường).

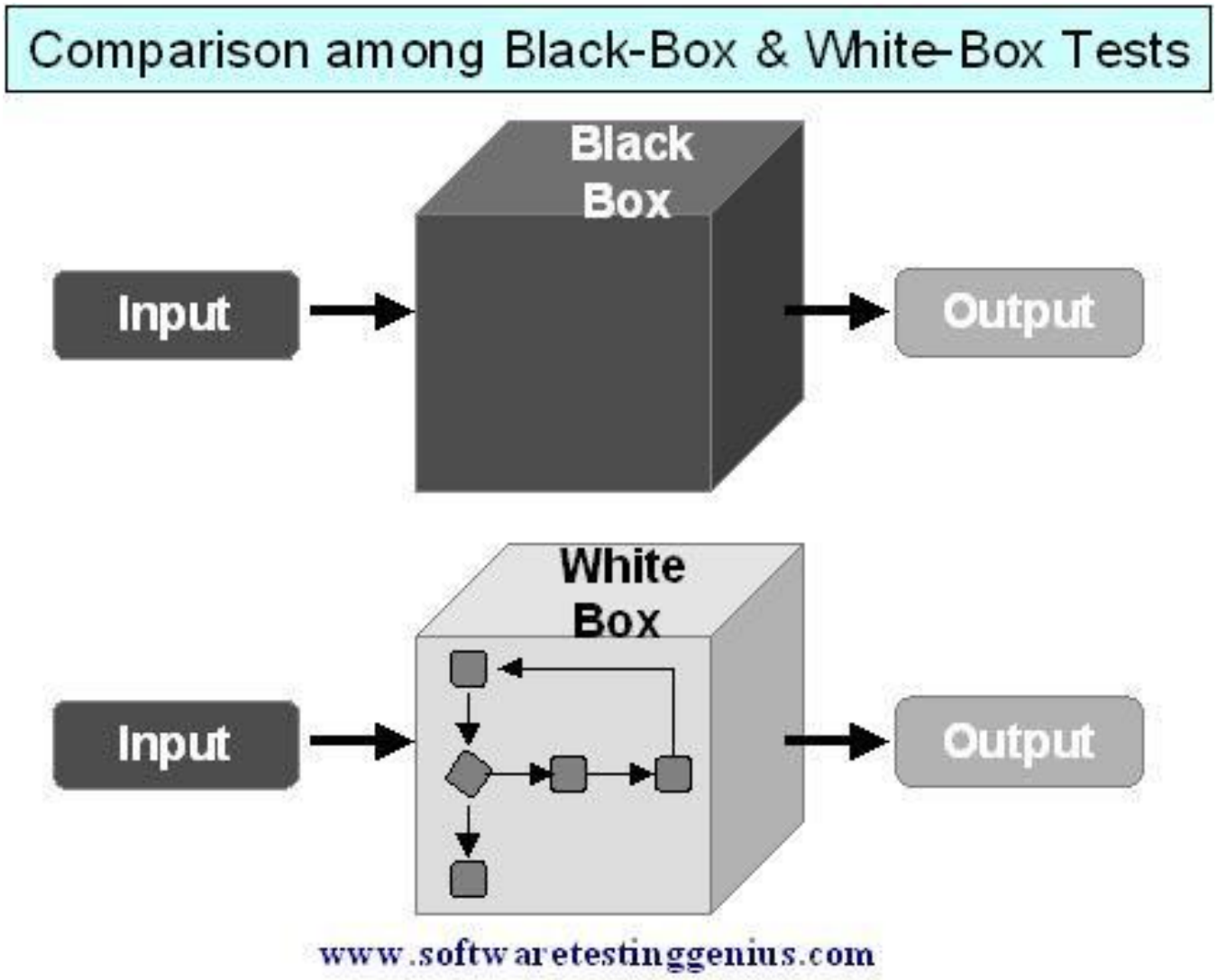
Tiến trình kiểm thử



Các phương pháp kiểm thử

- Các phương pháp kiểm thử:
  - Kiểm thử hộp đen (black box testing)/ kiểm thử chức năng (functional testing)
  - Kiểm thử hộp trắng (white box testing)/ kiểm thử cấu trúc (structured testing)
  - Kiểm thử hôp xám (gray box testing)

🕮 Kiểm thử hộp đen & hộp trắng



Kiểm thử hộp đen

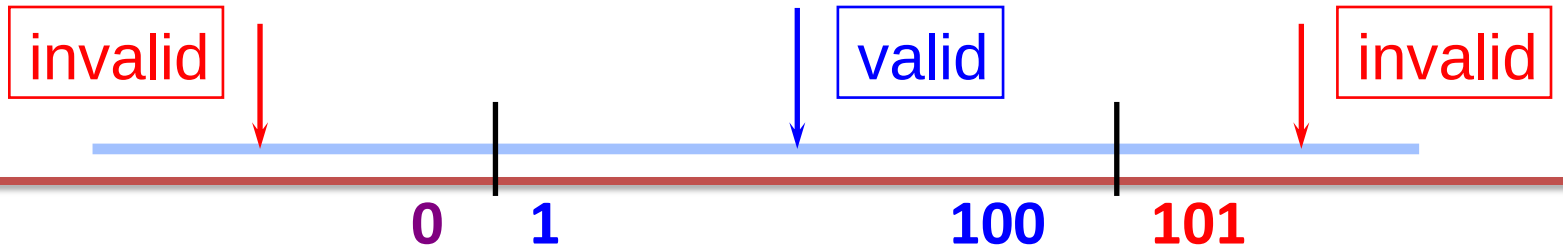
- Dùng để kiểm tra các yêu cầu chức năng của phần mềm.
- Người kiểm thử không quan tâm đến cấu trúc bên trong của chương trình
- Chỉ quan tâm tới dữ liệu đầu vào – đầu ra sau khi được xử lý
- Chỉ tập trung vào kiểm tra chức năng phần mềm
- Không quan tâm đến mã lệnh Black box testing
- Căn cứ vào tài liệu đặc tả, phân tích thiết kế để Test
- Thường do các Tester thực hiện

📖 Kiểm thử hộp đen

- Một số kỹ thuật kiểm thử
  - Kỹ thuật phân vùng tương đương – EP (Equivalence Partitioning)
  - Kỹ thuật phân tích giá trị giới hạn - BVA (Boundary Value Analysis)
  - Kiểm thử ngẫu nhiên (random testing)
  - Đồ thị nhân-quả (cause-efect graph)
  - Kiểm thử cú pháp

Kiểm thử hộp đen

- Kỹ thuật phân vùng tương đương – EP (Equivalence Partitioning)
  - Ý tưởng: Chia đầu vào thành những nhóm tương đương nhau, nếu một giá trị trong nhóm hoạt động đúng thì tất cả các giá trị trong nhóm đó cũng hoạt động đúng và ngược lại.
  - Ví dụ: một textbox chỉ cho phép nhập số nguyên từ 1 đến 100 → không thể nhập tất cả các giá trị từ 1 đến 100
  - → Chọn 3 test case để test trường hợp này: -5, 55, 102 hoặc 0, 10, 100…



Kiểm thử hộp đen

- Kỹ thuật phân vùng tương đương – EP
  - Tuy nhiên nếu nhập vào số thập phân (55.5) hay một ký tự không phải là số (abc)?
  - Trong trường hợp trên có thể chia làm 5 phân vùng như sau:
    - Các số nguyên từ 1 đến 100
    - Các số nguyên nhỏ hơn 1
    - Các số nguyên lớn hơn 100
    - Không phải số
    - Số thập phân
  - Như vậy, việc phân vùng có đúng và đủ hay không là tùy thuộc vào kinh nghiệm của tester.

Kiểm thử hộp đen

- Kỹ thuật phân tích giá trị giới hạn - BVA (Boundary Value Analysis)
  - Cơ sở: lỗi thường xuất hiện gần các giá trị biên của miền dữ liệu
  - Tập trung phân tích các giá trị biên của miền dữ liệu để xây dựng dữ liệu kiểm thử
  - Nguyên tắc: kiểm thử các dữ liệu vào gồm
    - Giá trị nhỏ nhất
    - Giá trị gần kề lớn hơn giá trị nhỏ nhất
    - Giá trị gần kề nhỏ hơn giá trị lớn nhất
    - Giá trị lớn nhất

Kiểm thử hộp đen

- Kỹ thuật phân tích giá trị giới hạn - BVA (Boundary Value Analysis)

  - Áp dụng kỹ thuật BVA cần 4 test case để test trường hợp này: 0, 1, 100, 101

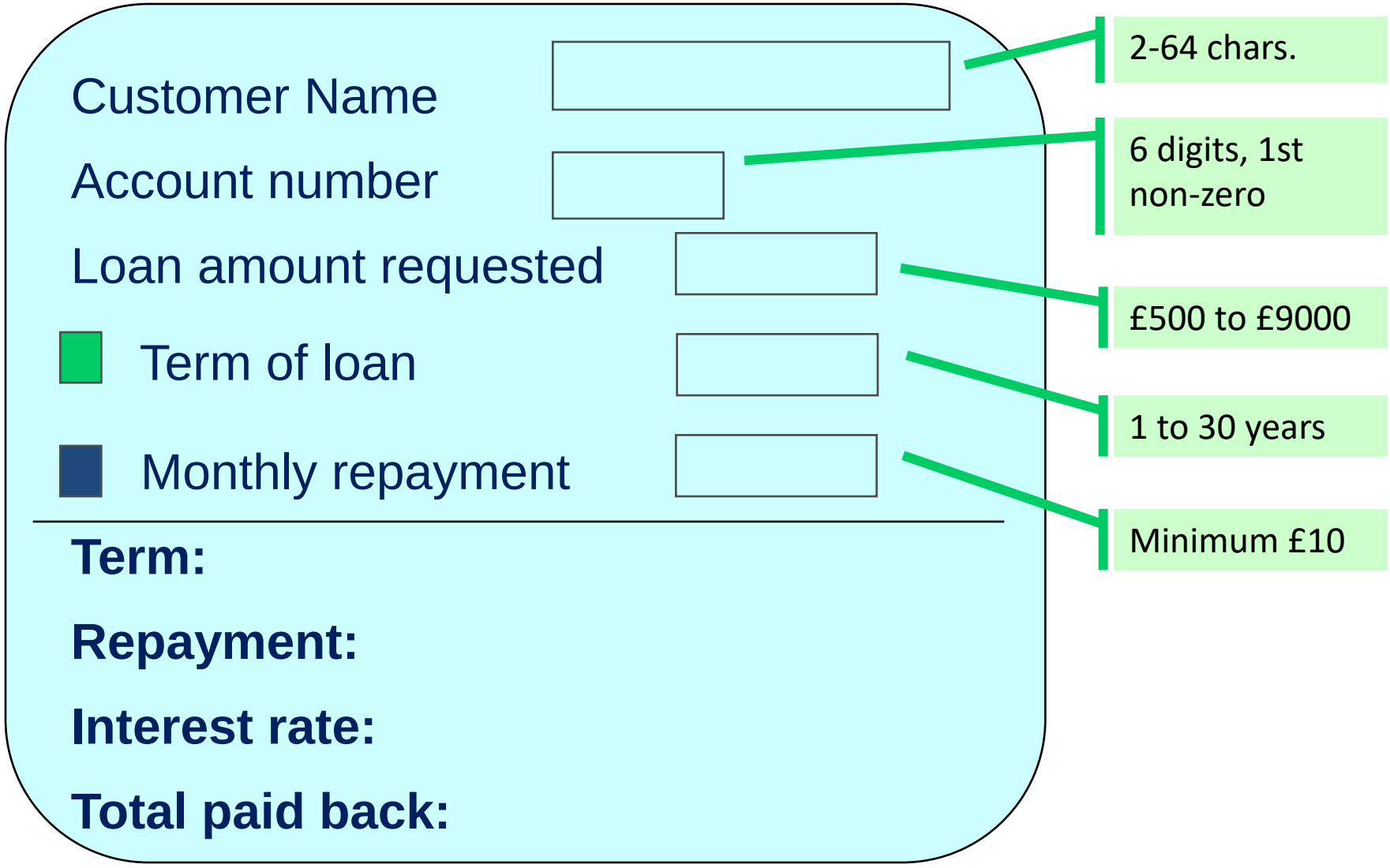
Kiểm thử hộp đen

- Xét ví dụ: Một ngân hàng trả lãi cho khách hàng dựa vào số tiền còn lại trong tài khoản. Nếu số tiền từ 0 đến 100$ thì trả 3% lãi, từ lớn hơn 100 $ đến nhỏ hơn 1000$ trả 5% lãi, từ 1000$ trở lên trả 7% lãi.
  - Dùng kỹ thuật EP:

Invalid partition	Valid (for 3% interest)		Valid (for 5%)		Valid (for 7%)
–$0.01	$0.00	$100.00	$100.01	$999.99	$1000.00

  - Kỹ thuật EP: -0.44, 55.00, 777.50, 1200.00
  - Kỹ thuật BVA: -0.01, 0.00, 100.00, 100.01, 999.99, 1000.00

Kiểm thử hộp đen



Kiểm thử hộp trắng

- Khi thiết kế test case và test, các tester truy cập thẳng vào bên trong source code, cấu trúc và thuật toán của chương trình để xác định xem đơn vị phần mềm thực hiện như thế nào?
- Người kiểm thử phải có kỹ năng nhất định để hiểu chi tiết về đoạn code cần kiểm thử
- Kỹ thuật này chủ yếu được dùng để kiểm thử ở mức đơn vị
- Thường được các dev thực hiện trong quá trình viết code ở giai đoạn kiểm thử đơn vị
  - Nguyên nhân: khi hệ thống được tích hợp hoàn chỉnh việc kiểm thử hộp trắng rất phức tạp và mất thời gian
- Chú ý: không phải dev nào cũng đồng ý cho phép tester join vào cấu trúc dữ liệu để test

Kiểm thử hộp trắng

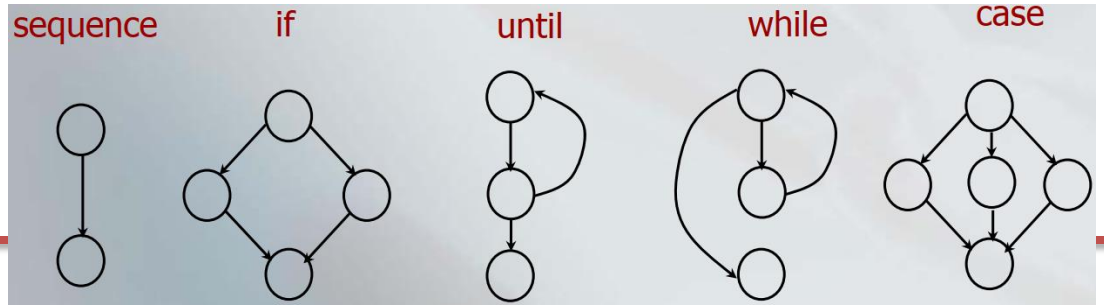
- Chia không gian thử nghiệm dựa vào cấu trúc của đơn vị cần kiểm tra:
  - Kiểm tra giao tiếp của đơn vị để đảm bảo dòng thông tin vào ra đơn vị luôn đúng.
  - Kiểm tra dữ liệu cục bộ để đảm bảo dữ liệu được lưu trữ trong đơn vị toàn vẹn trong suốt quá trình thuật giải được thực hiện.
  - Kiểm tra các điều kiện biên của các câu lệnh điều khiển, vòng lặp… để đảm bảo đơn vị luôn chạy đúng tại các biên này.
  - Kiểm tra để đảm bảo mọi con đường thực hiện phải được đi qua ít nhất một lần.
  - Kiểm tra sự thực thi của đơn vị trong các trường hợp ngoại lệ.

Kiểm thử hộp trắng

- Các kỹ thuật kiểm thử
  - Kiểm thử dựa trên đồ thị luồng điều khiển
  - Kiểm thử dựa trên đồ thị luồng dữ liệu
  - Kiểm thử đột biến (mutation testing)

Kiểm thử hộp trắng

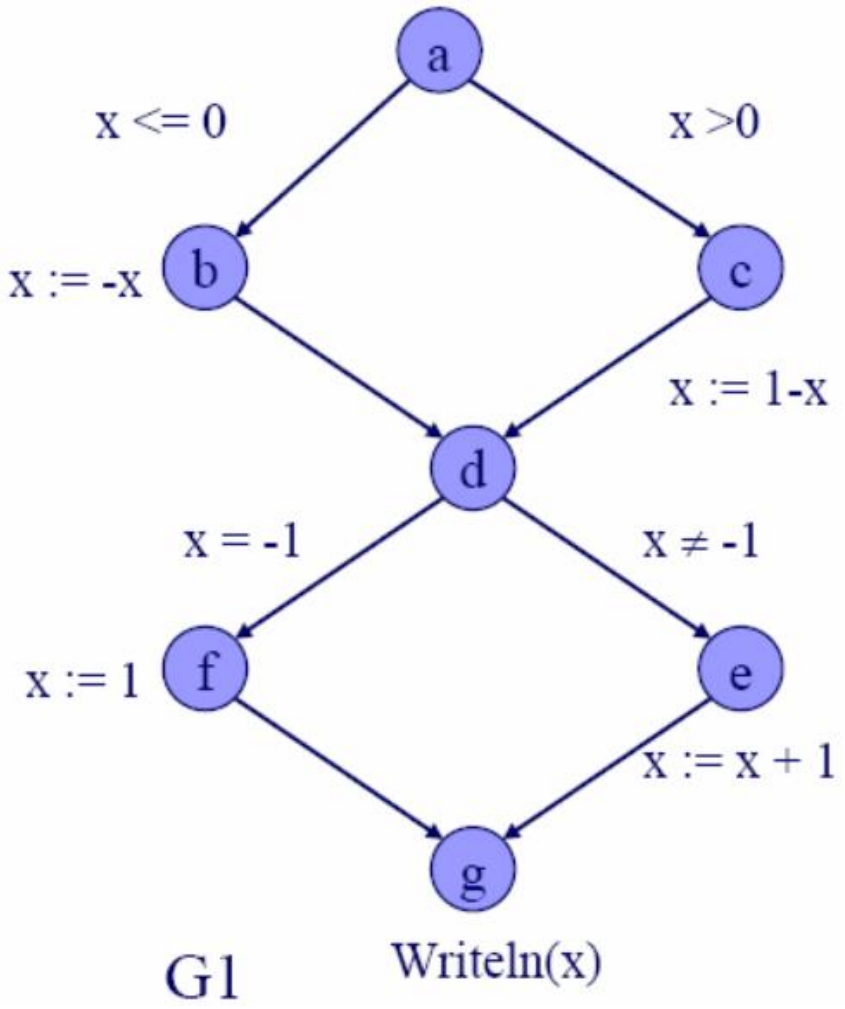
- Đồ thị luồng điều khiển
  - Đồ thị luồng điều khiển (Control Flow Graph) là đồ thị có hướng, biểu diễn một chương trình
    - Đỉnh: biểu diễn lệnh tuần tự hay khối lệnh
    - Cung: biểu diễn các rẽ nhánh
    - Một đỉnh vào và một đỉnh ra được thêm vào để biểu diễn điểm vào và ra của chương trình
  - Lộ trình (path)
    - Xuất phát từ đỉnh vào đi qua các đỉnh và cung trong đồ thị và kết thúc tại đỉnh ra



Kiểm thử hộp trắng

- Đồ thị luồng điều khiển
  - Ví dụ 1:

```
if x <= 0 then
        x := -x
else
        x := 1 -x;
if x = -1 then
        x=1
else
        x := x+1;
writeln(x);
```



  - Có 4 lộ trình
    - [a, b, d, f, g]
    - [a, b, d, e, g]
    - [a, c, d, f, g]
    - [a, c, d, e, g]

Kiểm thử hộp trắng

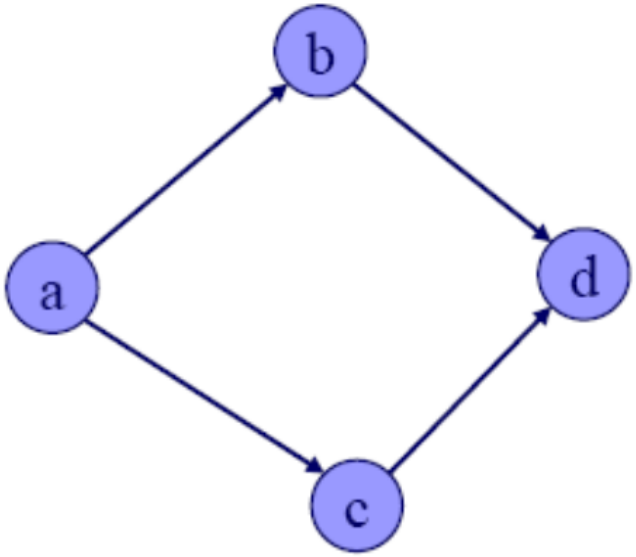
- Đồ thị luồng điều khiển: Ví dụ 1
  - Đồ thị G1 có thể biểu diễn dạng biểu thức chính quy:
    - G1 = abdfg + abdeg + acdfg + acdeg
  - Hay đơn giản:
    - G1 = a(bdf + bde + cdf + cde)g
    - G1 = a(b + c)d(e + f)g

Kiểm thử hộp trắng

- Đồ thị luồng điều khiển: Ví dụ 1

Biểu diễn các cấu trúc

Cấu trúc tuần tự: ab



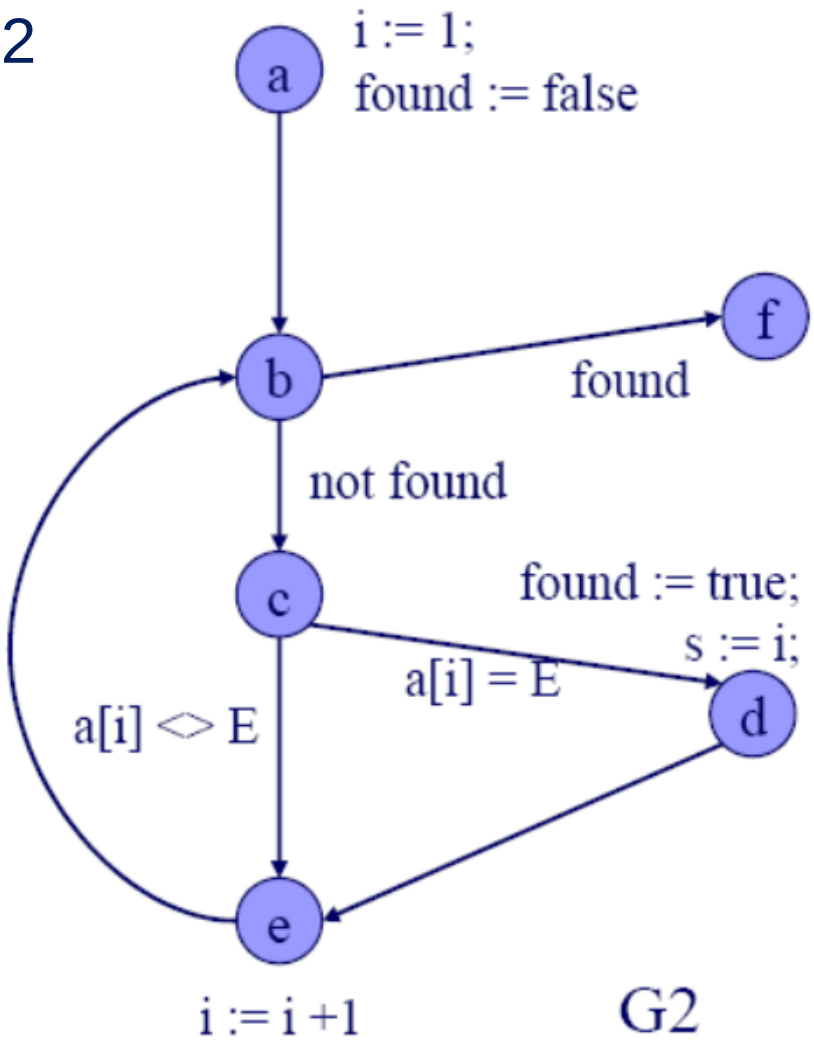
Cấu trúc rẽ nhánh: b(a + d)c

Cấu trúc lặp: ab(cb)*d

Kiểm thử hộp trắng

- Đồ thị luồng điều khiển: Ví dụ 2

```
i := 1;
found:= false;
while (not found) do
begin
if (a[i] = E) then
    begin
        found:= true;
        s := i;
    end;
    i := i + 1;
end;
```



– G2 = ab(c(e+d)b)*f

Kiểm thử hộp xám

- Là kỹ thuật kiểm thử kết hợp giữa kiểm thử hộp đen và kiểm thử hộp trắng
  - Quan tâm đến dữ liệu đầu vào, đầu ra và có đòi hỏi có sự truy cập tới cấu trúc dữ liệu
  - Mục đích: Nhằm tìm ra tối đa số lỗi về cấu trúc dữ liệu của hộp trắng cũng như lỗi chức năng của hộp đen.

Kiểm thử theo kịch bản

- Dựa vào các use-case để soạn ra các kịch bản
  - Ví dụ: một kịch bản cho hệ thống đăng ký môn học qua WEB
    - Login với username = “mã_sinh_viên”, password = “mật_khẩu”
    - Chọn chức năng đăng ký môn học
    - Chọn 5 học phần: CNPM, TTNT, XLTHS, PTTKHT, XSTK trong đó có 2 học phần trùng thời khoá biểu
    - Nhấnnút Đăng ký học
    - Chương trình phải báo lỗi và liệt kê 2 học phần bị trùng thời khoá biểu

Công cụ kiểm thử tự động

- Kiểm thử tự động dữ liệu:
  - Bộ sinh dữ liệu thử
  - Bộ xác minh kết quả
- Kiểm thử tự động cài đặt:
  - Bộ kiểm toán mã
  - Mô phỏng ứng xử của các module phụ
  - Bộ so sánh đầu ra
- Mô phỏng môi trường (ngoại lai).
- Bộ phân tích dòng dữ liệu (qui mô và tần suất).

4.3. Bảo trì phần mềm

- Sau khi phần mềm hoạt động, các lỗi được phát hiện, môi trường hoạt động thay đổi và yêu cầu người dùng mới phát sinh → cần phải tiến hành bảo trì phần mềm
- Mục tiêu:
  - Sửa đổi một phần mềm sau khi đã bàn giao để khắc phục các lỗi phát sinh, nâng cấp tính năng sử dụng, cải thiện hiệu năng của phần mềm.
  - Hoặc làm cho phần mềm có thể thích ứng trong một môi trường đã bị thay đổi.
- Bảo trì là cần thiết để đảm bảo rằng phần mềm có thể tiếp tục đáp ứng yêu cầu của người sử dụng

Các loại bảo trì phần mềm

- Bảo trì phần mềm được chia thành 4 loại:
  - Sửa lại cho đúng (corrective): là việc sửa các lỗi phát sinh trong quá trình sử dụng.
  - Thích ứng (adaptative): là việc chỉnh sửa hệ thống cho phù hợp với môi trường đã thay đổi.
  - Hoàn thiện (completive): là việc chỉnh sửa để đáp ứng các yêu cầu mới hoặc các yêu cầu đã thay đổi của người sử dụng.
  - Phòng ngừa (preventive): làm cho hệ thống dễ dàng bảo trì hơn trong những lần tiếp theo

Các đặc điểm

- 5 đặc điểm chính bao gồm các hoạt động của người bảo trì:
  - Duy trì kiểm soát các chức năng của phần mềm liên tục.
  - Duy trì kiểm soát việc sửa đổi phần mềm.
  - Hoàn thiện các chức năng hiện có.
  - Xác định các mối đe dọa an ninh và sửa chữa các lỗ hổng an ninh.
  - Ngăn ngừa việc xuống cấp hiệu suất tới mức không thể chấp nhận được.

Các vấn đề quan trọng

- Một số vấn đề quan trọng phải được xử lý để đảm bảo duy trì hiệu quả của phần mềm
  - Vấn đề kĩ thuật
    - Giới hạn về hiểu biết
    - Kiểm thử
    - Phân tích tác động
    - Bảo trì
  - Vấn đề quản lý
    - Liên kết với các mục tiêu của tổ chức
    - Nhân viên
    - Quy trình
    - Các khía cạnh về tổ chức bảo trì
    - Gia công phần mềm
  - Ước lượng chi phí
  - Đo lường

Các hoạt động lập kế hoạch

- Lập kế hoạch kinh doanh (cấp độ tổ chức)
- Lập kế hoạch bảo dưỡng (cấp độ chuyển tiếp)
- Lập kế hoạch yêu cầu thay đổi phần mềm cá nhân (cấp độ yêu cầu)
- Lập kế hoạch phát hành/phiên bản (cấp độ phần mềm)
  - Thu thập các các yêu cầu cá nhân.
  - Đồng ý với người dùng trên các nội dung của bản phát hành/ phiên bản kế tiếp.
  - Xác định các xung đột tiềm năng và phát triển các giải pháp thay thế.
  - Đánh giá nguy cơ của một thông cáo được đưa ra và phát triển một kế hoạch dự phòng trong trường hợp các vấn đề sẽ phát sinh.
  - Thông báo cho tất cả các bên liên quan

Bug

- Khái niệm:
  - Dùng để chỉ một lỗi nhỏ (lớn) trong quá trình kiểm thử phần mềm
  - Việc tìm ra bug khiến cho phần mềm càng được hoàn chỉnh, tránh sai sót tối đa khi đưa phần mềm ứng dụng vào thực tế
  - Bug phát hiện được cần phải liên hệ và phân công cho dev chịu trách nhiệm sửa lỗi này
- Khi Bug lỗi cần:
  - Thông tin đầy đủ để dev có thể hiểu được lỗi
  - Tester cho ý kiến về mức độ dự án để xác định mức độ nghiêm trọng của lỗi để fix lỗi
  - Xác định được lỗi ở chỗ nào
  - Có thể tái hiện lỗi khi cần
  - Mô tả các bước cần để tái hiện bug

Bug

Bug

- Dùng để chỉ một lỗi nhỏ (lớn) trong quá trình kiểm thử phần mềm
- Việc tìm ra bug khiến cho phần mềm càng được hoàn chỉnh, tránh sai sót tối đa khi đưa phần mềm ứng dụng vào thực tế
- Bug phát hiện được cần phải liên hệ và phân công cho Dev chịu trách nhiệm sửa lỗi này

📖 Bug

- Nghệ thuật gỡ rối
  - Gỡ rối là công việc khó khăn và dễ gây tâm lý chán nản bởi nguyên nhân gây ra lỗi nhiều khi lại mơ hồ:
    - Do timeout
    - Do độ chính xác
    - Do chủ quan lập trình
    - ...
  - Các hình thức gỡ rối (nên dùng kết hợp các hình thức này)
    - Brute force
    - Loại trừ nguyên nhân
    - Theo vết

📖 Câu hỏi thảo luận